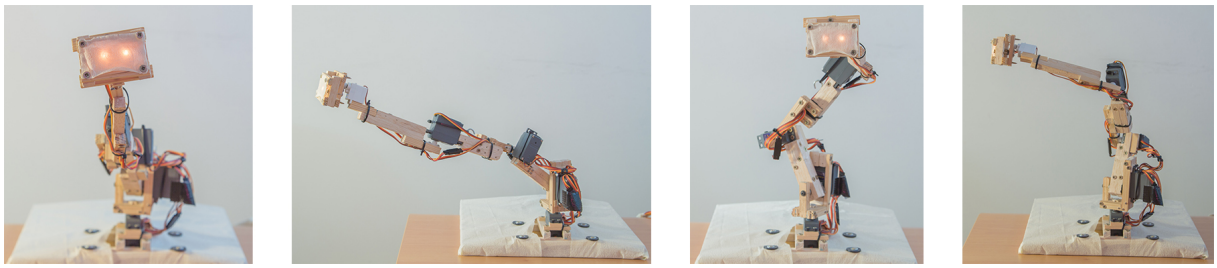
TÉCNICO LISBOA

UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO



Creating the Illusion of Life in Autonomous Social Robots

Tiago Guiomar Ribeiro

Supervisor: Doctor Ana Maria Severino de Almeida e Paiva

Thesis approved in public session to obtain the PhD degree in

Information Systems and Computer Engineering

Jury final classification: Pass with Distinction

2020

TÉCNICO LISBOA

UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Creating the Illusion of Life in Autonomous Social Robots

Tiago Guiomar Ribeiro

Supervisor: Doctor Ana Maria Severino de Almeida e Paiva

Thesis approved in public session to obtain the PhD degree in
Information Systems and Computer Engineering
Jury final classification: Pass with Distinction

Jury

Chairperson: Doctor José Manuel da Costa Alves Marques, Full Professor, Instituto Superior Técnico, Universidade de Lisboa
Members of the Committee:
Doctor Bram Vanderborght, Professor, Faculty of Applied Sciences, Vrije Universiteit Brussel and Planders Make, Belgium
Doctor Ana Maria Severino de Almeida e Paiva, Full Professor, Instituto Superior Técnico, Universidade de Lisboa
Doctor Duarte Nuno Jardim Nunes, Full Professor, Instituto Superior Técnico, Universidade de Lisboa
Doctor Alexandre José Malheiro Bernardino, Associate Professor, Instituto Superior Técnico, Universidade de Lisboa
Doctor Wendy Ju, Assistant Professor, Jacobs Technion-Cornell Institute, Cornell Tech, USA

Funding Institutions

Fundação para a Ciência e a Tecnologia, through the INESC-ID multiannual funding (PIDDAC), the project PEst-OE/EEI/LA0021/2013 and the grant SFRH/BD/97150/2013.
European Union through the 7th Framework Program (FP7/2007-2013) grant. 215554, the ICT-215554 project LIREC and the ICT-317923 project EMOTE.

2020

Dedicated to my parents,

who gave me the tools I needed to become an *inventor*, and also to my grandmother Avó Nela who has attentively been waiting for her first descendent to conquer a doctoral degree.

Acknowledgments

After eight and a half years working on my research, it is difficult to recall and mention every single person who has helped, pushed me in the right direction, or just stood there to listen while I debated to myself. I start by apologizing to anyone who I may have missed. My first thank you is for my advisor Ana Paiva for challenging and leading me into this PhD, and especially for giving me freedom to explore. In the midst of my confusion, she would enlighten me through words as simple as "do what you believe in". My major task force has lied within the GAIPS group at INESC-ID, and I will be unable to thank every single person. André Pereira and Iolanda Leite welcomed me into the 2n7.29 office in 2011 and taught me how to become a scientist and a roboticist. To them I owe a lot of mentorship and fraternity, so know that this thesis is also for you. Thank you Sofia for always listening and providing your fair judgement. Thank you Filipa for being an attentive learner and taking over some of my roles in the group. And thank you Patrícia for everything you taught me about psychology, interaction design and thinking out of the box - or out of engineering. Your insights and directions helped me to raise my standards on my final achievements and results. Nearly every single person at GAIPS contributed to the accomplishment of my work. I must thank all the professors, namely Francisco Melo, Carlos, Rui and João, and also Samuel and Joana, for the constructive discussions. An additional thanks to Eugenio who was part of a considerable portion of my work, and to Marco and Guilherme for the push-start they gave me. Thank you to all the other PhD and MSc students who used my work and made it a substantial contribution. From GAIPS I draw a last big thank you to Sandra Sá who was always there whenever I needed to manage logistics, journeys, shopping for equipment, and nearly any event or experiment in which I've participated. Without her the castle would just fall apart. I thank my family, namely my parents and my sisters Sara and Guida who always believed in me and supported me, and my grandmother Avó who's hope of seeing me through this period has always motivated me to keep moving forward and complete this thesis. Thank you Telma for standing by my side every day whether I'm on a workaholic spree, stressed to conquer a deadline or depressed whenever anything went wrong. Your words and your smile gave me the comfort I needed to never give up on any ambition that I've set to achieve. Within my dear friends I draw a particular thank you to Sergio Almeida, Pedro Lopes, Daniel, Fábio, Tânia, Ricardo, Marta, Vasco, Dário, Nuno Teixeira, Nuno Marques and Vivi who followed closely, and so many others for whom infrequent encounters did not impair their support, such as Ross Mead, Joana Botelho, Filipa Nunes, Kim Baraka, Vanessa Pedreiro and Vítor Abrantes. Many other people have structurally supported me, such as Júlia Oliveira from the academic services, Luís Revez and all of IST's GOP members for helping to manage experiment locations. I further extend a warm thank you too all the participants I had in my experiments, to CENTRA - Center for Atrophysics and Gravitation for giving access to their HPC cluster, and all the collaborators from the LIREC and EMOTE projects. I also thank the IST, INESC-ID and Fundação para a Ciência e a Tecnologia, and additionally all the committees from every conference I have participated in, including everyone who has reviewed my papers and contributed to make my work better, plausible, and achievable. Final thank you goes to Erik Satie, Claude Debussy and Pink Floyd in particular for inspiring me through the toughest times, but also every other artist who's music has accompanied me in my incredible journey.

Resumo

Os robôs estão-se a tornar numa nova forma de personagens animados, e estão a ser implantados na nossa sociedade para serem utilizados em diversas aplicações sociais que podem beneficiar do uso de tecnologia e de inteligência artificial, tal como as áreas de educação, entretenimento, ou de assistência de vida. Esta tese explora como é que tais robôs sociais, através do seu corpo fisicamente expressivo, e considerando as suas capacidades autónomas, poderão exibir a ilusão de vida tal como os personagens do cinema, enquanto interagem com humanos. Em particular, estamos interessados em trazer teorias e práticas da área de animação de personagens, e de desenvolver métodos e tecnologia que permitam que animadores tomem um papel estrutural no desenvolvimento de robôs sociais autónomos. Estabelecemos uma nova forma de animação, que designamos de *animação de robôs*, que pretende transferir conhecimentos e técnicas de animação tradicional e de CG para a área da robótica social. Estabelecemos também uma lista de princípios de animação de robots, baseados nos princípios de animação da Disney. O modelo e metodologia SERA foi criado para dar suporte à criação de robôs autónomos socialmente expressivos, assente numa metodologia centrada no utilizador, e de forma a incluir peritos não-técnicos tal como psicólogos ou animadores. O inovador motor de animação Nutty Tracks foi criado para permitir a combinação, durante a interação, de animações e posturas desenhadas por artistas, com movimento que é gerado em tempo-real. O Nutty Tracks estabelece uma ponte entre o nível simbólico de um agente inteligente, com a geração de movimento e de controlo de mais baixo nível, permitindo aos robôs exibir a ilusão da vida de forma que os utilizadores sejam capazes de entender as suas intenções comunicativas. Para suportar robôs articulados complexos tais como manipuladores industriais, criámos o ERIK, uma técnica de cinemática expressiva que mistura cinemática inversa (CI) com controlo postural através de cinemática direta (CD). Criámos ainda o Nutty Motion Filter, que permite interpolar e suavizar um sinal de movimento em tempo-real, de forma a respeitar as limitações cinemáticas de juntas ou de movimento espacial, fornecendo parâmetros que permitem ajustar a expressividade do sinal resultante. Diversos casos de uso são apresentados, que utilizam diferentes robôs. Em particular, desenvolvemos o cenário Ahoy no qual humanos participam num jogo de mímica com o robô artesanal Adelino. O robô faz uso da sua postura expressiva para dar pistas aos jogadores, enquanto mantêm o seu olhar na sua direção. De forma semelhante, o cenário AvantSatie foi criado, no qual o Adelino participa como um companheiro autónomo num jogo de piano. Os estudos com utilizadores demonstraram que os participantes foram capazes de entender a intenção do robô, mesmo que a solução em tempo-real do ERIK exibisse uma postura ligeiramente distorcida devido ao sistema resolver simultaneamente para os objetivos de orientação e postural. No longo prazo, as nossas teorias, métodos e técnicas estabelecem os alicerces para a criação de robôs autónomos expressivos, capazes de exibir a ilusão de vida através de uma abordagem com artistas, enquanto os mesmos interagem com humanos e o seu ambiente envolvente.

Palavras-chave: Animação de Robôs, Interação Pessoa-Robô, Robótica Social, Cinemática Expressiva, Cinemática Inversa

Abstract

Robots are becoming a new form of animated characters and are being deployed into our society to be used in various social settings that can benefit of the use of technology and artificial intelligence (AI), such as education, entertainment or assisted living. This thesis explores how such social robots, through their physically expressive embodiment, and considering their autonomous capabilities, may be able to convey the illusion of life just as movie characters do, while interacting with humans. In particular, we are interested in bringing in theories in practices from the field of character animation, and to develop methods and technology that will allow animation artists to take a structural role on the development of autonomous social robots. We establish and describe a new form of animation, called *robot animation*, which sets to bring the existing knowledge and techniques from traditional and CGI animation, into the field of social robots. Along it we have outlined a list of principles of robot animation, based on the original principles of animation from Disney. The SERA model and methodology was created to support the creation of autonomous socially expressive robots, which relies in user-centred design and includes non-technical experts such as psychologists and character animators. An innovative animation engine called Nutty Tracks was created to support the blending, during interactions, of animations and postures pre-designed by artists, with motion that is procedurally generated in real-time. Nutty Tracks bridges the symbolic level of an autonomous artificial intelligence agent, with the lower level of motion generation and control. This allows us to create autonomous social robots that can convey the illusion of life, in a way that users are also able to understand its communicative intentions. In order to support complex, articulated robots such as industrial manipulators, we have created ERIK, which is an expressive kinematics technique that acts by bring together inverse kinematics (IK) control with forward kinematics (FK) control. We add to that the Nutty Motion Filter, which is a signal processor that allows to interpolate and smooth a motion signal in real-time in order to comply with mechanical and kinematic limits of joints or spatial motion, while providing parameters that allow to tweak the expressivity of the resulting motion. Various use-cases are presented using different robots. In particular, the Ahoy interactive scenario was developed in which humans play a game of pantomime with the custom-built Adelino robot. The robot could use its expressive posture to convey hints to the players, while keeping an orientation constraint towards their face. Similarly, the AvantSatie scenario was created in which Adelino acts autonomously as a piano-game companion. User studies showed that the participants were able to decode the intention of the robot even if the ERIK solution, running in real-time, was slightly distorting the pre-designed postures in order to solve simultaneously for both orientational and postural goals. The results provide evidence that expressive postures (controlled using FK) could be used along with IK in order to provide arbitrary robots with an animation model that works *out-of-the-box*, with nearly no tweaking. In the long term, our theories, methods and technique pose as the foundation towards autonomous expressive robots that exhibit the illusion of life through an artist-enabled approach, while interacting with humans and their surrounding environment.

Keywords: Robot Animation, Human-Robot-Interaction, Social Robotics, Expressive Kinematics, Inverse Kinematics

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The art of animation was born more then one hundred years ago in 1896, when Georges Méliès invented the stop-motion technique. Twelve years later, Èmile Cohl became the father of animated cartoons with 'Fantasmagorie'. Windsor McCay, however, was coined as the father of animated movies for his 1911 work entitled 'Gertie the Dinosaur', in which he created what is considered to be the first animated character to actually convey emotions and an appealing personality [1]. Animation movies started to drive the attention of the audiences by providing compelling stories with rich new characters, each tailored to every story and the audience it aimed at.

These hand-drawn animated characters have been evolving since the early days, following the rise of major studios such as Fleischer Studio (e.g. 'Popeye the Sailor Man','Betty Boop' [2]), Pat Sullivan Studio (e.g. 'Felix the Cat' [3]), and of course, Walt Disney Studios (e.g. 'Steamboat Willie', 'Snow White and The Seven Dwarfs' [4]). Some individual animators also had major influence even working between different studios, such as Tex Avery and his 'Looney Tunes' characters [5].

Now recently, during the last thirty years, animated characters have become mainly computer-animated. Pixar Animation Studios, part of the Walt Disney Company, stands for most people as the world's major animation studio, competing with other studios like DreamWorks or Blue Sky Studios.

Walt Disney Animation Studios and Pixar's chief creative officer (formerly John Lasseter, now Jennifer Lee and Pete Docter) has stood through the last decades in the place where Walt Disney himself once stood - leading teams of some of the best artists in the world to create critically acclaimed animation films such as 'Toy Story', 'Monsters, Inc.', 'Tangled', 'Big Hero 6' or 'Frozen'. Two of Pixar's most popular films are 'WALL-E', which features a highly expressive animated robot as the main character, and 'Big Hero 6' which features a inflatable healthcare robot hero. All these characters were artistically crafted using computer graphics (CGI) and design techniques, in order to convey the illusion that they are alive.

Currently however, robots are becoming a new form of animated characters in order to be used in social applications backed up by technology and artificial intelligence (AI), in fields such as education, entertainment or assisted living. Such robots are physically embodied interactive agents, and as such, in the light of this thesis, rely deeply on the concept of *believable agents* (or characters) as described by Bates [6]. Bates has provided an

influential discussion of the creation of such *believable agents*, which in turn, are based on the notion from the arts of *believable character*, as "*one that provides the illusion of life, and this permits the audience's suspension of disbelief*" [6].

Moving into the scope of this thesis, *social robots* are defined by Breazeal as a class of robots to which "*people apply a social model to, in order to interact with and to understand*" [7]. Bartneck & Forlizzi have also defined a social robot to be "*an autonomous or semi-autonomous robot that interacts and communicates with humans by following the behavioral norms expected by the people with whom the robot is intended to interact*" [8]. In a more technical interpretation, social robots can be seen as a new form of human-computer interface, that provides the computer part with a physically expressive, active and perceptive embodiment, through which a sociable artificial intelligence agent engages in an interactive application with the human user and its surrounding environment.

Considering the social, communicative, autonomous, and believable aspects of social robots, our work is directed towards how they can culminate as actor-interfaces for the users, through the means of animation theories and practices applied to robots.

1.2 Research Goals

The ultimate goal of our work is to understand how these social robots, through their physically expressive embodiment, and considering their autonomous capabilities, may be able to convey the illusion of life just as movie characters do, while interacting with humans in/and their environment.

The key to this goal is in establishing a new form of animation, called *robot animation*. In the context of social robotics, our understanding is that *robot animation* is more than just making the robot move. It is about making the robot *seem* alive while interacting with humans in particular tasks or applications. Van Breemen had initially defined animation of robots as "*the process of computing how the robot should act such that it is believable and interactive*" [9]. It seems relevant however to note that what markedly distinguishes an animated robot from, e.g. a virtual animated robotic character, is the fact that the robot exists in our physical world. Moreover, we consider the concept of *robot animation* to be especially directed towards interactive applications where autonomous robots are controlled by an AI. In contrast, we still consider animatronics as the ability of making a robot move, following some predefined trajectory (e.g. for film or live performances), and following on traditional animation principles.

Upon assimilating all the social aspects, we therefore complement Van Breemen's definition by stating that *robot animation* consists of the *workflow and processes that give a robot the ability of expressing identity, emotion and motivated intention during autonomous interaction with human users*. The key words, in this definition, that guide our stance, are *expressing* and *autonomous*, i.e. *robot animation* is closely related to autonomous expression. The idea behind expressing *motivated intention* is that an animated robot should be able to portray its motivation (i.e. *story*, purpose of existence), throughout its actions, in a way that the human interactors are able to understand its underlying intentions, and therefore to interpret the robot's motivation during their interaction.

1.3 Contributions

With this thesis, we expect to enrich the field of robot animation and in particular, animation of autonomous social robots, by making the following contributions:

The Principles of Robot Animation that outline how traditional principles of animation can be adopted, adapted and used with autonomous social robots in order for them to convey the illusion of life;

The ERIK algorithm that allows to animate arbitrary articulated structures using forward kinematics and inverse kinematics simultaneously, allowing an endpoint to orient towards a target direction while the whole body is used to convey an expressive posture;

The Nutty Tracks Workflow and Programmable Animation Pipeline that provides non-linear animation capabilities to an expressive robot, by allows it to convey expressivity through a blend of pre-designed animations and postures, and procedural motion;

The SERA model that establishes tools and a methodology for creating autonomous socially expressive robots using re-usable components;

Robot Animation uses cases featuring the craft-built Adelino robot, to demonstrate how such contributions may be used within an autonomous social robot application, which in our cases are directed at entertainment.

Throughout the development of our work, we have authored and co-authored over 30 publications across a wide variety of conferences and workshops, mostly of high reputation. In particular, we highlight the following ones:

- Ribeiro, T. & Paiva, A. (2012). The Illusion of Robotic Life: Principles and Practices of Animation for Robots. *In ACM/IEEE International Conference on Human-Robot Interaction - HRI '12*, pp. 383–390, Boston, MA, USA. *** Best paper nominee**.

- Ribeiro, T., Dooley, D. & Paiva, A. (2013). Nutty Tracks - Symbolic Animation Pipeline for Expressive Robotics. *ACM International Conference on Computer Graphics and Interactive Techniques Posters - SIGGRAPH '13*, Anaheim, CA, USA. *** 3^{rd} place in the Student Research Competition**.

- Ribeiro, T. & Paiva, A. (2015). Creating Interactive Robotic Characters. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction - HRI Pioneers Workshop*, 215–216, Portland, OR, USA.

- Ribeiro, T. & Paiva, A. (2017). Animating the Adelino Robot with ERIK. In *proceedings of the ACM International Conference on Multimodal Interaction (ICMI'17)*, pp. 388–396, Glasgow, UK. ACM.

- Ribeiro, T. & Paiva, A. (2019). Expressive Inverse Kinematics Solving in Real-time for Virtual and Robotic Interactive Characters. arXiv preprint: cs.RO/1909.13875.

- Ribeiro, T. & Paiva, A. (to appear). The Practice of Animation in Robotics. In *Noceti, N., Sciutti, A., Rea, F. (Eds.), Modelling Human Motion*. Springer. ISBN: 9783030467326. *** Extended preprint published to arXiv titled *Nutty-based Robot Animation - Principles and Practices.***

- Sequeira, P., Alves-Oliveira, P., Ribeiro, T., Di Tullio, E., Petisca, S., Melo, F. S., ...Paiva, A. (2016). Discovering social interaction strategies for robots from restricted-perception wizard-of-oz studies. In *proceedings of the* 11^{th} *ACM/IEEE International Conference on Human-Robot Interaction (HRI'16)*, pp. 197–204, Christchurch, New Zealand. ACM. * **Best paper award**

- Paiva, A., Leite, I. & Ribeiro, T. (2014). Emotion Modelling for Social Robots. In *Calvo, R., D'Mello, S., Gratch, J., & Kappas, A. (Eds.), The Oxford Handbook of Affective Computing*. Oxford University Press. ISBN: 9780199942237.

1.4 Outline

This document is organized as follows. In the next chapter we introduce some theoretical background on expression of emotions in humans, and on character animation. In particular we describe the popular Twelve Principles of Animation from Disney, which have deeply inspired our work. Chapter 3 presents a review of the related work that has motivated or contributed to our research and development. In Chapter 4 we present our own theoretical foundations for creating the Illusion of Life in autonomous social robots, including our Principles of Robot Animation, the Dimensions of Kinematronics, and the Nutty Workflow. Chapter 5 introduces our general practices for developing robot animation in HRI scenarios, including the SERA model for building autonomous socially expressive robots, the Nutty Pipeline and discusses the use of tools for the animation of social robots. Chapter 6 presents the main technological contributions of the thesis, including Nutty Tracks, the Nutty Motion Filter, and ERIK, our expressive kinematics algorithm that allows any articulated robot to be expressive while interacting with humans. The next chapter presents various HRI scenarios in which both the SERA architecture, Nutty Tracks and ERIK were used. This chapter also introduces Adelino, an expressive robot that was built in order to challenge and test our work. Finally, Chapter 8 wraps up the thesis and provides some future work directions.

Chapter 2

Background

While this thesis focuses on the animation and expression of robots, our purpose it to provide such robots with meaningful expressive behaviour during interactions with human beings.

As such interactions are heavily based on both socially and emotionally expressive behaviours, we start by describing some of the theories of human emotion and expression that can inform the design of robots' socially expressive behaviours. These theories also support us in designing emotion. Product design is an example of a field that has also struggled between creating art/emotion and functional objects [10]. Design tells us how to communicate, and we want to understand how to enhance the communication of emotions by robots through animation. Hess [11] states that 'the ability to well communicate emotions is relevant for both the encoder, who would like to be understood, and the decoder, who strives to understand'.

Because animation is a cornerstone of this thesis, we complement this chapter with an overview of the major character animation theories and practices that have guided our work since the beginning.

2.1 Human Emotion and Expression

There is no general definition, classification or computational model for human emotions. In this section we present some of the most popular models, in order to understand how they connect with the expression of emotion. Considering such models within our work is relevant because social robots are generally backed by an affective and emotion-enabled artificial intelligent agent. Such socially- and emotionally-intelligent agents (SEIAs) may be build based on human emotion theories in order to provide more believable and adaptive social interactions with humans. Thus, when designing non-verbal behaviour mechanisms and expressions for a social robot, it is suitable to understand how such behaviour is actually linked with the emotional models that typically underlie SEIAs.

FACS by Ekman and Friesen [12] is one of the most referenced models of emotional expression. They argue that humans can universally recognize six basic emotions through facial expression: anger, disgust, fear, joy, sadness and surprise. This model is very popular in both character animation and in computation, because it is based on a small set of discrete emotions. It therefore provides an easy model on which to develop agents and characters, and is especially aimed at providing a legible visual interface to non-expert human users.

Ortony, Clore and Collins developed the OCC model [13], which defines 22 different emotional categories.

However, due to the complexity of the model, Bartneck [14] has developed another model that allows to use the OCC theory in emotional expression in a five-phase process: **Classification** What do I feel about what just happened?
Quantification How much do I feel about it?
Interaction How does this affect what I was already feeling?
Mapping What should I do to express this feeling?
Expression How should I do that?
Bartneck also suggests that a mapping from OCC to Ekman's model is possible, but not trivial. Within this model, the focus of our work mostly addresses the Mapping and Expression phases.

The Pleasure-Arousal-Dominance model (PAD) is another popular model, proposed by Mehrabian [15]. The PAD model is actually a three dimensional space in which each emotion is defined by its position in Pleasure, Arousal, and Dominance coordinates. An advantage of this model is that it is adequate for computation, as emotions can seamlessly transition from one to another through interpolation. However, depending on the emotions, the transition from A to B might actually go through an emotion C which might both be invalid within the context, or be unnatural for expression.

Various authors have published extensive and influential works on human non-verbal communication. Because this thesis will focus more on theories from character animation, and not ones from human models of expression, we will make only a mention to some of the works that we consider most relevant. Both Ekman and Allwood have attempted to understand how to generalize the description of human behaviour, which can be informative for the development of behaviour for interactive characters [16, 17]. Other authors such as Argyle and Wallbott have explored social and emotional non-verbal communication through bodily motion and posture [18, 19]. Argyle's work does however extend beyond human emotional expression, by including non-verbal communication in animals and other aspects of appearance such as clothing. Although it also includes a description about the use of gestures, we cannot refrain from mentioning Kendon's work as a leading authority on the topic of semiotics and gesture studies [20].

2.2 Character Animation Theory and Practices

We seem to know when to 'tap the heart'. Others have hit the intellect. We can hit them in an emotional way.

Walt Disney

Driving inspiration for the animation of socially intelligent characters from traditional animation has become a common practice. Bates claims that insights from character animation literature such as 'The Illusion of Life' [4] may provide key information for building computational models of *believable interactive characters* (either virtual or robotic), by also arguing that "*while scientists may have more effectively recreated scientists, it is the artists who have come closest to understanding and perhaps capturing the essence of humanity*" [6].

The same happened upon the emerging of computer animated cartoons. At that time, animators exploring the new technique also felt the need to look into what had already been done during the last decades, and discover how that knowledge could be adapted for computer animation. On that topic, Lasseter argued that the traditional principles of animation have a similar meaning across different animation medium [21].

Disney's twelve principles of animation are considered by most, to be the commandments of animation. They are a result of more than 60 years of Disney productions, and were compiled into a book called 'The Illusion of Life', by Thomas and Johnston, the last two of Disney's Nine Old Men [4] [1].

We have looked into each of these principles of animation, and analyzed what they can mean and how they can be used on robot animation.

2.2.1 Disney's Twelve Principles of Animation

We present here a small summary of the original Twelve Principles of Animation defined in 'The Illusion of Life'. We will further extend and relate them to robot animation4.1.

Squash and Stretch states that characters should not be solid. The movement and liquidness of an object reflects that the object is alive, because it makes it look more organic. If we make a chair squash and stretch, the chair will seem alive. One rule of thumb is that despite them changing their form, the objects should keep the same volume while squashing and stretching.

Anticipation reveals the intentions of the character, so we know and understand better what they are going to do next.

Staging is the way of directing the viewers attention. It is generally performed by the whole acting process, and also by camera, lights, sound and effects. This principle is related to making sure that the expressive intention is clear to the viewer. The essence of this principle is minimalism, keeping the user focused on what is relevant about the current action and plot.

Follow-Through and Overlapping Action are the way a character, objects or part of them inertially react to the physical world, thus making the movements seem more naturally and physically correct. An example of Overlapping action would be hair and clothes that follow the movement of a character. Follow-through action is for example the inertial reaction of a character that throws a ball. After the throw, both the throwing arm and the whole body will slightly swing and tumble along the throwing direction.

Straight Ahead Action and Pose-to-Pose is about the animation process. An animator can make a character go through a sequence of well defined poses (Pose-to-Pose action), or sequentially draw each frame of the animation without necessarily knowing where it is heading (Straight-Ahead action).

Slow In and Slow Out is how the motions are accelerated (or slowed down). Characters and objects do not start or stop abruptly. Instead, each movement has an acceleration phase followed by a slowing down phase. Slow out can be confused with follow-through; however, follow-through extends the action, while the slow-out finishes it smoothly. A movement should not start or stop suddenly, it should always have some acceleration within, unless it is clearly intended not to.

Arcs draw the trajectories of natural motions, making them feel less machine-like and more natural and organic. An example is a head that gazes from left to right. A robotic movement would make the head rotate only

[1] A group of nine animators that worked closely with Walt Disney since the debut feature Snow White and the Seven Dwarfs (1937) and onto The Fox and The Hound (1981).

along its vertical axis. A natural movement will make the head slightly lean up or down towards the midpoint of the trajectory while rotating.

Secondary Action is an action that does not contribute directly to the expression of an action, but adds personality and lifelikeness. An example would be breathing, blinking the eyes, or holding and scratching different parts of the body.

Timing is a dual principle that focuses especially on two different things. First, it can change how users perceive the emotion of a motion or the physical world in which the character exists. Second, it also relates to the story, and how the story is being told. It is about how the character pauses between the actions, and how it synchronizes to itself and the surroundings.

Exaggeration makes relevant features more wild and relevant, and is what makes the characters behave as cartoons, as opposite to the dull motion of humans in the real world. An example would be popping out the eyes when startled, or growing a huge red tomato-like head while shouting.

Solid Drawing is about correctly balancing volume and weight of characters and objects. It also warns against symmetric characters and expressions. Characters do not stand stiff and still, unless that is what they are intended to portray.

Appeal of a character is how it expresses and asserts its role, personality and relevance in a story. It is possibly the most subjective principle, as it also relates to how the character can make the viewers believe in its story.

2.2.2 Inspiration from TV Cartoons - Warner Bros., and MGM and FOX

Since The Golden Age of American Animation, Warner Bros. and MGM animators definitely marked their position as masters of animated cartoons. Although Exaggeration, for example, is already described in the Disney's list, these animators took it to another level, by given special focus on physical exaggeration, in which we can actually identify common subtypes of exaggeration, like extreme distortion or blowing-ups. Most of their animations were largely based on comic plots, which generally included sever physical damage to the characters, thus justifying why they developed so much into blowing-ups and heavy distortion of the characters' body.

While we do not want to blow up or physically damage robots while animating them, some of these practices can still provide interesting tips on some specific domains, like robots aimed at entertainment. While entertaining, we want a character to be as much expressive as possible, so entertainment robots will more likely promote the interest for developing and incorporating behaviors and mechanisms inspired by this kind of animation.

Tex Avery, one of the greatest animators of all time, coined the 'Tex Avery Expression', or just a 'Tex Avery', which is a very know eyes-popping-out expression generally used in fear or surprise situations [5]. The EMYS robotic head is an example of how an eyes-popping mechanism can be incorporated into a robot (Figure) [22].

Another common trait is that each character was made very unique and well adapted to its role (principle of Appeal). Some of the most popular characters created during this time were Bugs Bunny, Daffy Duck, Porky Pig, Elmer Fudd, Yosemite Sam, Tom and Jerry, Scooby Doo and Droopy [23]. They usually carry or use regular props that people end up associating with that character, independently of the plot. Most of them also feature unique catchphrases and often perform secondary action that helps to define the personality of the character they convey.

All these features together contribute to the illusion of the character as a being, and to the reinforcement of the connection between viewers and the characters.

Chuck Jones was one of the major animators from Warner Brothers (and later MGM), and has described animation at Warner Brother as "*Believability. That is what we were striving for ... belief in the life of the characters. That, after all, is the dictionary definition and meaning of the word 'animation': to invoke life*" [24]. This definition was also cited by Bates on his seminal paper on *believable agents* [6].

Unfortunately, the practice of these animators is not so well documented as the ones from Disney. As they were generally jumping around from one studio to another, each animator may have followed different guidelines along his career, so there are no compiled guidelines to describe their creative process. However, by viewing their work it is clear that some common traits were followed, just like in the case of extreme exaggeration or the development of characters that we described.

2.2.3 Puppet Animation

If we are looking at different kinds of animators to draw inspiration from, we must take a look at a genre that actually shares some practical obstacles with robot animation. Puppets are physical characters that are built in order to move and be expressive, and are subject to the laws of physics of our real world. If we replace the word 'Puppets' with 'Social Robots' in this last sentence, it would still be valid.

Puppet animation grew especially popular with Jim Henson's 'The Muppet Show' [25]. Henson's puppets (Figure 2.1) are generally very simple in movement. Most of them can only open and close their mouth, and wave their arms and body. But by developing their own non-verbal language, animators were able to portray all kinds of different plots with them. By watching episodes of the series we can find that whenever a muppet wants to close its eyes, it will cover them with their hands, as the eyes cannot gaze or shut. This kind of tricks is very inspiring for robot animation.

It is empirically clear that if a character has only a mouth that can open and close, it is impossible to portray emotion by using just its face. That is where animation takes place. Most of the emotional expressions we find in puppets comes from the movement, and not just the poses.

There is no defined happy pose for a muppet. Instead, there is a bouncy movement with the arms waving around, that elicits the feeling of excitement and happiness. For fear, the mouth will tremble a lot, and the muppet will probably cover its eyes and assume a posture of withdrawal. An angry expression is achieved by leaning the muppet against the object or character of hate, closing its mouth, and pulling back its arms.

Another interesting feature in puppets is that if correctly designed, they can benefit of the 'Noh mask' effect [26]. These are traditional japanese masks used in Noh drama [27].

Although the mask does not change in shape, it is designed to convey a different emotion depending on the angle at which it is viewed. When the carrier titls the head downwards, the mask is viewed as a happy face, while tilting it upwards conveys a sad, or angry expression (depending on the design and purpose of the mask).

As in most inspiration from art, the best way to learn their principles and practices of puppet animation is by watching the episodes and using them as reference footage.

Figure 2.1: The Muppet Show's Kermit the Frog.

Figure 2.2: The Noh mask effect [26].

2.2.4 Animation Curves

Animation Curves are tools that are particularly important for animators. An animation curve exists for each degree of freedom (DoF) that is being animated in a character, and it shows how that specific DoF varies over time [28].

Figure 2.3 shows the animation curve for the translation DoF of a hypothetical drag race car. In a drag race, the race car only drives forward at full speed. Because this animation curve shows the position changing over time, the speed of the car at some point of the curve is actually the tangent to the curve on that point (the first derivative). The second derivative (the rate of change of the tangent) thus represents the acceleration of the car.

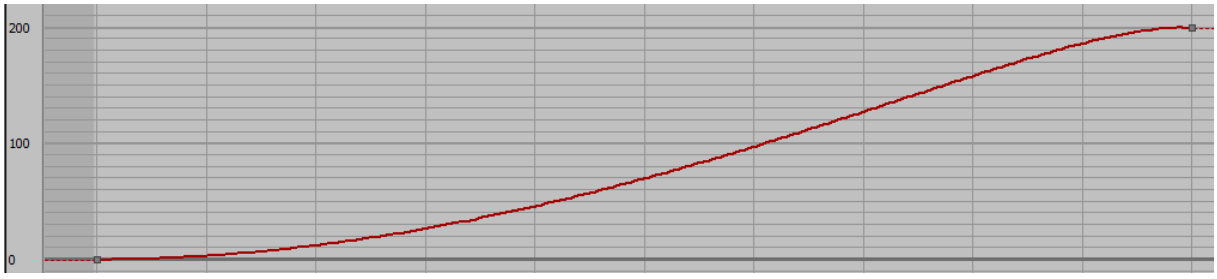


Figure 2.3: The animation curve of the translation of a drag car accelerating until it reaches a top speed, and then decelerating until it halts.

By analyzing the curve, we see that the car starts by accelerating until about halfway through, when it reaches

its maximum speed. We notice this because during the first part of the curve there is an accentuated concavity. Once the curve starts looking straight, the velocity is being kept nearly constant. In the end the car decelerates until it halts.

Animation curves can also be used to represent Rotation or even Scaling. Figure 2.4 shows the animation curve of the rotation of the pivot of a pendulum that is dropped from a height of 20 degrees. It then balances several times until it stops.

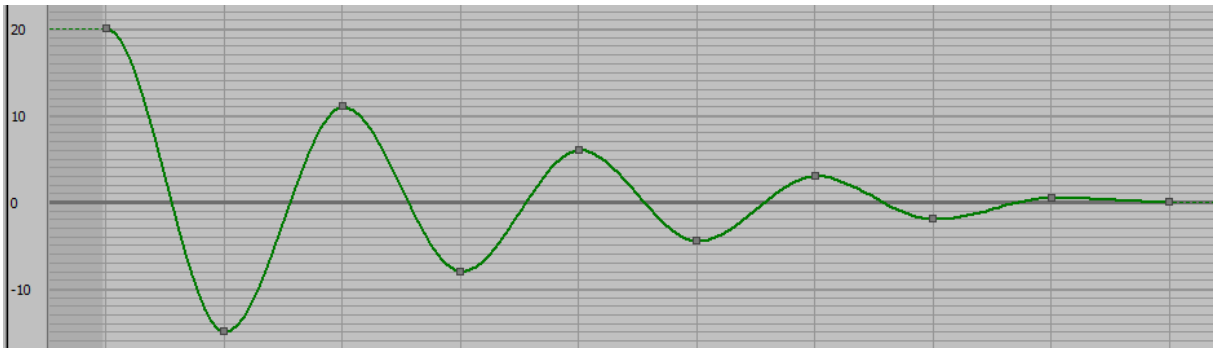


Figure 2.4: The animation curve of the rotation of a pendulum that is dropped from 20 degrees and balances until it stops.

In this curve we see some grey squares where the curve changes. These squares are actually keyframes that were used to design the animation. The curve is a spline interpolation of the movement between these keyframes.

By looking at each keyframe, we see that that the angle goes from 20 degrees to -15, then to 11 and so on. Just like in the translation animation curve, the tangent of this curve also represents the velocity of rotation.

If we imagine the pendulum going through the lower-most position of its trajectory (which is the position in which it travels faster), that point would correspond to the 0 degrees line, thus making sense that the curve in this point is steeper than in the rest of the trajectory. As the pendulum loses energy and balances less, the steepness is also lower, which also reflects in a lower speed.

Animation curves thus stand as a very important tool for analyzing and adjusting animations. They can also be computationally processed just like a signal, in order to warp the animation and create animation effects.

During the last decade, robotic platforms have been evolving in a way that we now have full robotic characters with a large number of degrees of freedom, which brings their expressiveness closer to what we may find in animated cartoons. Besides inspiring the design of robots, animation has also already inspired the design of robot animation [29, 30, 9, 31, 32, 22].

Chapter 3

Related Work

3.1 Interactive Embodied Characters

One of the major fields that grounds this thesis, and has performed many developments of interest for the creation of autonomous socially expressive robots, is the field of Intelligent Virtual Agents (IVAs). This field, however, is also commonly associated with other sub-fields or adjacent fields such as Socially Intelligent Agents or Embodied Conversational Agents (ECAs). All of them are generally associated with virtual agents.

Both Bates and Reilly have extensively described the concept of *believable agents* (or *characters*) [6, 33]. Reilly in particular, argues that "*the problem of creating believable agents lies somewhere between the arts and artificial intelligence*" [33]. He further adds that "*Artists know how to create believable characters*", and that "*AI researchers know how to create autonomous agents*", thus "*joining these two disciplines allows to produce autonomous, interactive agents that have the abilities that have made the non-interactive characters of traditional media believable*". On the term believable, he mentions that it is a term taken from the arts to describe *characters that "work"*. He also adds three lessons from the arts about the fundamental nature of believability:

1. *Believable agents may not be intelligent. AI systems designed for rationality and intelligence would be inappropriate for building believable agents*.

2. *Believable agents may not be realistic. It is better to go with less realistic characters which meet the audience's expectations than to go with more realistic characters which don't*.

3. *Believable agents will have strong personalities. These personalities should affect everything about the agent, including how the agent moves, thinks, and talks. Also, idiosyncratic quirks are extremely important parts of the agent's personality*.

Regarding the type of agents we are interested in developing, we have taken the designation of *believable interactive characters* by Bates [6], and also stepped back from the idea of *virtual agents*, in order to make the concept's relation with the arts more explicit by calling them *Characters* instead of Agents. Because robotic characters represent the apex of animated characters, we also reinforce the idea of them being *Embodied*. Throughout our work, we therefore drop the designation of such entities as Agents, and adopt the use of the term *Characters* in general. As such, we define **Interactive Embodied Characters** as *any type of artificial embodied agent, virtual*

or robotic, that performs believable, interactive behaviour with humans in given tasks, regardless of human or anthropomorphic form, and of communication being verbal or non-verbal.

3.1.1 Architectures

The creation of interactive virtual characters has been explored for quite a time now. Some authors have established foundations on this topic. Here we especially cite the works by Bates, Reilly, Badler and Perlin & Goldberg's.

Bates has undoubtedly established the foundation of interactive virtual *characters* through his concept of *believable agents*. Along with Loyall and Reilly he described the Tok architecture, which addresses the capabilities of perception, reactivity, goal-directed behaviour, emotion, social behaviour, natural language analysis and natural language generation [34]. As such it presents as one of the earliest architectures for socially interactive virtual agents.

The Perlin & Goldberg presented Improv, a system that allows to create virtual actors that respond to users and to each other in real-time, focusing on the need of the authors that build such virtual actors [35]. The system is composed of an animation engine and a behaviour engine. The behaviour engine provides mechanisms to run scripts and rules that control the actors. The animation engine provides animation layers that allow to create and blend continuous motion using procedural techniques. Both works have built an architecture that connects a flexible animation system with some form of AI agent. The latter especially considers the fact that such a system will be mostly useful if it considers not only the run-time environment, but also the authoring process, by providing ways of establishing how the "mind" of the character connects and communicated with the "body". The performing character is actually part of a more complex artificial intelligence (AI) agent. While a fixed-storytelling character may be fully controlled by an animation that is pre-designed to follow and match such story specifically, interactive characters need animation systems that can adapt and change in response to the users' and environment's events in interaction-time.

Badler has presented Jack, another one of the earliest interactive virtual human (VH) systems [36]. The author strived to achieve a system that was able to create a VH that could exhibit both pre-design and procedural animation that was controlled in real-time by an AI agent that responded to a user via a language based interface.

Many interactive character systems have since then been following a three layer intention-behaviour-realization framework, which was formerly proposed by Kopp and colleagues as the SAIBA framework [37], illustrated in Figure 3.1. This framework was created especially for virtual humanoid characters. It splits the whole agent architecture into a first layer capable of performing some decision-making on the *Intention* of the agent, which is then made into a *Behaviour* plan by the middle layer, and finally, performed and rendered as the virtual character through the *Realization* layer.

Figure 3.1: The SAIBA framework as described in [37]

The SAIBA framework has also been used with robots (e.g. [38, 39, 40, 41]), and was later extended to also consider interactive applications and environment-awareness by myself and colleagues, calling it the Socially

Expressive Robotics Architecture (SERA) [42]. SERA is described further in Chapter 5.1.

Schröeder developed the SEMAINE API, which was used in the EU FP7 Semaine Project[1]. This is a component integration framework, based on the principles of asynchronous messaging middleware. Its architecture, however, has a pipeline message flow, meaning that it follows in the traditional sense-think-act loop of interactive agents. The author points out two key requirements for a framework of this kind: Infrastructure, meaning that components must be able to run on different programming languages and operating systems; and Communication, meaning that components must follow suitable representation formats, which should be standards where possible[43].

CMION was developed in the context of the EU FP7 LIREC[2]. It is a mind-body framework for integrating sensors and actuators through various degrees of abstraction. It was designed especially for allowing agent migration (transferring the agent's identify to a different embodiment). As such, it abstractly encapsulates functionalities of an embodiment into what they call competencies. These competencies share information through a blackboard component. By defining an embodiment as a set of competencies, agents can then migrate to other embodiments, as long as those implement the same type of competencies[44].

ROS - Robot Operating System is a popular middleware for robotics that provides a common communication layer to enable different types of sensors, motors and other components to exchange data [45, 46]. ROS is module-based, meaning that a ROS-based robot actually runs several different modules, being each one of them responsible for controlling one or more components of the robot. They communicate based on a message oriented middle-ware (MOM). This is accomplished through a publish-subscribe pattern, in which each module specifies the type of messages it wants to receive (subscription), so that each time another module produces that message (publication), the subscribed modules receive it.

3.1.2 Behaviour

Several authors have proposed different languages and schemes that allow to model and represent non-verbal behaviour in SIAs such as BEAT [47], CML[48], MPML [49] or APML [50]. The performance of such behaviour tends however to be based solely on the selection or blending of pre-designed animations specific to the used embodiment.

Badler [51] has presented an Expressive MOTion Engine (EMOTE) that implements LMA [52] using high-level parameters for human animation control; however, this solution is designed for anthropomorphic characters.

Schröder et al. have proposed EARL, a general mark-up language that is not dependent of any emotion model or theory, thus marking a possible step for an abstract and broad specification of emotive behaviour [53]. However, their language provides only a structure to gather the description of an emotional expression, with no means on how to accomplish it.

Within the SAIBA framework [37], presented in the previous subsection, we also find the Behavior Markup Lalguage (BML), a markup language used to represent synchronized multi-modal behaviour, that can be somewhat seen as a successor to MURML [54] The purpose of this language is to provide a specification of basic expressive channels and modalities that different authors can use in order to specify behaviour in a generic way. The modalities currently defined in the BML 1.0 Standard are: Face, Gaze, Gesture, Head, Locomotion, Posture and Speech. More

[1] `http://www.semaine-project.eu/` (accessed January 12, 2019)
[2] `http://lirec.eu/` (accessed January 12, 2019)

detail about the BML actions can be found in [55].

EMBR is a realtime animation engine for interactive embodied agents, designed to work with BML [56]. It proposes that between a high-level behaviour description layer such as one provided by BML, and SAIBA's realizer/rendering layer (e.g. a 3D engine) one must include an animation layer that can be scripted. For that purpose they propose the EMBRScript which can be used to realize particular BML actions such as gazing at another character with a given emotional expression, therefore allowing such action realization to become more procedural. However, because it is designed as a layer to be fit into the BML architecture, it remains heavily directed at the animation of virtual human characters.

3.1.3 Animation and Control

Tomlinson has provided a description about how animating characters for interactive applications is different then animation characters for film and video [57]. He therefore distinguishes two types of animation: Linear Animation for film and video, and Interactive Animation otherwise.

Smartbody is a procedural animation system developed especially for virtual humans [58]. It takes a BML specification of behavior as input in order to be controlled by any type of AI agent. This behavior is scheduled and executed in several motion controllers, which are combined in each frame to generate a set of skeletal joint rotations and translations. Smartbody can procedurally generate and adapt gestures using an example-based motion synthesis approach. The system is heavily based on [59], an example-based motion synthesis technique for locomotion, reach and object manipulation. This technique takes a takes a large set of example postures of a given embodiment e.g. reaching towards different directions, and is then able to produce a grasping pose for ano direction by blending the previously authored examples.

Levine and colleagues have recently developed a technique to animate characteres based on motion learning [60]. Artists first train the system by creating example motions that are associated with task specification. This trained probabilistic model can later be used to generate new motions that accomplish new tasks.

Moussa et al. have embarked on one approach for this, using MPEG-4 Facial Animation Parameters (FAP) [61] applied to a humanoid robotic face [62]. However, FAPs are designed for human faces and thus comprehend an extensive number of expressive features for the human face.

On the other hand, movement can also be dissociated of the actual body, and instead related to its meaning, as used in acting and other performance arts such as the Delsarte system [63] and the Laban Movement Analysis [64, 65].

3.2 Inverse Kinematics

In general, the computation for the animation of a hierarchical, articulated structure (kinematic chain) is done through Forward Kinematics (FK) and Inverse Kinematics (IK). This section briefly introduces some fundamental concepts and techniques regarding these processes. IK is a very extensive field and we will therefore focus on the aspects of it that are most related with our work.

We start by introducing the lexicon and fundamental concepts used in this paper, regarding both FK and IK. Figure 3.2 provides visual guidance on each of the elements that compose a kinematic chain, throughout the

following description:

- given an articulated structure of N joints (J_i) that connect N segments,
- being the first joints called $Root$ and the tip of the last segment called $EndEffector$,
- having P_i as the world-space position coordinates of each joint L_i,
- having the P_1 (or P_{Root}) located at the origin (O) of the world-space,
- with each joint allowing for a rotation α about an arbitrary axis R_i with angular limits such that $min\alpha_i \leq \alpha \leq max\alpha_i$,
- being a *Kinematic Solution (KS)* given by the configuration of angles $\alpha_1, ..., \alpha_N$ that are applied to each rotation axis $R_1, ..., R_N$, of each joint $L_1, ..., L_N$,
- being a *Posture* represented by the configuration of world-space positions $P_1, ..., P_N$ of each joint $L_1, ..., L_N$,

Forward Kinematics allows to compute the final *Posture* achieved from a given *Kinematic Solution*, while Inverse Kinematics allows to compute the *Kinematic Solution* that allows to achieve a given *Posture*. In reality, IK is generally used to compute the KS that allows solely the end-effector S to achieve a given target T . The transform of an end-effector $S = S_{pos}S_{ori}$ that moves in 3D space may contain up to six DoFs: three for a position in world-space, and three for an orientation in world-space. Therefore most IK techniques created to date allow to calculate the KS that allows the chain's end-effector to achieve either a given position S_{pos}, or a given orientation S_{ori}, or both.

The IK problem is generally addressed either through an analytical solution, or through a numerical solution. The main difference between both is that an analytical solution is a closed-form expression that takes as input the desired posture, and outputs the (set of) kinematic solutions for it, solving the IK problem for a particular kinematic chain. This means that if any change is made to any joint configuration regarding its rotation axis, angular limit, or even a segment's length, then the solution-expression needs to be re-calculated. On the other hand, a numerical solution can be more generalizable, but is generally implemented as a non-linear programming problem in which typically the algorithm iterates towards an approximate solution (modelled as an optimization problem). Analytical solution-expressions are therefore faster for computing, than numerical techniques, but take a lot of effort to build and are embodiment-specific [66]. They are especially appropriate for specific, well-defined limbs such as the human arm subsystem. Numerical solutions can provide flexibility to better adapt to different types of kinematic chains. In our work we try to address the IK problem in a general way, so that it can be used with any embodiment

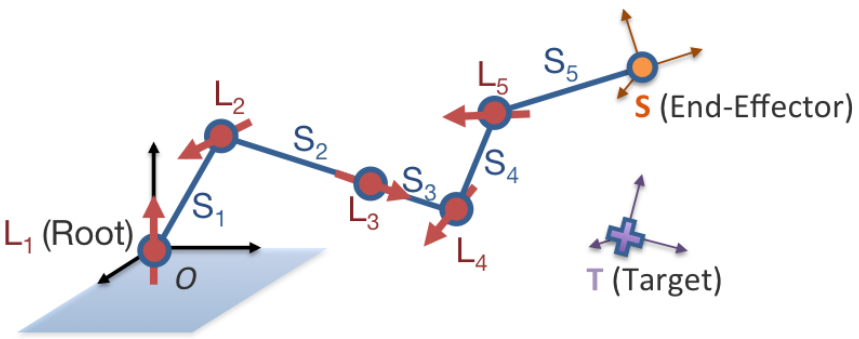


Figure 3.2: An articulated structure (kinematic chain) as used in both Forward Kinematics (FK) and Inverse Kinematics (IK). Also shown is a given target T that is to be reached by the end-effector S.

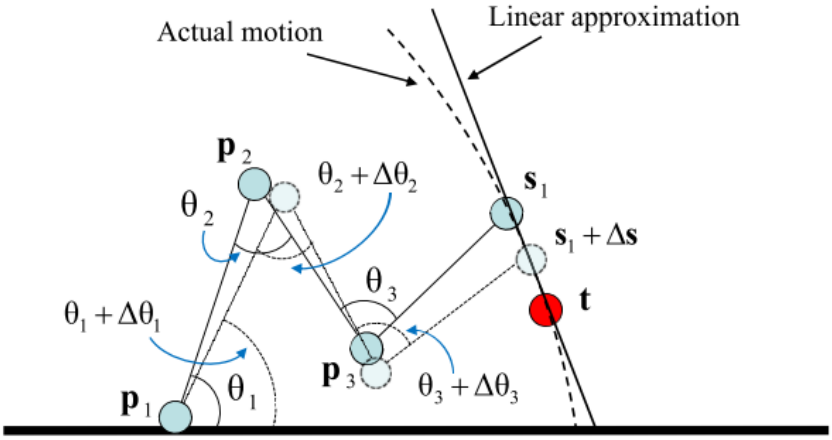


Figure 3.3: The Jacobian solution as a linear approximation of the actual motion of the kinematic chain. Description and image cited verbatim from [67].

and with any morphology, with minimum (or in absence of) parameter customization. Therefore we follow with a brief description of some of the most popular numerical techniques.

A comprehensive summary of the most popular IK techniques has already been gathered by Aristidou et al. [67] and is recommended to the interested reader. Given that the latter one is recent and already describes nearly every option of inverse kinematics up to date, we will refrain from extending this literature section beyond the bare minimum. As such we describe here only the techniques that are central to our contribution, while solely providing a mention to various other relevant techniques such as [68, 69, 70, 71].

3.2.1 Jacobian Inverse Methods for IK

While in general, the IK problem is highly non-linear, the Jacobian methods provide linear approximations to it. They are based on the computation and inversion of the Jacobian matrix which contains the partial derivatives of the entire chain system, relative to the end-effectors.

An extensive explanation of these methods is provided by Buss [72] and should be consulted for more details. The problem is illustrated by Figure 3.3. In simple terms, given the current position and/or orientation $\vec{s}$ (i.e. *transform*) of an end-effector, and a target position and/or orientation $\vec{t}$ that it should achieve, let $\vec{e} = \vec{t} - \vec{s}$ represent the error vector (or *task*) between the end-effector and the desired target values, and $\theta = (\theta_1, ..., \theta_n)^T$, the current joint angles of the system, having n as the number of joints. The value m will be the dimension of $\vec{e}$ (and consequently, of both $\vec{s}$ and $\vec{t}$) and will depend on the target IK task. If the task is e.g. the 3D position constraint or 3D orientation constraint of a single end-effector then $m = 3$. If it is to control both the 3D position and 3D orientation, then $m = 6$. However one might choose a task that controls the orientation of only two of the rotation axes, in which case $m = 2$. Alternatively, one might also require to set one end-point to a given XY position, regardless of its position in the Z axis; in that case m would also be 2.

Note that for position control, the task is directly calculated as $\vec{e} = \vec{t} - \vec{s}$, while for orientation control many parametrizations exist. When using Euler angles, one option is to calculate it the same way, i.e., solving at the differential level, by using the desired angular velocity vector. When using quaternions, we may take the vector part of the target quaternion orientation $q_{\vec{v}}$, and use this vector as the task.

$\psi_{i,j} =$		Joint j is	
		Prismatic	Revolute
Task i is	Translation	$\vec{\text{rot}}_z^{\,i}$	$\vec{\text{rot}}_z^{\,i} \times (\vec{\text{pos}}^{\,i} - \vec{\text{pos}}^{\,i-1})$
	Rotation or Posture	0	$\vec{\text{rot}}_z^{\,i}$

Table 3.1: Calculation of the Jacobian terms $\psi_{i,j}$.

The Jacobian matrix J of size $m \times n$ (rows$\times$ columns) is a function of the current θ values defined by

$$J(\theta) = \left(\frac{\partial s_i}{\partial \theta_j} \right)_{i,j} \tag{3.1}$$

It will result in a matrix such as

$$J = \begin{bmatrix} \psi_{1,1} & \psi_{1,2} & \cdots & \psi_{1,n} \\ \psi_{2,1} & \psi_{2,2} & \cdots & \psi_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{m,1} & \psi_{m,2} & \cdots & \psi_{m,n} \end{bmatrix}$$

where each column represents the influence of joint j over each task i. A simple rule for calculating each element $\psi_{i,j}$ is presented in Table 3.1.

Let T_i be the transform matrix for the frame of joint i:

$$\mathbf{T_i} = \begin{array}{c} \begin{array}{cccc} \vec{\text{rot}}_x^{\,i} & \vec{\text{rot}}_y^{\,i} & \vec{\text{rot}}_z^{\,i} & \vec{\text{pos}}^{\,i} \end{array} \\ \left(\begin{array}{ccc|c} \begin{bmatrix} r_{x_1} \\ r_{x_2} \\ r_{x_3} \end{bmatrix} & \begin{bmatrix} r_{y_1} \\ r_{y_2} \\ r_{y_3} \end{bmatrix} & \begin{bmatrix} r_{z_1} \\ r_{z_2} \\ r_{z_1} \end{bmatrix} & \begin{bmatrix} x \\ y \\ z \end{bmatrix} \\ \hline 0 & 0 & 0 & 1 \end{array} \right) \end{array}$$

The term $\vec{\text{pos}}^{\,0,i}$ is the translation between the root frame and joint i's frame, while $\vec{\text{rot}}_z^{\,0,i}$ is the z-vector of the rotation between the root frame and joint i's frame. Assuming that the root frame is located at $[0, 0, 0]$ and that its rotation is equal to I_3 (i.e., $T_{\text{root}} = I_4$), we can take the values of both $\vec{\text{rot}}_z^{\,0,i}$ and $\vec{\text{pos}}_i^{\,0,i}$ directly from matrix T_i. If that is not the case, then either T_i must be transformed by T_{root}^{-1}, or both vectors $\vec{\text{rot}}$ and $\vec{\text{pos}}$ must be transformed by that inverse.

Please refer to [72] or [73] for more information on how to calculate the Jacobian matrix, or alternatively to [74] for a fully detailed description. This matrix allows to approximate the change in the end-effector's transform given

an increment in the system's joint angles of $\Delta\theta$:

$$\Delta\vec{s} \approx J\Delta\theta \tag{3.2}$$

The problem will be solved by seeking a value for $\Delta\theta$ such that $\Delta\vec{s}$ becomes approximately equal to $\vec{e}$, by making:

$$\vec{e} = J\Delta\theta \tag{3.3}$$

This equals the question *how much must I increment each joint angle θ in order for the end-effector to move by the amount $\vec{e}$?*

A solution to the IK problem is therefore given by equation (3.2) for $\Delta\theta$, using the inverse of the Jacobian:

$$\Delta\theta = J^{-1}\vec{e}$$

The implementation of any variation of the Jacobian methods typically follow a similar approach, which is as an optimization problem that minimizes the residual error $e^{\text{total}} = \|\vec{e}\|$.

In most cases however, this equation cannot be solved uniquely, as Jacobian J may be non-square, non-invertible, or nearly singular (which would provide poor and unstable results). Several alternatives have been found to calculate the Jacobian's inverse. One of them is to use the Jacobian's transpose J^T instead of its inverse, and multiplying it by an appropriate scalar α (Equation 3.4).

$$\Delta\theta = \alpha J^T\vec{e} \tag{3.4}$$

Another possibility is to use its pseudoinverse $J^\dagger$ (also called the *Moore-Penrose inverse of J*) as shown in Equation 3.5.

$$\Delta\theta = J^\dagger\vec{e} \tag{3.5}$$

Using the pseudoinverse method also allows to perform a projection into the nullspace of the Jacobian, meaning that we may further optimize the solution towards a secondary task as shown in Equation 3.6. An example of that would be to use the end-effector's orientation as the main task $\vec{e}$, for which the solved $\Delta\theta$ would minimize the error $J\Delta\theta - \vec{e}$, while choosing a $\vec{z}$ vector of the same dimension as θ, that would attempt to keep the resulting angles as close as possible to zero (secondary task), without disrupting the main task.

$$\begin{aligned} P_{N(J)} &= I - J^\dagger J \\ \Delta\theta &= J^\dagger\vec{e} + P_{N(J)}\vec{z} \end{aligned} \tag{3.6}$$

The $\vec{z}$ vector can be calculated by minimizing a criterion $h(\theta)$, using $\vec{z} = \xi\nabla h(\theta)$, where ξ is a gain factor. Baerlocher shows an example of the typical application of keeping the joint angles as close as possible to some desired values (e.g. to zero) [73], by using $h(\theta, \theta_{desired}) = \|\theta - \theta_{desired}\|^2$. The example of keeping the joint angles close to zero would therefore be to have just $h(\theta) = \|\theta\|^2$. Alternatively, if the secondary task e_2 is clearly represented as a Jacobian matrix J_2, then we might also use Equation 3.7, as explained by [73].

$$z = (J_2 P_{N(J_1)})^\dagger(\vec{e_2} - J_2 J_1^\dagger\vec{e}) \tag{3.7}$$

Both the transpose and the pseudoinverse methods however, suffer from either approximation errors, or from instability near singularities. Such methods also suffer from poor results when the target is too distant from the current position or orientations. One method to mitigate that problem is also presented by Buss [72], and consists in clamping the $\vec{e}$ vector so that its norm is never greater than a constant value D_{max}, as shown in Equation 3.8.

$$\vec{e} = \texttt{ClampMag}(\vec{t} - \vec{s}, D_{max}) \tag{3.8}$$

$$\text{ClampMag}(w, d) = \begin{cases} w & \text{if } \|w\| \leq d \\ d\frac{w}{\|w\|} & otherwise \end{cases}$$

The damped least squares method (DLS), also called the Levenberg-Marquardt method further attempts to address these issues, by including a non-zero damping constant. This constant however, must be chosen carefully depending on the kinematic configuration of the system and on its purpose, in order to remain numerically stable near singularities, without keeping the convergence rate too slow. Equation 3.9 shows how to calculate $\Delta\theta$ using the DLS method, where λ is the damping constant, which must be carefully selected based on the details of the multibody and expected target positions, in order to ensure stability. A larger damping value allows the solutions to become more stable near singularities, however if the constant is too large then the convergence rate will be lower (as it will require more iterations).

$$\begin{aligned} J^{\dagger^\lambda} &= J^T(JJ^T + \lambda^2 I)^{-1} \\ \Delta\theta &= J^{\dagger^\lambda}\vec{e} \end{aligned} \tag{3.9}$$

Alternatively, the DLS method may also be implemented through the Singular Value Decomposition method (SVD), which decomposes a matrix J of $m \times n$ into three matrices U $(m \times m)$, D $(m \times n)$ and V $(n \times n)$, such that $J = UDV^T$. D is the singular value matrix of J, with its only non-zero values being along its diagonal $d_{i,i} = \sigma_i$, being σ_i the i^{th} singular value of J. Also, because σ_i may be zero, let r be the largest value such that $\sigma_r \neq 0$, with σ being sorted such that $\sigma_i \geq \sigma_{i+1}$. Based on the SVD of J and following the elaboration by [72], the DLS method can also be expressed as in Equation 3.10:

$$\begin{aligned} J^{\dagger^\lambda} &= (\sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda^2})\mathbf{v}_i\mathbf{u}_i^T \\ \Delta\theta &= J^{\dagger^\lambda}\vec{e} \end{aligned} \tag{3.10}$$

As mentioned before, the major issue with the DLS technique is the selection of an appropriate damping factor. Buss and Kim [75] address this issues with the Selectively Damped Least Squares (SDLS) method that adjusts the damping factor for each singular vector of the Jacobian's singular value decomposition (SVD). This method converges faster than DLS and does not require ad hoc damping constants. First a global $\gamma_{\max}$ is chosen, for which they recommend a typical value to be $\pi/4$ (45 degrees). This will be the maximum permissible change in any joint angle in a single iteration. Then we take the SVD of $J = UDV^T$ and express the desired change in end-effector position as $\vec{e} = \sum_i \alpha_i u_i$ where u_i is the i^{th} column of U and $\alpha_i = \langle \vec{e}, u_i \rangle = u_i^T\vec{e}$. Let also $\rho_{\ell,j} = \|\partial s_\ell / \partial \theta_j\|$ be the relative magnitude of the change of the $\ell\, th$ task variable in response to a small change in the jth joint angle

(from Equation 3.1). We further define the auxiliary N and M vectors along with the selective damping factor γ:

$$\begin{aligned} N_i &= \sum_{j=1}^{m} \|u_{j,i}\|, \forall i \in [1,n] \\ M'_{i,\ell} &= \sigma_i^{-1} \sum_{j=1}^{n} |v_{j,i}| \rho_{\ell,j}, \forall i \in [1,m], \forall \ell \in [1,m] \\ M_i &= \sigma_{\ell=1}^{m} Mi, \ell, \forall i \in [1,n] \\ \gamma_i &= \min(1, \frac{N_i}{M_i}) \cdot \gamma_{\max} \end{aligned} \tag{3.11}$$

Finally, the SDLS solution is expressed as $\Delta\theta$:

$$\begin{aligned} \varphi_i &= \texttt{ClampMaxAbs}(\sigma_i^{-1} \alpha_i v_i, \gamma_i) \\ \Delta\theta &= \texttt{ClampMaxAbs}(\sum_{i=1}^{r} \varphi_i, \gamma_{\max}) \end{aligned} \tag{3.12}$$

Baerlocher introduced techniques that allow to solve the IK problem for multiple tasks with priorities, i.e., by specifying the priority in which each task should be achieved [73]. In particular he aimed at solving the problem of postural control for virtual humans, by allowing to specify e.g. a task for one hand to reach a certain goal position, plus another task for the head to face a certain direction, while keeping the whole body balanced. His technique is actually a rewritten version of the solution initially proposed by Maciejewski [76], upon also being modified to account for algorithmic singularities. We found his approach to be the most significant one to compare to given our goals. Equation 3.13 presents Baerlocher's formulation of the DLS applied to two tasks $\vec{e_1}$ and $\vec{e_2}$, whose corresponding Jacobian matrices are J_i and damping constants $\lambda_i,\ i \in [1,2]$, with the first task having a higher priority than the second.

$$\Delta\theta = J_1^{\dagger^{\lambda_1}} \vec{e_1} + (J_2 P_{N(J_1)})^{\dagger^{\lambda_2}} (\vec{e_2} - J_2 J_1^{\dagger^{\lambda_1}} \vec{e_1}) \tag{3.13}$$

He finally elaborates towards a formulation that supports more than two levels of priority, by following the same approach. In that case, given a set of tasks $[\vec{e_1}, \vec{e_2}, ..., \vec{e_p}]$, for which J_i and $\lambda_i,\ i \in [1,p]$ are the corresponding Jacobian and damping constants, with $i = 1$ corresponding to the highest priority, and $i = p$ to the lowest, Equation 3.14 presents the general formulation for the multiple-task-priority method:

$$\begin{aligned} \Delta\theta_i &= \Delta\theta_{i-1} + (J_i P_{N(J_{i-1}^A)})^{\dagger^{\lambda_i}} (\vec{e_i} - J_i \Delta\theta_{i-1}) \\ \Delta\theta_1 &= J_1^{\dagger^{\lambda_1}} \vec{e_1} \\ J_i^A &= \begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_i \end{bmatrix} \end{aligned} \tag{3.14}$$

The major difference between his problem statement and ours is that his problem is especially directed at virtual humans (VH) with many DoFs while ours is directed at robots with much fewer DoFs than the VH, therefore his

Algorithm 1: Pseudocode for a typical Jacobian method's iterative solver.

input : $\theta, \vec{t}$; // initial joint angles, // target task variables (position and/or orientation)

$\theta' \leftarrow (\theta_1, ..., \theta_N)^T$
$\dot{\theta} \leftarrow \vec{0}$
$\text{best}_{\dot{\theta}} \leftarrow \dot{\theta}$
$\text{best}_{\text{error}} \leftarrow$ MAX_FLOAT
$\vec{s} \leftarrow \text{ForwardKinematics}(\theta')$; // calculate EE position and/or orientation from θ'
for $N \leftarrow 1$ **to** *MAX_ITERATIONS* **do**
  $\vec{e} \leftarrow \text{ClampMag}(\vec{t} - \vec{s}, D_{max})$; // where $\vec{t}$ is the target position and/or orientation
  if $\|\vec{e}\| \leq best_{error}$ **then**
    $\text{best}_{\text{error}} \leftarrow \|\vec{e}\|$
    $\text{best}_{\dot{\theta}} \leftarrow \dot{\theta}$
  if $\|\vec{e}\| \leq$ *ERROR_TOLERANCE* **then**
    break
  $J \leftarrow \text{Jacobian}(\theta')$// calculate Jacobian of θ'
  $J^{-1} \leftarrow \text{CalculateInverse}(J)$// using one of the possible methods
  $\dot{\theta} \leftarrow J^{-1} \cdot \vec{e}$
  $\theta' \leftarrow \theta' + \dot{\theta}$
  $\vec{s} \leftarrow \text{ForwardKinematics}(\theta')$; // calculate EE position and/or orientation from θ'
end
return $\theta + best_{\dot{\theta}}$

problem is more under-constrained (or redundant) than ours. One of the consequences of that is that the null-space projection operator in his situation will allow for the secondary task to perform much better than in our case.

Finally, within his techniques, Baerlocher also suggests the use of Maciejewski's method for computing an appropriate damping factor based on the minimum singular value of the Jacobian [77]. Let b_{max} be a bound on the norm of the solution such that $\|J^{\dagger^{\lambda}} \Delta x\| \leq b_{\text{max}}$, then Maciejewski's damping factor can be calculated through Equation 3.15.

$$\lambda = \begin{cases} \frac{d}{2} & \text{if } \sigma_{\text{min}} \leq \frac{d}{2} \\ \sqrt{\sigma_{\text{min}}(d - \sigma_{\text{min}})} & \text{if } \frac{d}{2} \leq \sigma_{\text{min}} \leq d \\ 0 & \text{if } \sigma_{\text{min}} \geq d \end{cases} \qquad d = \frac{\|\vec{e}\|}{b_{\text{max}}} \tag{3.15}$$

Conclusions drawn from the comparison of several Jacobian techniques (e.g., Jacobian Transpose, Damped Least Squares (DLS), Selectively Damped Least Squares (SDLS)), both by Buss [72] and by Aristidou [67] are that the Jacobian methods are mostly appropriate for single end-effector situations, not always suitable for time-critical situations (e.g. real-time computation) and the incorporation of constraints using this family of methods is neither straightforward nor controllable towards an optimal solution. Furthermore, while the SDLS seems to be the most promising method, it is not clear how to use it along with a secondary task.

To conclude this section we share the base pseudocode for such methods in Algorithm 1.

3.2.2 Data-driven, Probabilistic and Hybrid Approaches for IK

Regarding expressive posture control, Neff & Fiume have presented the Body Shape Solver [78] which addresses the problems of pose modelling, balance, and world-space and body-space constraints into a single integrated inverse kinematics solver for humanoid skeletons. The technique can be used by animators to solve for character poses either based on a given set of parameters, or by selecting a shape set. However their algorithm is specific to the human body, as it is a hybrid technique that uses both analytical and optimization methods.

Grochow et al. propose an IK system that is trained through a set of human poses [79]. The poses selected will therefore define the style of the resulting motion. By training with different poses, one can drive the solver to produce different styles of animation. A key feature is that it can both extrapolate a new pose from a style training set, while also allowing to interpolate between different styles. However, despite addressing the problem of style and expressivity of IK, the system was especially developed for motion capture, and requires off-line training, which confines the results to be highly dependant on the quality of the training data.

Courty and Arnaud propose the Sequential Monte-Carlo IK (SMCIK) solver, that models the problem using a probabilistic point of view, using a Monte Carlo approach [80]. The SMCIK solver is formulated as a filter whose state is the entire complex articulated figure. An interesting aspect of the algorithm is that it produces a complete motion, from initial position, to the target position, as a result of the optimization process. This solution however does not offer proper control over the quality of the resulting animation, and does not guarantee that a solution is found. Also, due to the random nature of algorithm, each execution will produce a different solution. In the best cases, what it guarantees is the achievement of a solution, but not how consistent or reliable it is.

The Particle IK Solver, featured in the video-game *Spore*, was developed to allow characters with various custom morphologies to walk naturally and to perform actions in their surrounding environment such as looking towards a direction, or grasping an object [81]. It allows such creatures to behave coherently by performing locomotion and animated actions in a procedurally generated world.. They preserved a traditional animation workflow so that artists could take a central role in the development process. The main concern was to keep animations looking as natural as possible, and allowing artists to design them in a generalizable way, so that their stylistic details could remain across characters. The *Spore* engine is aimed specifically at the types of creatures used in the game, which contain leg groups and arm groups, and perform a set of pre-determined actions. Hecker mentions that they failed to attain naturally controlled poses using the common iterative non-linear solvers such as CCD, Jacobian methods or Constrained dynamics. The Particle IK Solver was therefore developed to allow characters with various custom morphologies to walk naturally and to perform actions in their surrounding environment such as looking towards a direction, or grasping an object. Particle IK can solve for various goals by using embodiments that result in an *underdetermined system*, i.e., ones that will result in more DoFs than IK goals. Therefore the remaining DoFs can be used to achieve secondary objectives. The solver runs in two phases. First it solves for the spine of the character and then for the limb poses, while treating the spine as fixed. Their argument was that a single-phase solver based on existing techniques did not allow them to make specific ad hoc tuning adjustments or treat special cases, without compromising the quality of the solution in other areas of the pose. By elaborating a new solver, they managed to achieve *local control* over the solution, which was not possible using conventional IK techniques. As a down side, the system proposed by Hecker is heavily directed at the type of creatures used in Spore, and uses techniques that assume the existence of 3-DoF joints, which we do not consider in robotics.

3.2.3 Heuristic IK Techniques

This sub section presents IK techniques that are solved as an iterative search. Cyclic Coordinate Descent (CCD) is a popular IK technique, both in computer graphics animation, robotics, and even in protein science [82, 67]. Some of its main advantages are that it is very easy to implement, fast to compute, and has linear-time complexity regarding the number of DoFs. In each iteration it starts from the end-effector, and moves inwards towards the base, adjusting each joint angle at a time, in order to minimize the distance between the end-effector and the target position. This procedure is repeated until either the error is considered to be minimal, or until a maximum number of iterations has been ran. Despite its simplicity and efficiency, the enforcing of constraints remains as a difficult problem. Constraints are applied locally, and it does not provide an intuitive way to enforce them globally. Figure 3.4 shows the execution of CCD.

Running CCD with constrains also implies in many cases that throughout the iterative process, the solution might not always improve, thus requiring an adequate heuristic to select the best solution from within all the ones that were computed. It was initially designed for a single end-effector, although a multiple-chain method has been described in [83] by dividing the structure into smaller sub-chains, and solving them each independently.

The major problems pointed out to CCD however, are the production of unrealistic and non-continuous motion across subsequent solutions, and the overemphasising of the movement of the joints near the end-effector which leads to unnatural motion.

Johnson has proposed an Expressive IK solution that also uses expert body knowledge (example poses given by animators) to augment the quality of the results given by a CCD algorithm [84]. The examples are both used to estimate joint constraints, and also to perform multi-target pose blending which would then be used as an initial solution before the IK algorithm is ran (this step was not developed, however).

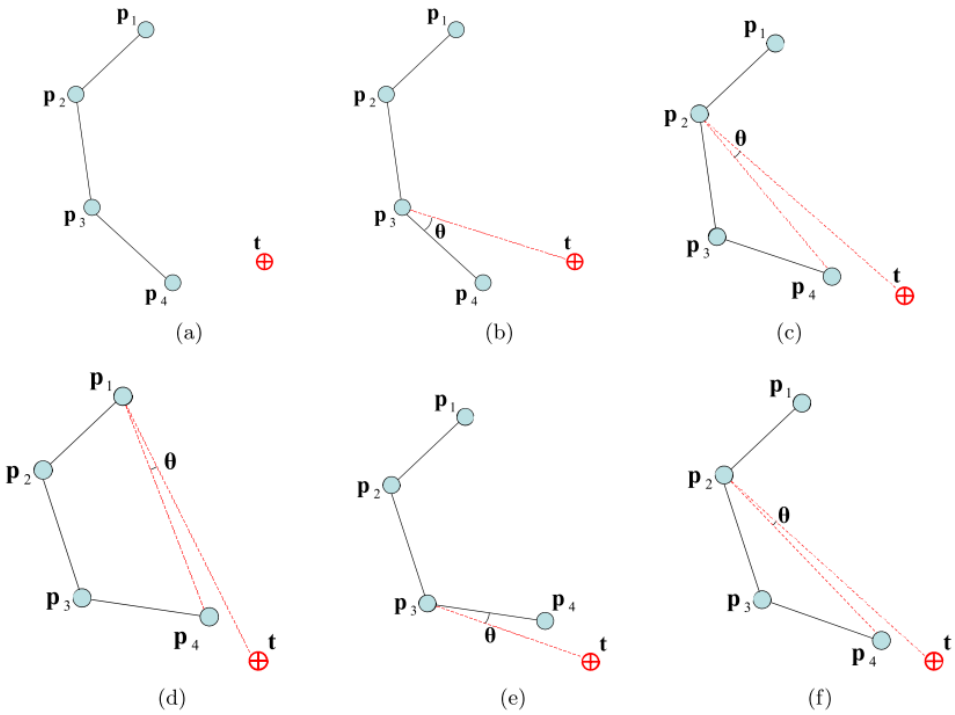


Figure 3.4: An example of a visual solution of the IK problem using the CCD algorithm. (a) The initial position of the manipulator and the target, (b) find the angle θ between the end effector, joint p3 and the target and rotate the joint p4 by this angle, (c) find the angle θ between the end effector, joint p2 and the target and rotate joints p4 and p3 by this angle, (d), (e) and (f) repeat the whole process for as many iterations as needed. Stop when the end effector reaches the target or gets sufficiently close. Description and image cited verbatim from [85].

The algorithm, QuCCD, is a Quaternion-based version of the popular CCD algorithm. QuCCD includes a fast joint-limit constraint approach similar to [86], that takes on a geometrical approach instead of clamping angles as usual (which would require converting the quaternion to Euler angles, clamp, and then back to a quaternion).

Some of Johnson's proposed techniques were used to animated Anemone, an expressive IK robot [87]. This robot used a hybrid between pose-blending, for the DoFs near its base, and QuCCD, to animate the upper half, so that it could both maintain an expressive posture, while still facing its "head" towards things in its environment. The whole computation was performed through quaternions, holding off the conversion until "just-in-time", before converting and sending the actual Euler angles to the motors. Despite presenting promising results for 3D animated characters, the author does end up announcing that "this method tends to produce very slow convergence for 1 DOF joints which are constantly bumping into a boundary".

FABRIK is an iterative method that takes on a geometric approach to the IK problem [88, 89]. It was inspired by the knot-tying problem [90] and borrows the idea of iterating through each joint individually as in CCD, but instead works in the joint-position space (instead of angles), and each iteration includes a forward step (traversing from the end-point to the base) followed by a backward step (that traverses from the base back to the end-point).

The adjustment of each joint is treated as a a problem of finding a point in a line. Figure 3.5 illustrates the execution of the algorithm, as further described.

We must first establish that $d_i = |P_{i+1} - P_i|$, for $i = 1, ..., N$, is the length of each segment i. FABRIK starts by moving the end-effector P_N to the target position t, becoming P'_N. This is an operation that can only be performed

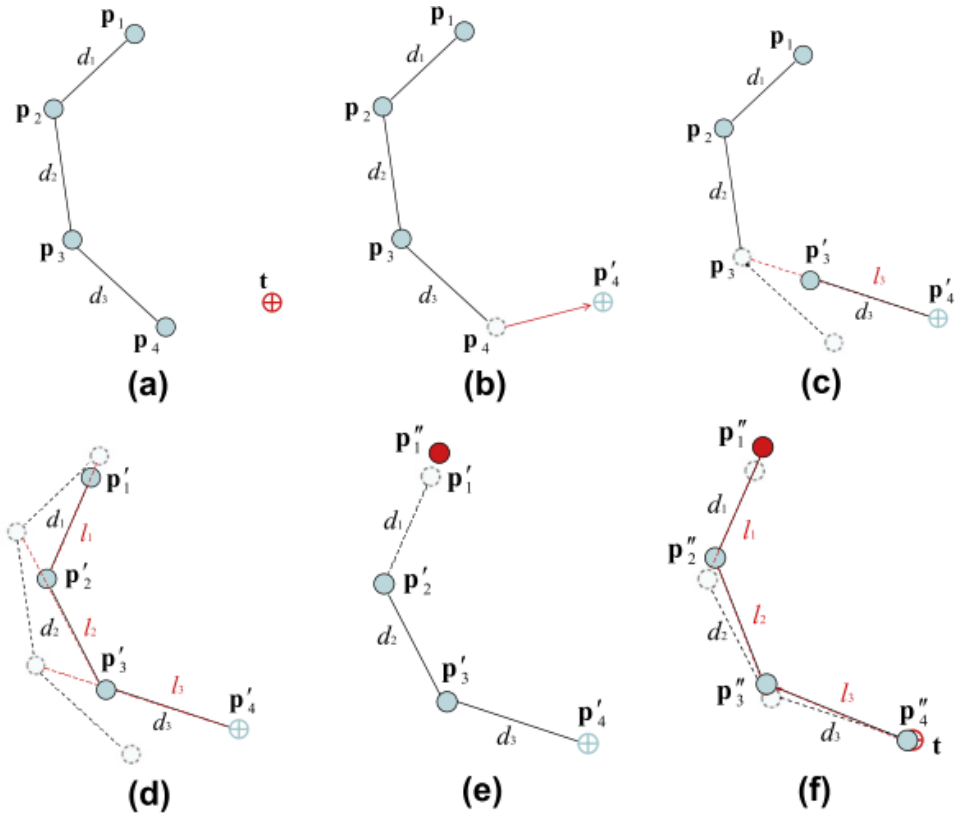


Figure 3.5: An example of a full iteration of FABRIK for the case of a single target and 4 manipulator joints. (a) The initial position of the manipulator and the target, (b) move the end effector p4 to the target, (c) find the joint p03 which lies on the line l3 that passes through the points p04 and p3, and has distance d3 from the joint p04, (d) continue the algorithm for the rest of the joints, (e) the second stage of the algorithm: move the root joint p01 to its initial position, (f) repeat the same procedure but this time start from the base and move outwards to the end effector. The algorithm is repeated until the position of the end effector reaches the target or gets sufficiently close. Description and image cited verbatim from [88].

in virtual space, as it intentionally breaks the kinematic configuration of the system by stretching the last segment. However, after this initial move, each successive link P_i is moved to a new position, towards P'_{i+1}, following the joint position update rule:

$$\begin{aligned} P_i &= (1-\lambda)P_{i+1} + \lambda P_i, \\ \lambda &= \frac{d_i}{|P_{i+1} - P_i|} \end{aligned} \tag{3.16}$$

After the forward phase, the Root joint will most likely end up in a position that is not the Origin of the space as it was initially. This happens because each joint, starting at the end-point, was pulled or pushed, in Cartesian space. In order to bring the kinematic chain back to the Origin, the backward phase starts by moving the Root P_1 so that $P_1 = O$. Just as in the first step of the forward phase, this operation also stretches (or shrinks) the first link to an invalid length. So again, but now in inverse order, each joint is traversed and moved to reset the segments to their initial length, while keeping the Root centred at the Origin, and having successfully pulled the end-point closer to

the target position. The backward joint position update rule is similar to the forward one:

$$\begin{aligned} P_i &= (1-\lambda)P_i + \lambda P_{i+1}, \\ \lambda &= \frac{d_i}{|P_{i+1} - P_i|} \end{aligned} \tag{3.17}$$

After the forward phase, the Root joint will most likely end up in a position that is not the Origin of the space as it was initially.

This happens because each joint, starting at the end-point, was pulled or pushed, in Cartesian space. In order to bring the kinematic chain back to the Origin, the backward phase starts by moving the Root L_1 so that $P_1 = O$. Just as in the first step of the forward phase, this operation also stretches (or shrinks) the first link to an invalid length. So again, but now in inverse order, each joint is traversed and moved to reset the segments to their initial length, while keeping the Root centred at the Origin, and having successfully pulled the end-point closer to the target position: The backward joint position update rule is similar to the forward one:

$$\begin{aligned} P_i &= (1-\lambda)P_i + \lambda P_{i+1}, \\ \lambda &= \frac{d_i}{|P_{i+1} - P_i|} \end{aligned} \tag{3.18}$$

This technique was created for, and works in virtual space, as it intentionally breaks the kinematic configuration of the system by stretching each segment during the Forward phase, which most likely ends up bringing the base joint to a position that is not the origin of the space as it was initially. However the Backward phase solves this, while bringing the whole kinematic solution closer to a solution. By working directly in the joint-position space, FABRIK avoids calculation of angles, which is one of its main advantages, making it even faster to compute than CCD. Other of its main features are that it does not suffer from singularity problems, produces naturally smooth and continuous motion, and emphasises movement in the joints closer to the base. Following an approach similar to [83], it also supports multiple end-effectors, and as such, full-body IK solving. Regarding the application of constraints, the authors present successful results in a system where each link is modelled as a generic 3-DoF, by decomposing the induced quaternions into swing and twist components, and enforcing limits on them separately following on the method described in [86].

Starke presents the Hybrid Genetic Swarm Algorithm (HGSA) for IK, using a biologically-inspired optimization technique as a universal IK solution for arbitrary joint chains [91]. The technique merges both the concept of Genetic Algorithms (GA), and of Particle Swarm Optimization (PSO). GA typically provides high-quality solutions to optimization and search problems, driven by theories of natural evolution. PSO is an optimization technique inspired by the behaviour of bird flocks and schools of fish, following the idea that complex behaviour can emerge from a collectivity of simpler organisms. HGSA is reported to achieve a sucess rate of nearly 100% within 10-60ms. However it does not support multiple end-effector or self-collision avoidance.

3.3 Expressive and Animated Social Robots

This section starts by presenting existing theories and design principles that have been proposed regarding specifically the design of robotic expression. These works are intended as broad, or general theories, without being tied to a

specific embodiment or application. It follows with a list of various robotic embodiments, along with example works that have explored their expressive capabilities.

Regarding the design of believable social robots, Dautenhahn [92] divided the design process in two dimensions: the Universal dimension, in which the universal features of a behaviour or expression are abstracted; and the Abstract dimension, in which the designer of the behaviour or expression is free to be creative and develop a more artistically based result.

Meerbeek et al. also follow the Universal vs. Abstract dimensions of design, stating that 'since human expressions cannot be mapped one-to-one with expressions of the robot, we abstracted the human expressions first' [93]. The same authors defend that the design of behaviour and expressions of robots should be a blend between an artistic approach and an iterative cycle to evaluate and refine the result, which follows the usual practice both in engineering and usability design. They also consider that using virtual 3D models for animating and visualizing the expressions of a robot is useful, especially if the virtual model is designed with resemblance to the real physical model and its behaviour.

An important expressive feature in robots that is absent in human expressivity is the use of lights and sounds. Bethel [94] has studying the expression of emotions in robots that do not possess typical expressive capabilities fuck a face or arms. Her work focuses on robots that are mean to be functional, such as search-and-rescue or military robots, and how to use multi-modal expression for the correct communication of the robot's emotional state and empathic behaviour.

Saerbeck and Bartneck have also studied an abstraction of robot expression by attempting to correlate robotic motion with the perceived emotions by the users [95], and concluded that there exists a correlation between a robot's acceleration, and the perceived arousal.

Hoffman & Ju have presented some techniques, especially based on previous experiences, about designing robots with their expressive movement in mind [96]. They provides useful insights on how the embodiment and expressive motion are tightly connected, and how the design of expressive behaviour may be considered as part of the design of the actual robot, and not as an after-step.

Knight has developed the Computational Laban Effort (CLE) framework [97]. Based on the Laban Effort System, it allows a low degree-of-freedom robot to modify its task motion in order to convey varying internal states. The main goal is to enable an expert system from dance and acting training to be used in robotic systems, while prioritizing task completion over expression. The process does however require multiple steps and procedures in order to implement the framework into one particular robot, for a particular task, as described by the following five steps:

1. Select the Laban features that the robot can use, given its degrees of freedom and task;
2. Specify the motion generation parameters for each feature;
3. Calibrate the motion generation with human experts;
4. Follow a user-centred or interactive design methodology, or machine learning approach to establish a Laban Effort parametrization for each desired communicative state;
5. Deploy the expressive motion to the robot's behaviour system.

3.4 Animated Robots

Various authors have already presented a large range of robots and papers describing the design and use of robots' expressive behaviour. In particular, Schulz, Torresen & Herstad have recently provided a literature review on the use of animation techniques in human-robot interaction user studies [98]. We highly recommend the interested reader to consult that document for an extensive read on the topic. In this section we present a selected short review of various existing social robots applications with emphasis on their expressive features and capabilities.

Out selection includes not only papers referring to studies in the field of HRI, but also to companies and research laboratories that produce robots or are related to the field of HRI, in order to gather direct references to robots (i.e. if it is a commercial product), or to its original release (i.e. if it is produced by a particular laboratory). We did not include ones that we considered too outdated, nor ones designed with very domain-dependant expressive features, both of which would not provide such a valuable knowledge for general use at date (e.g. expressivity towards children with autism). We further recommend the reader to visit *The Robot-Facebook*[3] website, which collects a varied set of real and fictional robots, organized into various categories. The robot list below is order by first release or public presentation date, and within it is organized in alphabetical order. Some exceptions might occur where more than one robot is presented together, when they have a close relation and share the same creator.

Aibo[4] (1992), Sony

Sony is famous for creating AIBO early in 1992, the dog-shaped robot that can play tricks and is aimed at becoming a member of our family. Since then it has evolved through many versions, with its latest (fourth) generation dating from 2018. In this paragraph we will refer to the AIBO ERS-7 from 2003, which belongs to the third generation [99]. AIBO has 20 degrees of freedom, with 3 DoFs in each leg, a pan, tilt and roll joints on the head, a 2-DoF controllable tail, plus an additional joint for the mouth, and one for each ear. Its head also features two LED panels that can be used to display various emotional eye shapes.

PaPeRo (1997), NEC

The PaPeRo is a small childcare mobile robot with a minimal face that contains only two eye-spots (only *holes*, and not actual stylized eyes) an LED for the mouth, and a pair of LEDs for the ears [100]. Its head can tilt up and down, and pan sideways. It was used in various different scenarios with children in which it was controlled by an operator. Within those, it could perform speech, sound and music effects, move around, and use its mouth and ears LEDs to express itself, while reacting to face recognition, some keywords (through speech recognitions), and to touch, using sensors spread around its body.

eMuu (2002), Christoph Bartneck

Bartneck created the eMuu robot as an abstract face containing only an eyebrow and a lip, which was sufficient to express emotions [14]. eMuu can tilt and pan its head in order to exhibit *keep-alive* behaviour. Its internals were built using Lego Mindstorms, while its outside (hull) was made of soft polyurethane.

Roomba[5] (2002), iRobot

The popular autonomous robotic vacuum cleaner Roomba is considered to be the first ever robot to succeed in the consumer and home appliance market. it has also been used in studies regarding domestic robot ecology, and

[3] `http://robotfacebook.edwindertien.nl` (accessed January 12, 2019)
[4] `http://www.sony-aibo.com` (accessed January 12, 2019)
[5] `http://www.irobot.com` (accessed January 12, 2019)

how people accept and adopt robot devices in their homes in the long term [101]. Although it appears to be an expressionless robot, it is, in first instance, a mobile robot which can move forwards and turn 360°in place, thus allowing it to perform expressive motion through its trajectory. Additionally, the robot allows to be hacked and extended. Roomba has been coupled with an RGB-LED mood ring, which allows it to communicate using a multi-color halo [102], and has also been added an expressive tail, which, inspired by canine behaviour, can wiggle to signal certain events and emotions, such as wiggling happily when it finds dust and dirt, or exhibiting an fearful, apprehensive posture when it becomes stuck [103].

Interactive Theatre (2003), MIT Media Lab

In 2003, Breazeal and colleagues presented the Interactive Theatre [29]. This is one of the first robot animation systems to be developed with interactivity in mind, by blending AI and an artistic perspective. Several robotic anemones were animated in collaboration with animators to portray a lifelike quality of motion while reacting to some external stimuli like the approach of a human hand. These animations were driven by parameters which were controlled by a behaviour based AI to dynamically change the appearance of its motion depending on events captured by a vision system [104].

Kismet[6] (2003), MIT MedialLab

The Sociable Machines Project at the MIT Media Lab developed Kismet, an expressive robotic face with stylized anthropomorphic features [105, 30]. It is equipped with visual, auditory and proprioceptive capabilities, and can express itself through vocalizations, facial expressions, gazing and eye- and head-direction. It contains a total of 15 DoFs in its face, which control its ears, eyebrows, eyelids, lips and jaws. Its affective state is based on a PAD space (arousal, valence and stance), and specifies 10 emotions: fear, sorrow, surprise, boredom, joy, interest, disgust, calmness and anger. Based on each emotions PAD position, Kismet can also display emotions that result of blending between these.

QRIO[7] (2003), Sony

QRIO is a bipedal humanoid entertainment robot, approximately 60cm tall, with legs that can be used for postural expression of to walk and run at up to 23 cm/s, arms that can be used to perform gestures and balance, and a face that contains only two illuminated eyes slots [106].

Leonardo[8] (2004), MIT MedialLab

One of MIT Media Lab's most famous robots is Leonardo, a 65-DoF robot that is approximately 75cm tall and has been specifically designed for expressive social interaction with humans [107, 108, 104]. It was designed as an anthropomorphic furry creature in collaboration with Stan Winston Studio [9]. Leonardo can interact and communicate with people through speech, vocal tone, gestures, facial expressions, and also perform simple object manipulations. Inspired by Kismet, its computational architecture contains affective factors that influence and interact with the cognitive elements of the system. It affect space is however modelled in a two-dimensional system (valence and arousal) over which seven facial poses are defined.

Robosapien[10] (2004), WowWee

The Robosapien X is a biomorphic toy robot that can either be programmed to perform entertaining moves, or be

[6] `http://www.ai.mit.edu/projects/sociable/baby-bits.html` (accessed January 12, 2019)
[7] `http://www.sony.net/SonyInfo/CorporateInfo/History/sonyhistory-j.html` (accessed January 12, 2019)
[8] `http://robotic.media.mit.edu/portfolio/leonardo` (accessed January 12, 2019)
[9] `http://www.stanwinstonschool.com` (accessed January 12, 2019)
[10] `http://wowwee.com/robosapien-x` (accessed January 12, 2019)

remote controlled using a control with 21 different buttons. It was used in an early example of trying to evoke the illusion of life in robots through character animation practices [109, 110]. However the authors explore a very shallow concept of the illusion of life, and present character animation practices as the designing of animations for robots using 3D animation software (Autodesk Maya), from which the exported motion requires a post-processing step to correct for the robot's balance.

iCat[11] (2005), Philips Electronics

The iCat is a plug and play desktop robot that can perform facial expressions, move its head around, display expressive lights, and to respond to touch on its paws. As the name suggests, it is designed to resemble a cartoonish cat, with expressive eyebrows and a eyelids, orientable eyes and its rubber lips that allow to portray smooth mouth shapes. It became notoriously useful and popular as a robot that is ideal for interaction with children, and in scenarios in which personality and animation qualities play a role [9, 111]. In particular, Leite et al. developed and autonomous scenario for long-term interaction in which the iCat plays and teaches chess to school children [112]. It explores an empathic model that drives the robot's emotional display and decision-making, by e.g. allowing the child to choose another option after selecting a bad move, feeling sad when the child performs bad, and remembering and recalling on previous interactions.

Keepon[12] (2007), Hideki Kozima, National Institute of Information and Communications Technology, BeatBots LLC

The Keepon is a small yellow table-top robot that is shaped like two glued tennis balls (albeit yellow) [113]. However unlike tennis balls, its body is made of soft rubber, which allows it to bend, squash and stretch while performing motion based on its four motors: pan, tilt, roll and bounce (stretching up and down). It has two decorative eyes and a nose. It is distinguished as a very simple, minimalist, small and affordable robot that is also child-friendly. With that purpose, it was used on a collaborative build-a-rocket computer game [114]. As a minimalistic, abstract robot, it was used to explore Laban Efforts by Knight & Simmons [115]. Because the Keepon contains no sensors, it relies on external ones such as a Microsoft Kinect, to provide user-perception. It was also used in the E-Fit scenario, in which the robot is delivered to people's homes so that children can interact with it daily. The robot takes the role of an *alien* whose space ship broke down on Earth and who needs help from the child in order to return home. The child helps the robot by exercising routinely and transferring the acquired "energy" to the robot by doing so [116].

Nexi[13] (2007), MIT Media Lab

The Nexi robot, created by MIT Media Lab, is an anthropomorphic robot consisting of a very expressive face, placed on a self-balancing two-wheeled mobile base which also has two arms and hands [117, 104]. Its head and neck are highly polished and white, with smooth edges that convey a stylized human like look. In order to be expressive it can move its eyebrows, mouth, and orbit its eyes, all while performing 3D motion with its head (pan, tilt, roll).

Paro[14] (2007), Intelligent Systems Research Institute, AIST

The Paro is a seal-shaped therapeutical robot that is considered to convey a healing effect, especially in the elderly population [118]. Its body is soft and furry, and is equipped with tactile sensors that allow it to respond to human touch. It also contains a light sensor, sound source direction detection, speech recognition and balance sensors.

[11] `http://robotfacebook.edwindertien.nl/product/icat` (accessed January 12, 2019)
[12] `http://beatbots.net/my-keepon` (accessed January 12, 2019)
[13] `http://robotic.media.mit.edu/portfolio/nexi` (accessed January 12, 2019)
[14] `http://www.parorobots.com` (accessed January 12, 2019)

While shaped like an animal, it is not intended to move around freely. Instead it was designed to be held by a human or kept in its lap, and can express itself through neck movements, front and rear fin movements, and independent eyelid movements.

Zeno[15] (2007), Hanson Robotics

The Zeno Robokind Zeno is a humanoid robot with a mechanical structure that is similar to that of the NAO, but featuring a very expressive lifelike face. A particular work that we found explains how a closed-form IK system was built for such a robot [119]. A closed-form solution is an analytical solution, made specifically for this robot. However, the authors describe how they captured a human motion using the Microsoft Kinect, and then transferred that motion to the Zeno using the mentioned closed-form IK solution.

AUR (2008), MIT Media Lab

The AUR is a robotic desk lamp with 5 DoFs and an LED lamp which can illuminate in a range of the RGB color space [120]. It is stationary and mounted on a workbench. It is controlled through a hybrid control system that allows it to be used for live puppeteering on stage, as an actor. The purpose was to allow the robot to be expressive while also being responsive. In general, a responsive robot would not have proper control for its expressivity, while a properly expressive robot would have to rely on pre-designed animations, and as such. Considering their case, robots used on stage were generally controlled by several extensively trained puppeteers, and still, proper eye contact was virtually impossible to achieve. For AUR, the motion was composed through several layers. The bottom-most layer moves each DoF based on a pre-designed animation that was made specifically for the scene of the play. If the robot was set to establish eye contact, several specific DoFs would be used to calculate an IK solution using CCD, and the result was used to override the motion of the *scene* layer. A final **animacy** layer added smoothed sinusoidal noise, akin to breathing, to all the DoFs, in order to provide a more lifelike motion to the robot.

NAO[16] (2008), Aldebaran Robotics, Softbank Robotics

The NAO robot is one of the most popular robots used in research and in human-robot interaction. We will mention some works that we have found to contribute specifically to autonomous expressivity using this robot. Our own previous work has included NAO into the Nutty Tracks animation engine that allows animators to create expressive behaviours for the robot and to merge them with procedurally-based motion [121]. It has also been used within the SERA ecosystem for the EMOTE project, in which NAO became an autonomous empathic robotic tutor for classrooms [42]. In EMOTE, the robot interacted simultaneously with two children and with a virtual learning application that was running on a large touch table. The behaviour of the robot was designed to drive the children's attention to the learning application and learning goals, while still providing social behaviour as both a peer and a tutor. In general attention was directed through gaze and gestures, with gaze being controlled with a gaze-state machine that considered gaze-breaking, or gazing towards the students' actions (e.g. gazing at them when they were speaking, or directly at the point of the touch table where they clicked). The robot relied on a Microsoft Kinect and on lavalier microphones in order to manage its gazing between the students' actions, the task, and its own actions. In [122], NAO was used as an autonomous social robotic tutor for second language learning in a one-to-one setting. Its behaviour was also designed for high nonverbal immediacy, with random gazes towards the child, gestures during speech, forward lean of the body, and keeping the arm's motors noises low to give the impression of being relaxed. The work by [123] also presents NAO as a robotic tutor, in a factions calculation scenario for school children. A

[15] `http://www.hansonrobotics.com/zeno` (accessed January 12, 2019)

[16] `http://www.softbankrobotics.com/emea/en/nao` (accessed January 12, 2019)

less recent but relevant work that explores the expressivity of the NAO robot is [124], where the LEDs of the robot's eyes are used to provide it with eye shapes. While NAO does not have mechanical eyes, and was not designed to provide expression through them (except for changing the colors of the LEDs), the authors suggest an interesting control of individual LEDs to create expressive eye shapes for Neutral, Happiness, Sadness, Anger, Fear, Disgust and Surprise.

Simon (2008), Georgia Institute of Technology

Gielniek, Thomaz and colleagues have been exploring ways of actually algorithmizing some of the principles of animation as motion signal processing algorithms which can be used in real time, as they demonstrate using the Simon robot (e.g, [32]). In particular they created an algorithm that generates exaggerated variants of a motion in real-time, relying only on one tuning parameter α, as the authors believe optimality is context-dependent and so the exaggeration factor should be controllable.

CoBot[17] (2009), Carnegie Mellon University

The CoBots are a family of mobile indoor service robots that have been developed at the Carnegie Mellon University since 2009. They operate on a four-wheeled omnidirectional drive, have a touchscreen for direct user input, and a container to carry and deliver diverse objects [125]. The CoBot robots perform autonomous indoor localization in order to navigate through multiple levels of a building. In order to use the elevator, they resort to shared autonomy, i.e., while placing themselves in from of an elevator door, they will 1) call out for some by-passer to press the elevator button; 2) ask someone in the elevator to press the button corresponding to its destination level. When faced with an obstacle, it can also perform an auditory notification so that people can clear away its path. Additionally, light can be appended to it in order to further reveal its internal state [126]. Thanks to its omni-drive, it can perform complex motions, which may also be exploited for their expressivity. Knight & Simmons have explored the use of Laban Effort Features to perform expressive motion in the XY (horizontal travel) and in the θ (rotation) dimensions.

HERB (2009), Carnegie Mellon University

HERB, the Home Exploring Robotic Butler is a mobile robot with two large arms with hands that can grasp objects (the first version had only one arm) [127, 128]. The challenge of providing legible and predictable motion to autonomous collaborative robots has been addressed by Dragan et al. using the HERB robot [129], which demonstrates the benefits of including such properties into motion planners. Their technique however focuses on these two properties in particular, and rely on motion-planning for trajectory generation. Admoni has also used HERB to study nonverbal communication in socially assistive HRI [130].

iCub[18] (2009), Italian Institute of Technology

The iCub is a humanoid robot that was developed for research in embodied cognition [131]. It mimics a three and a half year old child, is able to crawl and sit while manipulating objects.

Mung (2009), Korea Advanced Institute of Science and Technology

The Mung robots are a set of three simple shaped robots consisting only of a round body and two eyes [132]. The three versions of it correspond to a bread-bun-shaped body, to an egg-shaped body, and to a pot-shaped robot. The bodies are made of translucent acrylic in order to allow it to display *internal* emotional states using color LEDs. The eyes are made of metal. Using the internal LEDs, the robots can exhibit various expressions which are seen through its translucent body as *bruises* or *blushes*.

[17] `http://www.cs.cmu.edu/~coral/projects/cobot` (accessed January 12, 2019)

[18] `http://www.icub.org/` (accessed January 12, 2019)

Snackbot (2009), Carnegie Mellon University, United States Naval Academy

The Snackbot robot is a service robot with a minimalistic anthropomorphic shape, with a simple head, and two non-functional arms that hold a tray upon which it can carry objects [133, 134]. In particular, as its name implies, it was created to deliver various types of snacks within two connected buildings. It uses a differential-drive mobile base along with bumpers, sonars and a laser scanner in order to navigate while detecting and avoiding collisions. The head was designed with interesting factors in mind. It is round, but wider than taller, in order to suggest it being a young, friendly robot. The details of the eye sockets were kept minimal so that the customers would not develop false expectations about the robot's intelligence. A 3x12 LED display was placed on the mouth region in order to convey more meaningful expression such as lip shapes, colours and movement during interaction. Finally, minimalistic non-functional ears were also added so that the customers would understand that the robot could hear them.

AIDA[19] (2010), MIT Media Lab

Various interactive social robots have been created at MIT's Media Lab [104], in particular the AIDA, which is a friendly driving assistant for the cars of the future. AIDA interestingly delivers an expressive face on top of an articulated neck-like structure to allow to it move and be expressive on a car's dashboard.

PR-2[20] (2010), Willow Garage

The use of animation principles was explored by Takayama, Dooley and Ju [31] using the PR-2 robot. This is a large mobile robot with two arms, that can navigate in a human environment. A professional animator collaborated on the design of the expressive behaviour so that the robot could exhibit a sense of *thought*, by clearly demonstrating the intention of its actions. *Thought* and *Intention* are two concepts that are central in character animation, and in the portrayal of the illusion of life. In this case they focused on making both the intention of the robot, and the robot's reaction to its own action more readily apparent to interactants and bystanders, in order to facilitate coordination of their actions with those of the robot. They formulation a robot's animations as functional motions (e.g. grabbing a door knob), and expressive motions (e.g. looking at the door handle and scratching its head), although these are not completely separate concepts (e.g. in some situations being expressive is a functional part of the task). The conducted a study measuring readability of robot forethought, perception of robot forethought, and perception of robot reaction, and conclude that showing forethought makes a robot seem more appealing and approachable.

Robovie[21] (2010), Vstone Co., ATR

Vstone Co. and ATR created the Robovie (R3) robot, which is a healthcare and guide robot. Its mobility is enabled by an omni-wheel base, while its body is anthropomorphic, with arms and a nearly faceless head. Its face contains only two eye placeholders, which do not move, but instead, contain cameras that allow it to perceive its surrounding environment.

Shimon and Travis (2010, 2012), Georgia Institute of Technology, IDC Herzliya

Hoffman has created interactive robots that behave in a musical environment. Shimon is a gesture based musical improvisation robot that plays marimba [135]. Its behaviour is a mix between his functionality as a musician, for which he plays the instrument in tune and rhythm, and being part of a band, for which he performs expressive behaviour by gazing towards his band mates during the performance. Travis is a robotic music listening companion

[19] `http://robotic.media.mit.edu/portfolio/aida` (accessed January 12, 2019)

[20] `http://www.willowgarage.com/pages/pr2/overview` (accessed January 12, 2019)

[21] `http://www.vstone.co.jp/english/products/robovie_x` (accessed January 12, 2019)

also created by Hoffman, that acts as an interactive expressive music dock for smart phones [136]. The system allows a user to dock a smartphone and request it to play a music from some play-list. The robot plays it through a pair of integrated loudspeakers while autonomously dancing to the rhythm. The music beat is captured by real-time analysis in order to guide the robot's dance movements. Those movements are simple "head banging" and "foot tapping" gestures that are easily programmable. Both robots are controlled by a similar control system that relies on *dead-reckoning*. Beat-matching is insured by an *overshoot and interrupt* approach, and the motion is smoothed using a high-frequency trajectory interpolator, insuring that the final motion is rendered at a fixed rate of 50Hz. The authors present two main advantages to this approach: (a) they are "*able to generate a more expressive spatio-temporal trajectory than just a trapezoid, and we can add animation principles such as ease-in, ease-out, anticipation, and follow-through*"; and (b) "*since the position of the sliders is continuously controlled, collisions can be avoided at the position request level*".

Sparky[22] (2010), Walt Disney Imagineering Research

The Walt Disney Imagineering Research & Development has developed the stylized anthropomorphic Sparky Minimatronic™. It is a robotic marionette that uses 18 R/C servo motors: 4 for each arm, 2 for each leg, 2 for the neck, 1 for the mouth, 1 for the eyes, 1 for the eyelids, and 1 for its spine [137]. Its puppet-like structure allows it to be lightweight and to perform highly dexterous, fluid and natural movements.

EMYS (prototype, commercial)[23] (2010, 2018), Wrocław University of Technology, EMYS Inc.

The original EMYS robot was developed back in 2010 for the EU FP7 LiREC project[24] [138]. Pereira and colleagues showcased an EMYS robot that continuously interacts with both users and the environment while playing a multi-player board-game, in a way to provide a more lifelike experience[25] [139]. This was the first autonomous robot to interact simultaneously with several human players through a video game running on a large touch-table. The social nature of this application required EMYS to be able to blend several animation modalities in real time, such as gazing towards a person while performing an expressive emotional animation, or changing the overall look of its idle behaviour in order to portray its internal emotional state, while still reacting to the presence of the other players. Such highly expressive requirements lead the creators to also drive inspiration from animation principles and practices, and designed EMYS's animations following an artistic approach, thus becoming one of the first autonomous social robots to be animated using professional animation software [22]. These technical developments were later extended and became the Nutty Tracks animation system that can be used for any robotic embodiment [121]. More recently, EMYS has been used along with Nutty Tracks to play the *Sueca* card game, in which human users play with the robot using real physical cards [140]. Note that all these works refer to the prototype Emys (2011), which was developed in two versions: MkI and MkII, while we add a reference to the new commercial version of Emys (2018) which is being released soon and although it contains many of the features of the prototype Emys, its design and build was revamped and some of its expressive channels were removed or modified (e.g. no neck tilt, but eyes became LCD displays and includes and additional screen display)

Probo (2011), Ghent University, Vrije Universiteit Brussels

The Probo huggable robot is used for robot assisted therapy applications. It was designed with the purpose of

[22] `https://rasc.usc.edu/sparky.html` (accessed January 12, 2019)
[23] `http://emys.co` (accessed January 12, 2019)
[24] `http://lirec.eu` (accessed January 12, 2019)
[25] `http://vimeo.com/56200151` (accessed January 12, 2019)

exhibiting the illusion of life so as to facilitate engaging interactions with humans, which is especially relevant within social therapy applications [141]. The robot has a fully actuated head with 20 DOF capable of communicating emotions an attention, and is implemented as a semi-autonomous system, with shared control with a human operator. The system is composed of various modules such as Perceptual-System, a Control-System, an Expressional-System, and a Motor-System. The first two working together allow the robot to track a certain face or object, or to gaze towards a the direction of a sound source. Within the Control-System there is also an emotional state that is based on internal needs (modelled as a homeostatic system), which are influenced by the perceived actions, and influence the selection of its expressive behaviour (facial expressions). The authors argue that this composition in which each block has its own functionalities allows them to be used with other robots or agents. The animation system allows to combine pre-designed motion sequences with direct control from the operator. It includes several graphical tools that allow the authoring of keyframe-based animations (Sequence Editor), and real-time operation and manipulation of how these are mixed and played-back on the robot (Animation Player and Motion Mixer). The various outputs are combined using a Combination Engine, which considers several types of keyframes to specify how overlaid animation sequences should be blended. In order to allow a smooth initiation and termination of an animation sequence, when played over another previous sequence or expression, each sequence must be authored in order to consider an initial time-interval during which the underlying motion can be blended out of, and then back into. The resulting motion is filtered using a cascade of first order software low-pass filters, with a different attenuation parameter used for each body part.

Sphero[26] (2011), Sphero

Sphero is a spherical robot launched in 2011 that can move by rolling under control of a smartphone or tablet. While it seems not to contain any expressive features it can in fact portray an expressive character through motion and through its lights. Faria et al. developed communicative intentions using the Sphero 's motion in order to get attention, to convey that it wanted a person to follow it and to express happiness and sadness [142].

Baxter[27] **& Sawyer** [28] (2012, 2015), Rethink Robotics

Both Baxter and Sawyer were developed by Rethink Robots, a company founding the Rodney Brooks, the same founder of iRobot, which created Roomba. Their first robot was Baxter, an industrial collaborative robot meant to work together with employees at factories and warehouses. It has two arms and a screen that can be used to display information or an expressive face to its users. It is supported by an adjustable pedestal, which can make it up to around 1.9 meters tall, and weigh around 140Kg. Its purpose is mostly to perform repetitive tasks such as loading, unloading, sorting and handling materials, and can be programmed to do so on-site by an employee without much effort or training. Baxter was later discontinued and succeeded by Sawyer, which is slightly more compact given that it has only one arm, but its design and functionally meet the same targets as Baxter's. Besides working in industrial settings, it has also been used to perform handwriting tasks [143] or to play games with people [144].

Dragonbot[29] (2012), MIT Media Lab

The Dragonbot, developed by MIT's Media Lab, is a small furry desktop robot with a dragon-like appearance [145]. It is particularly aimed at child applications such as educational games [146] The robot can perform squash-and-

[26] `http://www.sphero.com` (accessed January 12, 2019)
[27] `http://robots.ieee.org/robots/baxter` (accessed January 12, 2019)
[28] `http://www.rethinkrobotics.com/sawyer` (accessed January 12, 2019)
[29] `http://robotic.media.mit.edu/portfolio/dragonbot` (accessed January 12, 2019)

stretch through its internal Delta robot platform, which can move its body in 4 DoF. A stretchy synthetic fur allows its shape to go along with the movement with a natural look. It additionally has a neck tilt DoF and a wiggable tail. Two hands are used solely as adornments given that they do not have any actuated movement. One of its most interesting features is that it is fully controlled through a smartphone which is inserted in a slot on its face, and therefore the smartphone's screen acts as the robot's expressive eyes.

Karotz[30] (2012), Violet (*extinct*), Mindscape

The Nabaztag robots are ambient smart electronic devices shaped like a rabbit. Karotz is the third and final generation. Its two ears can be controlled to portray different stances, along with internal RGB lights the illuminate portions of its body. However most of its communication is performed audibly, either through speech or by playing sounds and music. Even as a simple platform, its communicative capabilities and minimalistic, stylized form and expressivity have lead it to be used in user-studies directed at the usefulness, adoption and engagements that people feel towards such having devices in their households [147, 148].

Furhat[31] (2013), Samer Al Moubayed, Furhat Robotics

The Furhat is a hybrid robotic head that provides rich and fluent multimodal interaction using speech, head motion and facial expressions [149]. It's face is backprojected into a physical, translucent mask, using an internal micro projector. It is then digitally animated in a way that matches the design of the physical mask. The backprojection face method solves the problem of fluent and natural facial expression, given that such face and expressions are all rendered using CGI techniques, and therefore it is not constrained by the dynamics of mechanical servos or artificial skin. Displaying a virtual face on a 3D surface, in contrast to using a 2D display also allows it to perform better in multiparty interaction by eliminating the Mona Lisa gaze effect [150].

Jibo[32] (2014), Jibo Inc.

The robotic home assistant Jibo presented in 2014, created by MIT Media Lab professor and social robotics pioneer Cynthia Breazeal. It is a small tabletop robot that is somewhat shaped like a spotlight. Its *light* is however an LCD touchscreen which allows for bidirectional interaction, i.e., for the robot to display information, expressive and emotions, but also for the users to directly interact back with it. Additionally it has two speakers that allow it to speak, play sounds and music. In order to perceive the users and surrounding environment, it contains full body touch sensors and can perform 360°sound localization. One of the most interesting features in Jibo's design is the way its body with a tilted *triple-pan* full-revolute mechanism. Its body can be seen as composed of two parts: its torso and its head. Both the torso and the head can pan, however both pan at an angle, which allows Jibo to modify the apparent shape of its body while these two joints pan in inverse directions. Additionally the base of the torso can also pan in order to perform these shapes towards any direction.

Pepper[33] (2014), Aldebaran Robotics, Softbank Robotics

The Pepper robot is developed by Softbank Robotics (formerly Aldebaran Robotics) and is a modern stylized humanoid following the design of the same company's NAO robot. However Pepper is 1.2 meters tall, and stands on a mobile base, which gives it a better ability to interact with human in their natural environments [151, 152]. Pepper can express itself through its omnidirectional motion [153], using its arms to perform gestures, speech, a

[30] `http://www.nabaztag.com` (accessed January 12, 2019)
[31] `http://www.furhatrobotics.com` (accessed January 12, 2019)
[32] `http://www.jibo.com` (accessed January 12, 2019)
[33] `http://www.softbankrobotics.com/emea/en/pepper` (accessed January 12, 2019)

tabled that is attached to its chest, or its face which can either move around (pan, tilt) or by changing the colour of the LEDs surrounding its eyes and ears.

Kip1[34] (2015), IDC Herzliya

The Kip1 desktop robot was designed to promote non-aggressive conversation between people. By monitoring a conversation's tone, it reacts with an emotional display of fear whenever the conversations seems aggressive, otherwise performing a calm behaviour designed to communicate curious interest. The robot's figure is reminiscent of a lamp and designed to be both *peripheral*, *evocative* and *fragile*. It has a pan base that can rotate the robot towards any direction, a tilting head mechanism that is activated by a string, and a multi-action linkage that allows the robot to seem to stretch and squash. Additionally, the head, being modelled as a passive DoF, reacts to the robot's shaking and gravity, giving it a more natural feeling, in what can be seen as the *overlapping* principle of animation (despite it being called *secondary action* by the authors, upon consulting [4] or [154] one finds that it is in fact *overlapping action*). [155]

Cozmo & Vector[35] (2016, 2018), Anki

Both Cozmo and Vector are a mix between a home assistant and a smart toy robot, both developed and sold commercially by Anki. Cozmo was their first robot, and was succeeded by Vector. Both robots follow the same principles and design, although Vector contains additional computational features such as speech recognition. They are designed as a small truck, with an active forklift that can both play a functional roll (interact with and lift up objects) and an expressive roll (raise and shake like they were its hands). It moves on two tank-like tracks, which allows it to move forwards and backwards while turning and also to rotate in place. Its most distinguishing feature is the expressive screen, which is used mostly to display its eyes, but can also display other information, icons, animations, expressions, ir even be used as a game screen. All the expressions were carefully designed by professional animators with previous experience in the movie industry, which endowed Cozmo and Vector with the illusion of life, making them into some the most successful consumer robots for the home environment.

Buddy[36] (2016), Blue Frog Robotics

Bluefrog Robotics presented Buddy, a small home companion robot aimed at various tasks such as home assistance, elder care, entertainment and edutainment [156]. It has a mobile base with three wheels and sensors that allows it to perform house mapping, localization and collision and obstacle avoidance. Its head is supported on a pan-tilt neck and contains a screen where it displays and expressive face or applications and games.

Adelino[37] (2017), Tiago Ribeiro, INESC-ID

The Adelino robot was created by ourselves in the context of this thesis. It is a craft, 5-DoF articulated desktop robot with a simple face containing only two LEDs that can blink or change in brightness. Its design was inspired both by an animated snake, and from lines of action, an element that is intrinsically part of character animation. It has been used as autonomously for games and entertainment [157], which we will further describe in Sections 7.3 and 7.4. As an articulated structure, similar to an industrial manipulator, it requires inverse kinematics that can solve both for orientational and full-body postural targets in order to convey motivation, intention and emotional expression to its users. Given that it was developed in the context of this thesis, it will be further described in detail in Section 7.2.1.

[34] `http://milab.idc.ac.il/teaching/projects/kip` (accessed January 12, 2019)
[35] `https://anki.com` (accessed January 12, 2019)
[36] `http://www.bluefrogrobotics.com/robot` (accessed January 12, 2019)
[37] `https://tiagoribeiro.pt` (accessed January 12, 2019)

YOLO[38] (2017), Patrícia Alves-Oliveira, Cornell University

The YOLO (*Your Own Living Object*) is a minimal abstract robot designed for child-robot interaction [158, 159]. It is aimed at stimulating creativity during play, and uses implicit interaction modalities such as motion and lights to communicate with children. The lights can display different colours using various brightness levels to convey emotional expressions and personality. Using its three omni-wheels it can also *reactively* and *proactively* interact with children by navigating in any direction and by performing different navigation patterns at varying speeds.

Blossom[39] (2018), Cornell University

The Blossom is an open-source desktop social robot that uses a tensile mechanical structure to perform smooth, natural and safe movements, while being covered by handcrafted replaceable skins such as knitted or crocheted ones [160]. Besides its unique and distinguishable craft appearance, its internal mechanisms, built using servo-controlled strings and elastic bands, allow it to exhibit squash-and-stretch, and to perform a wide variety of natural movements that to not appear to be mechanical. Additional custom appendages can be added to its head, which can also be controlled using a servo. Its internal structure is built mostly out of laser-cut wood or acrylic.

CLOi[40] (2018), LG Electronics

The LG Electronics company has released a series (or family) of CLOi robots, designed and built with various purposes in mind [161]. This robot family includes a robot for the home, a lawn mower robot, a guide bot, a large cleaning bot, a transporter bot (e.g. to carry luggage), a serve bot (e.g., butler), and a cart bot (e.g., for shopping), each aiming at different application scenarios. All except the lawn mower and cleaner bot exhibit a display with a minimalistic face, representing a minimal face that is used as a point of attention for users, to express gazing direction, and various other facial expressions.

Walt[41] (2018), Vrije Universiteit Brussels, Hasselt University, Melexis, Audi, Softkinetic, AMS Robotics, Robovision

Walt is a social collaborative robot that that helps factory workers assemble cars [162]. It uses a screen to exhibit an expressive face, icons or short animations. Its body is a concealed articulated structure that allows it to gaze around at its co-workers.

Kiki[42] (2019), Zoetic AI

The Kiki robot created by Zoetic AI is a small desktop robot [163]. It has a fixed torso base with a pan-tilt neck and a head that resembles some animal with small ears such as a kitten or a fox. Its face is a screen with eyes that can display a wide variety of expressions, and its AI is focused on delivering personality, affection and attachment.

[38] `http://patricialvesoliveira.com` (accessed January 12, 2019)
[39] `http://guyhoffman.com/blossom-handcrafted-soft-social-robot` (accessed January 12, 2019)
[40] `http://www.lg.com/global/lg-thinq-appliances/cloi` (accessed January 12, 2019)
[41] `http://www.sulu.be/Walt` (accessed January 12, 2019)
[42] `https://www.kiki.ai` (accessed January 12, 2019)

Chapter 4

Robot Animation in Theory

4.1 The Principles of Robot Animation

In the context of social robotics, our understanding is that robot animation is not just about motion. It is about making the robot seem alive, and to convey thought and motivation while also remaining autonomously and responsive. And because robots are physical characters, users will want to interact with them. Therefore robot animation also becomes a robot's ability to engage in interaction with humans while conveying the illusion of life.

One of the major challenges of bringing concepts of character animation into Human-Robot Interaction (HRI) is at the core of the typical animation process. While in other fields, animation is directed at a specific story-line, timeline, and viewer (e.g. camera), in HRI the animation process must consider that the flow and timeline of the story is driven by the interaction between users and the artificial intelligence (AI), and that the spacial dimension of the interaction is also linked to the user's own physical motion and placement. Robot animation becomes intrinsically connected with its perception of the world and the user, given that it is not an absent character, blindly following a timeline over and over again. This challenge is remarkable enough that character animation for robots can and should be considered a new form of animation, which builds upon and extends the current concepts and practices of both traditional and Computer-Graphics (CGI) animation and establishes a connection between these two fields and the field of robotics and AI.

Various authors have previously moved towards the idea of robot animation, as a well specified field that could even include its own principles of animation. Van Breemen initially defined animation of robots as 'The process of computing how the robot should act such that it is believable and interactive' [9]. He also showed how 'Slow In/Out' could be applied to robots, although he called it Merging Logic.

Wistort has also proposed some principles that should be taken in account when animating robots, that do not accurately follow the ones from Disney [164]. His list of principles refer to 'Delivering on Expectations', 'Squash and Stretch', 'Overlapping/Follow through animation' (although he refers to it as Secondary Action), 'Eyes', 'Illusion of Thinking' and 'Engagement'. We actually consider that 'Delivering on Expectations' implies the same as Disney's 'Appeal', 'Illusion of Thinking' is related to 'Anticipation' and 'Engagement' refers to 'Staging'. Furthermore it is discussable whether or not Eyes must be part of robots at all.

Takayama et al. have focused on the use of Anticipation, Engagement, Confidence and Timing to enhance the

readability of a robot's actions [31]. Once again, the authors refer to 'Engagement', when in fact they do 'Staging'. Indeed, 'Staging' doesn't sound like a correct term to use in robot animation, because for the first time, we are having animated characters in real settings, and not on a set-up stage.

Mead and Mataric also addressed the principles of Staging, Exaggeration, Anticipation and Secondary Action to improve the understanding of a robot's indentions by children with autism [137]. For exaggeration, they were inspired by a process used in the generation of caricatures by exaggeration of the difference from the mean.

More recently, Gielniak et al. have sucessfully developed an algorithm that creates exaggerated variants of a motion in real-time by contrasting the motion signal, and demonstrated it applied to their SIMON robot [32].

Before we move on to define our principles of robot animation, we must first define robot animation. Most animation principles and guidelines report on designing particular motions. In the context of social robotics, our understanding is that robot animation is not just about motion. It is about making the robot seem alive, and to convey thought and motivation while also remaining autonomous and responsive. And because robots are physical characters, users will want to interact with them and therefore robot animation also becomes a robot's ability.

We therefore complement Van Breemen's definition by stating that *robot animation consists of all the processes that give a robot the ability of expressing identity, emotion and intention during autonomous interaction with human users*.

It is important to emphasize the word *autonomous*, as we don't consider robot animation to be solely the design of expressive motion for robots that can be faithfully played back. Instead it is about creating techniques, systems and interfaces that allow animation artists to program ***how*** the motion will be generated, shaped and composed throughout an interaction, based on the behaviours that are computed by the AI.

A general list of Principles of Robot Animation should also address principles related to human-robot interaction. In our list however, we refrain from deepening such topic that is already subject of intensive study [165, 166, 167, 168]. Instead, we have looked into principles and practices of animators throughout several decades, and analysed how the scientific community can and has been trying to merge them into robot animation.

We have noted that not all principles of traditional animation can apply to robots, and that in some cases, robots actually reveal other issues that had not initially existed in traditional animation. Most of these differences are found due to the fact that robots a) interact with people b) in the real, physical world.

The following sections reflect our understanding of how the Principles of Robot Animation can be aligned. Although they are stated towards robots, the figures presented show an animated human skeleton, as an easier depiction and explanation of use. Each principle is also demonstrated on the EMYS and the NAO robots in an online video[1], which can be watched as a complement to provide further clarification. The video first demonstrates each principle using the same humanoid character presented in this section, and then follows with a demonstration of each principle first using the NAO robot, and then using the EMYS robot.

4.1.1 Squash and Stretch

For robots to use this principle, it sounds like the design of the robot must include physical squashing and stretching components. However, besides relying on the design [96, 169], we can also create a squash and stretch effect by using poses and body movement.

[1] `https://vimeo.com/49122495` (accessed January 12, 2019)

In Figure 4.1 we can see how flexing arms and legs while crouching gives a totally different impression on the character. Following the rule of constant volume, if the character is becoming shorter in height, it should become larger in length, and a humanoid robot can perform that by correctly bending its arms and legs. Figure 4.2 presents a snapshot from the video[1] illustrating how this principle looks like on the NAO robot.

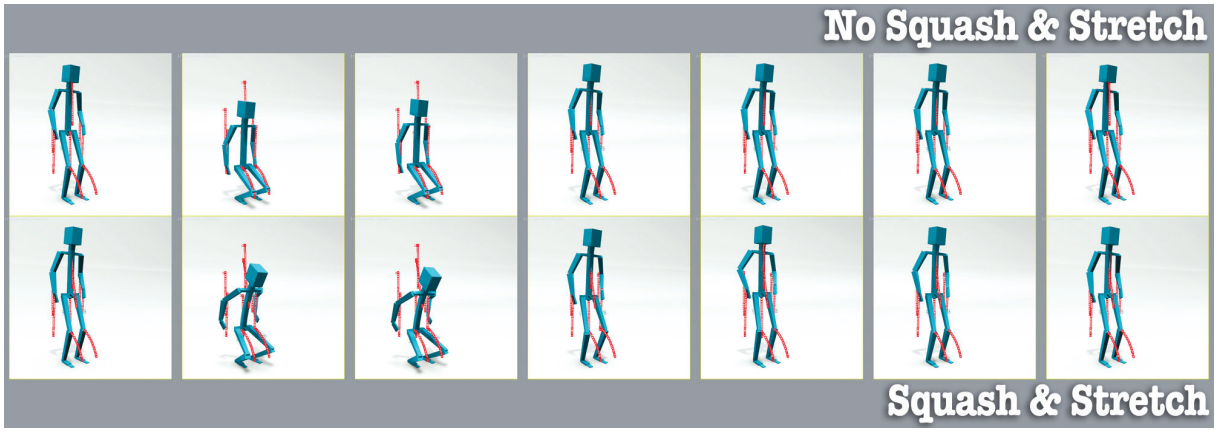


Figure 4.1: An animation sequence denoting the principle of Squash & Stretch. The red marks represent the trajectory of the most relevant joints.

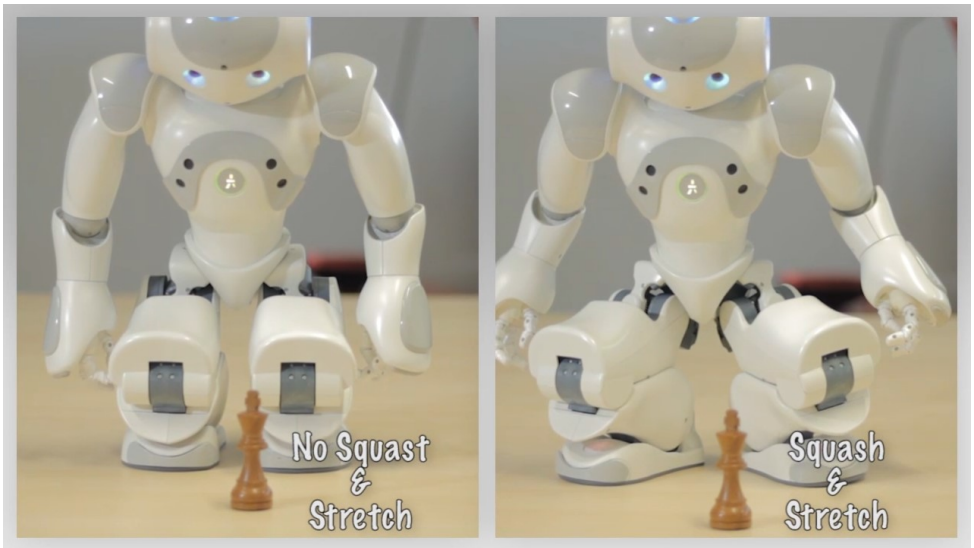


Figure 4.2: The principle of Squash & Stretch shown on the NAO robot.

4.1.2 Anticipation

Anticipating movements and actions helps viewers and users to understand what a character is going to do. That anticipation helps the user to interpret the character or robot in a more natural and pleasing way [31].

It is common for anticipation to be expressed by a shorter movement that reflects the opposite of the action that the character is going to perform. A character that is going to kick a ball, will first pull back the kicking leg; in the same sense, a character that is going to punch another one will first pull back its body and arm. A service robot that shares a domestic or work environment with people can incorporate anticipation to mark, for example, that it is going to start to move, and in which direction, e.g., before picking up an object, or pushing a button.

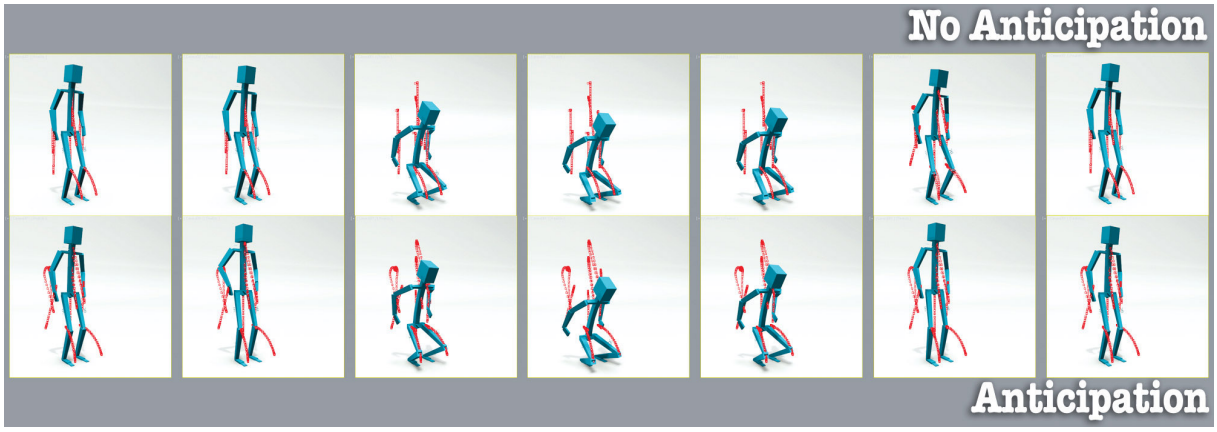


Figure 4.3: An animation sequence denoting the principle of Anticipation. The red marks represent the trajectory of the most relevant joints.

In Figure 4.3 we can see how a humanoid character that is going to crouch may first slightly stretch upwards.

The concept can be better explained by looking at a simple animation curve example. Figure 4.4 shows two animation curves for a 90 degrees rotation of an object. On the left we see a simple animation curve, and at the start and end keyframes we see the tangent of the curve at that point.

On the right we have the same keyframes, but the tangent of the initial keyframe has been changed. Just by adjusting this tangent we have made the object start by slightly rotating 10 degrees backwards before performing the mentioned 90 degrees rotation, thus creating an anticipation effect.

4.1.3 Intention

This principle was formerly known as Staging in the traditional principles of animation. In robots, staging results in several things. First, it notes that sound and lights can carefully be used to direct the users' attention to what it is trying to communicate. Second, if a robot is interested in, for example, picking up an object, it can show that immediately by facing such object [31]. In either cases, the key here is showing the intention of the robot.

We can see in Figure 4.5 a simple idea of a humanoid character that is crouching over a teapot to eventually pick it up. The character immediately looks at the teapot, so users know it is interested in it, and eventually guess that it is going to pick it up, much before the action happens.

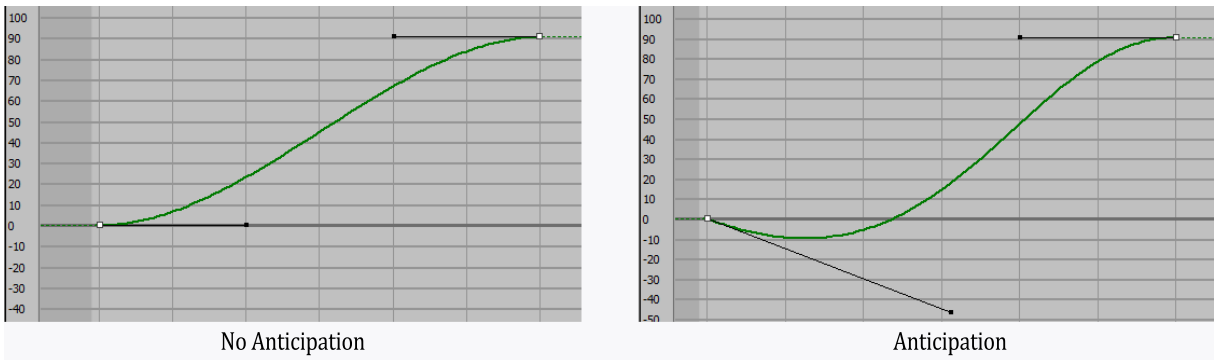


Figure 4.4: Animation curves demonstrating anticipation. The left curve does not have anticipation; The right curve does.

That connects Intention with Anticipation; the difference is that while Anticipation should give clues about what the robot is going to do immediately, Intention should tell users about the purpose of all that he is doing, as a pre-action, before the actual action starts. In a crouch-and-pick-up situation, for example, the robot will perform three actions - crouch, pick-up and stand. We should see Anticipation for each of these actions. The Intention, however, should reflect the overall of what the character is *thinking* - it will start looking at the object even before crouching, and will start looking at the destination to where it will take the object even before starting to turn towards that direction.

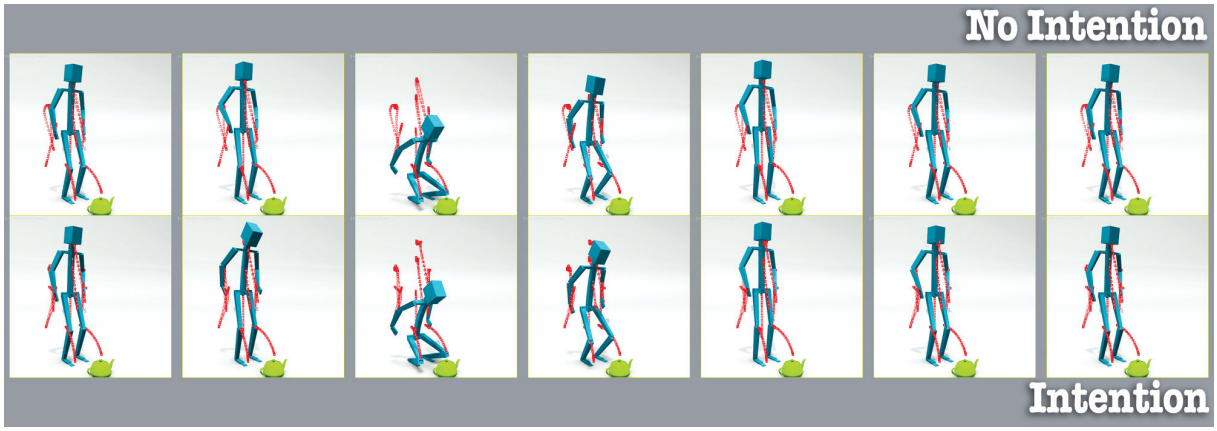


Figure 4.5: An animation sequence denoting the principle of Intention. The red marks represent the trajectory of the most relevant joints.

4.1.4 Animated, Procedural and Ad-hoc Action

This principle was adapted from the Straight-Ahead and Pose-to-Pose action and has strong technical implications on the animation system development. It originally talks about the method used by the animator while developing the animation. Straight-ahead animation is used when the animator knows what he wants to do but has not yet foreseen the full sequence, so he starts on the first frame and goes on sequentially animating until the last one. In pose-to-pose, the animator has pre-planned the animation and timing, so he knows exactly how the character should start and end, and through which poses it should go through.

In robots, this marks in the difference between playing a previously animated sequence, a procedural sequence, or an ad-hoc sequence. As a principle of robot animation, it results in a balance between expressivity, naturalness and responsiveness.

A previously *animated* sequence is self-explanatory. It was carefully crafted by an animator using animation software, and saved to a file in order to be played-back later on. That makes it the most common type of motion to be considered today in robot animation. However it suffers from a lack of interactivity, as the trajectories are played-back faithfully regardless of the state of the interaction. The motion is *procedural* when it is generated and composed from a set of pre-configured motion generators (such as sine-waves). On the other hand, it is *ad-hoc* if it is fully generated in real-time, using a more sophisticated motion-planner to generate the trajectory (e.g. obstacle-avoidance; pick-and-place task). We can say that playing an animation sequence that has previously been designed by an animator is a pose-to-pose kind of animation, while, for example, gaze-tracking a person's face by use of vision, or picking up an arbitrary object would be straight-ahead action.

A pose-to-pose motion can also contain anchor points at specific points of its trajectory (e.g. marking the beat of a gesture), so that the motion may be warped in the time-domain to allow synchronization between multiple motions. Those anchor-points would stand as if they were *poses*, or key-frames in animation terms. The concept of pose-to-pose can also become ambiguous in some case, such as in multi-modal synchronization, where, e.g. an ad-hoc gaze and an animated gesture should meet together at some point in time using anchor-points that define the meeting point for each of them. In that case, the straight-ahead action, planned ad-hoc, can result in an animated sequence generated in real-time, and containing anchors placed by the planner. From there it can be used as if it was a pose-to-pose motion to allow both motions to meet.

It currently sounds certain that the best and most expressive animations we achieve with a robot are still going to be pre-animated. However the message here is that these different types of animation methods imply their own differences in the robotic animation system, and that such system should be developed to support them.

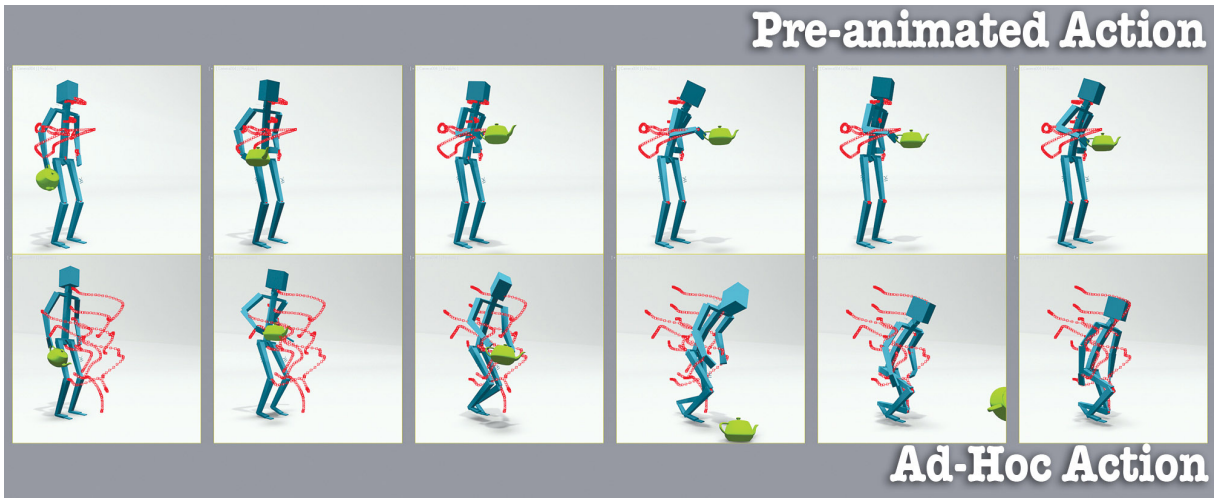


Figure 4.6: An animation sequence denoting the principles of Pre-animated and Ad-hoc Action. The red marks represent the trajectory of the most relevant joints.

In Figure 4.6 we can see on top a character performing a pre-animated and carefully designed animation, while in the bottom it is instantaneously reacting to gravity which made the teapot fall, and as such is performing an ad-hoc, straight-ahead animation.

While performing ad-hoc action, like reacting immediately to something, it might not be so important, in some cases, to guarantee principles of animation - if someone drops a cup, it would be preferable to have to robot grab it before it hits the ground, instead of planning on how to do it in a pretty way and then fail to grab it. In another case, if a robot needs to abruptly avoid physical harm to a human, it is always preferable that the robot succeeds in whatever manner it can. An ad-hoc motion planner therefore is likely to not contain many rules about animation principles, but act more towards functional goals (see the "Functional vs. Expressive Motion" section in [31]).

4.1.5 Slow In and Slow Out

For robot animation, Slow In and Out motion may me implemented within software in two different modalities: interpolation or motion filtering.

The former can be applied when the motion is either pre-animated, or fully planned before execution, so that

the system has the full description of the trajectory points. By tweaking the tangent type of the interpolation of the animation curve, it is possible to create accelerating and slowing down effects. By using a slow in and slow out tangent, the interpolation rate will slow down when approaching or leaving a key-frame. This means that in order to keep timing unchanged, the rate of interpolation will have to accelerate towards the midpoint between two key-frames. Van Breemen called this Merging Logic and showed how it could be applied to the iCat [9]. In alternative, when the motion is generated ad-hoc, a feed-forward motion filter can be used to saturate the velocity, the acceleration and/or the jerk of the motion.

A careful inspection of the red trajectories in Figure 4.7 will show us the difference between the top animation and the bottom animation. Each red dot represents an individual frame of the interpolated animation, using a fixed time-step. We can see that in the bottom animation the spacing between the frames changes. It gathers more frames near the key-poses, and less between them. This causes the animation to have more frames on those poses, thus making it slow down while changing direction. Between two key poses the animation accelerates because the interpolation generated less frames there.

This is more noticeable if we look at the animation curves. Figure 4.8 shows a very simple rotation without Slow-In / Out (left) and with (right). In the left image we used linear tangents for the interpolation method, while in the right we used smooth spline tangents.

We can see that with a linear interpolation, the curve looks straight, meaning that the velocity is constant during the whole movement. By using smooth tangents the movement both starts, stops and changes direction with some acceleration, which makes it look smoother.

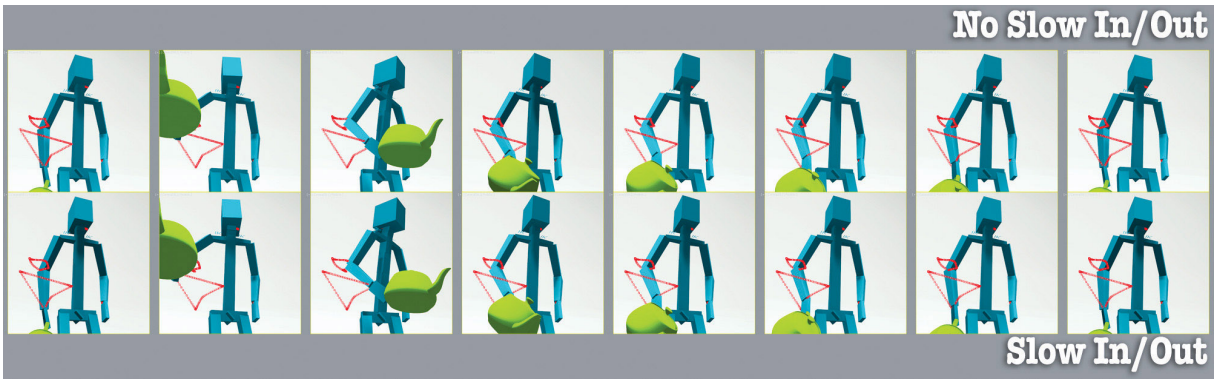


Figure 4.7: An animation sequence denoting the principle of Slow In/Out. The red marks represent the trajectory of the most relevant joints. Notice how more frames are placed at the points of the trajectory where the motion changes in direction, in particular within the triangular-shaped portion. More spacing between points, using a fixed time-step, yields a faster motion.

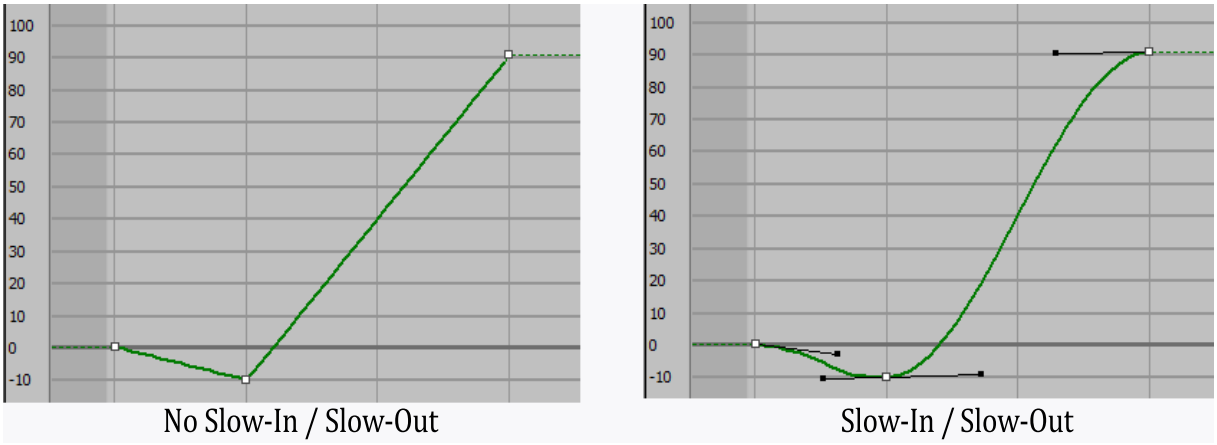


Figure 4.8: Animation curves demonstrating Slow In and Slow-Out. The left curve does not have Slow In / Out; The right curve does.

4.1.6 Arcs

Taking as example a character looking to the left and the right. It shouldn't just perform a horizontal movement, but also some vertical movement, so that its head will be pointing slightly upwards or downwards while facing straight ahead. We can see that illustrated in Figure 4.9.

This principle is easy to use in pre-animated motion. However, in order to include it in an animation system, we would need to be able to know in which direction the arcs should be computed, and how wide the angle should be. If we have that information, then the interpolation process can be tweaked to slightly bend the trajectory towards that direction, whenever it is too straight.

What actually happens with robots is that depending on the embodiment, it might actually perform the arcs almost automatically. Taking as example a humanoid robot, when we create gestures for the arms, they will most likely contain arcs, due to the fact that the robot's arms are rigid, and as such, in order for the them to move around, the intrinsic mechanics will lead the hands to perform arched trajectories. In traditional animation this principle was extremely relevant as the mechanics of the characters were not rigidly enforced as they are in robots. Arcs still pose as an important principle to be considered in robot animation, both for pre-animated motions and also as a rule in expressive motion planners.

Figure 4.10 shows a character gazing sideways. The yellow cone represents the gazing direction at each frame. The red curve illustrates the motion trajectory on the panning DoF (horizontally) and the Pitch DoF (vertically). On the top motion, no movement is performed on the Pitch joint (straight line). On the bottom motion, instead of performing only Yaw movement while looking around, the head also changes its Pitch between each keyframe of the Yaw movement.

4.1.7 Exaggeration

Exaggeration can be used to emphasize movements, expressions or actions, making them more noticeable and convincing. As such, it can also make robots seem more like actual characters and not just machines.

Although there are several levels of exaggeration, for robots it is interesting to look at exaggeration of actual

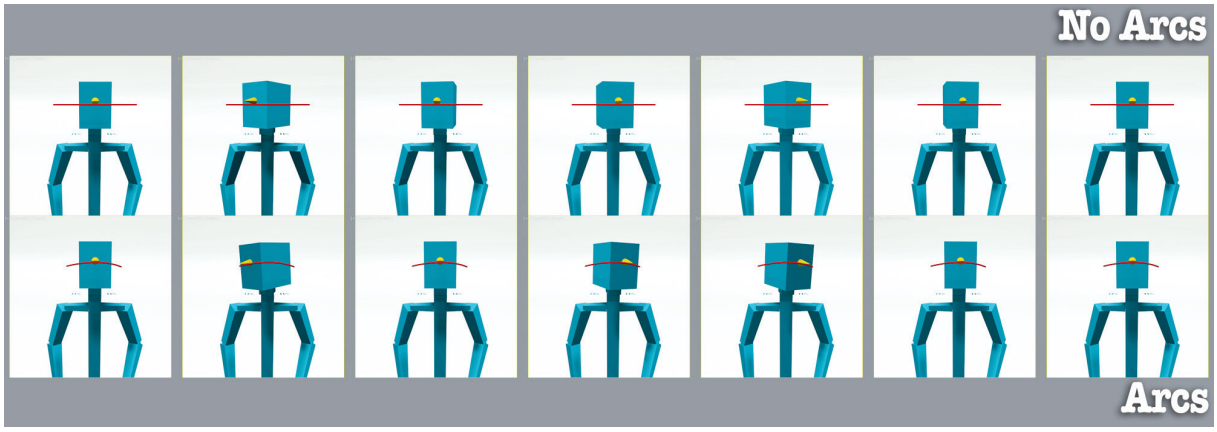


Figure 4.9: An animation sequence denoting the principle of Arcs. The red marks represent the trajectory of the most relevant joints.

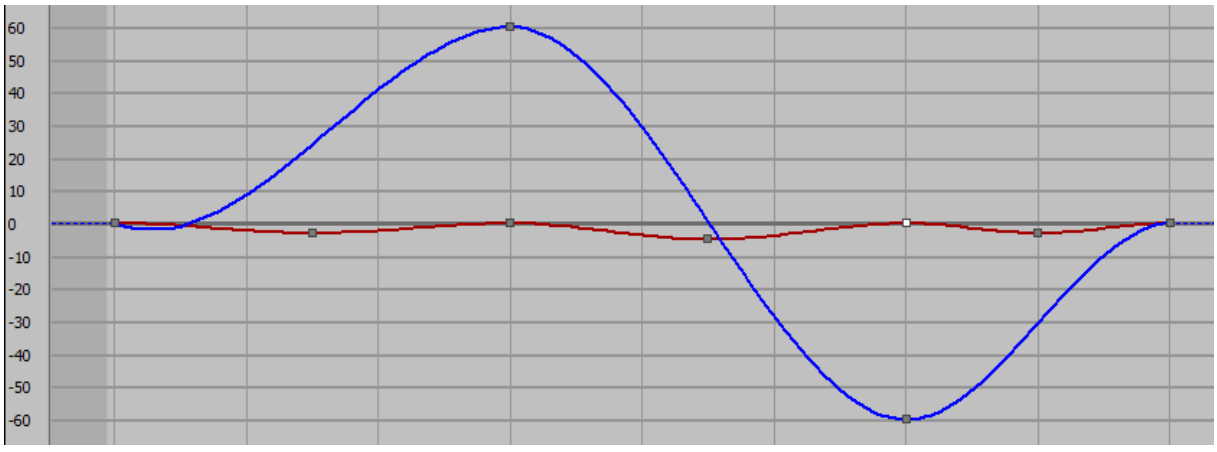


Figure 4.10: Animation curves demonstrating Arcs. The blue curve is the Panning DoF, rotating from the rest pose, to its left (60 degrees) and then to its right (-60 degrees), and then back to rest. During this motion, the Pitch joint (red curve) slightly waves between those key-frames.

movements. It is actually a feature that can be implemented in animation systems by contrasting the motion signal [32].

Figure 4.11 shows not only an amplification of the most relevant features of an animation, but also an added feature - an 'anticipation' backward step. This is meant to show that exaggeration can consist of more then just contrasting the signal, and that by exaggerating the anticipation we can also make the actual action seem more powerful. Because this kind of practice may endanger the robot's surroundings and users if not correctly planned, it is recommended only within pre-animated motion, or for performance and entertainment robots in which the robot's surroundings and mechanical reach are guaranteed to be safe.

Figure 4.12 presents a snapshot from the video[1] illustrating how this principle looks like on the NAO robot, while Figure 4.13 show the same for the EMYS robot.

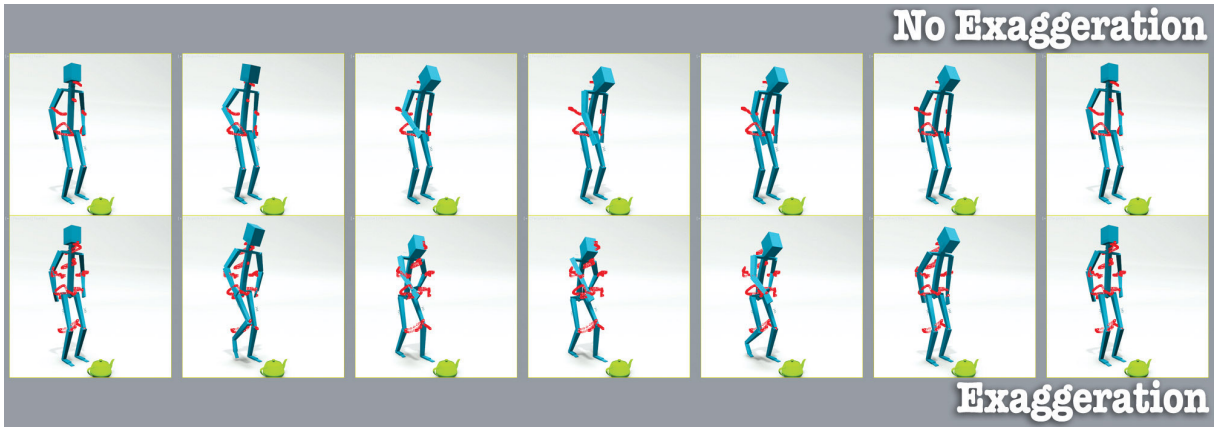


Figure 4.11: An animation sequence denoting the principle of Exaggeration. The red marks represent the trajectory of the most relevant joints.

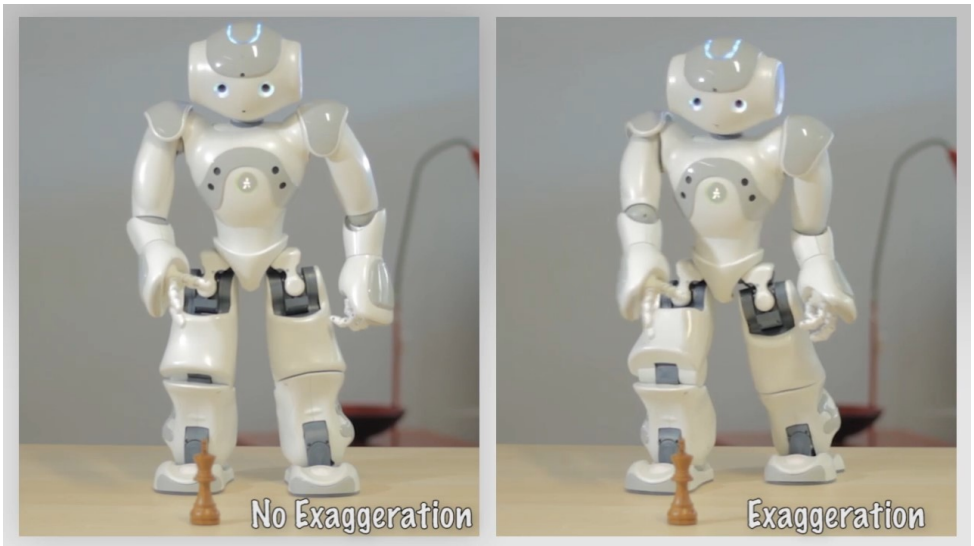


Figure 4.12: The principle of Exaggeration exemplified on the NAO robot.

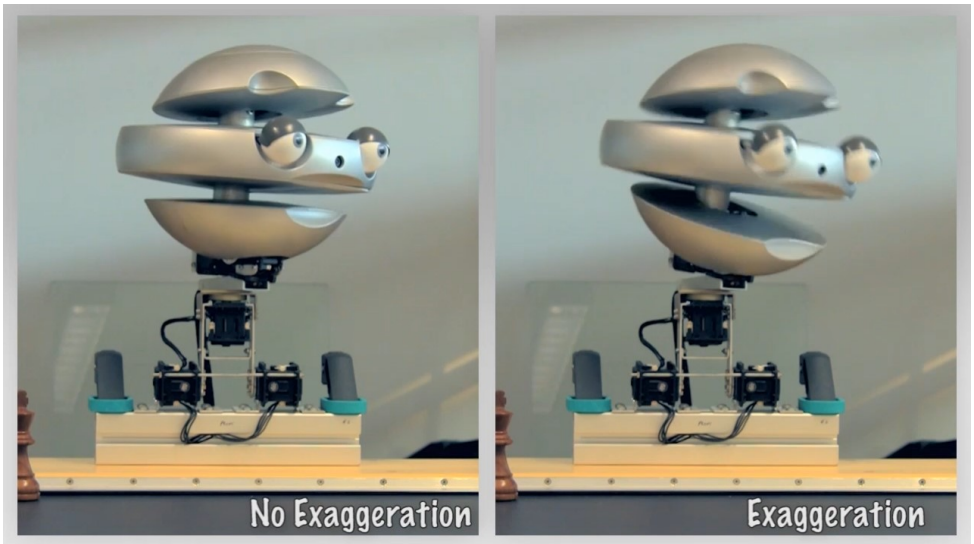


Figure 4.13: The principle of Exaggeration exemplified on the EMYS robot.

4.1.8 Secondary Action and Idle Behavior

During a conversation, people often scratch some part of their bodies, look away or adjust their hair. In Figure 4.14 we can see a character that is crouching to approach the teapot, and in the meanwhile scratches its gluteus. Using secondary action in robots will help to reinforce their personality, and the illusion of their life.

A character should not stand stiff and still, but should contain some kind of Idle motion, also known as *keep-alive*. Idle motion in robots can be implemented in a very simplistic manner. Making them blink their eyes once and a while, or adding a soft, sinusoidal motion to the body to simulate breathing (lat. *anima*) contribute strongly to the illusion of life.

In the case of facial idle behaviour such as eye-blinking, during a dramatic facial expression these will often go unnoticed or may even disrupt the intended emotion. It is better to perform them at the beginning or end of such expressions, rather than during. Similarly, blinking also works better if performed before and between gaze-shifts.

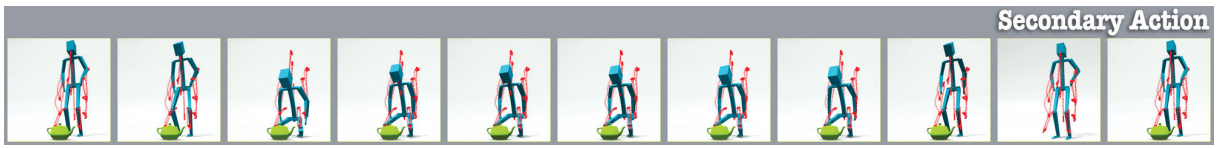


Figure 4.14: An animation sequence denoting the principle of Secondary Action. The red marks represent the trajectory of the most relevant joints.

4.1.9 Asymmetry

This principle was derived from the traditional principle of Solid Drawing. Although the traditional principle seemed not to relate with robots, it actually states some rules to follow on the posing of characters.

It states that a character should neither stand stiff and still, nor does it stand symmetrically. We generally put more weight in one leg than on the other, and shift the weight from one leg to the other. It also suggests the need for the idle behaviour, and how it should be designed.

The concept of asymmetry stands both for movement, for poses and even for facial expression. The only case in which we want symmetry is when we actually want to convey the feeling of stiffness.

Figure 4.15 shows a character portraying another Principle - Idle Behaviour, while also standing asymmetrically. This Idle Behaviour is performed by the simulation of breathing and by slightly waving its arms like if they were mere pendulums.

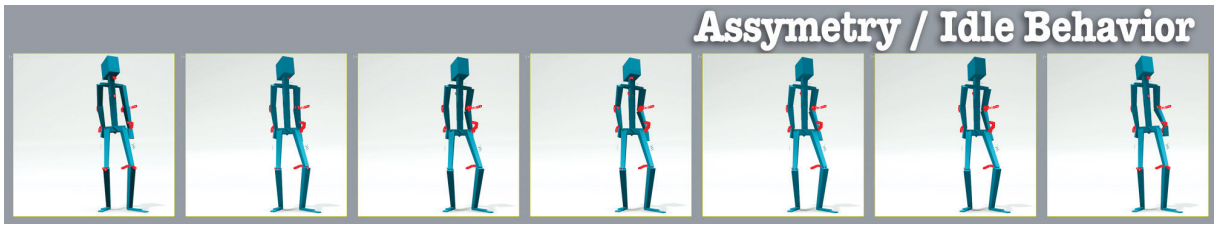


Figure 4.15: An animation sequence denoting the principles of Asymmetry and Idle Behaviour. The red marks represent the trajectory of the most relevant joints.

4.1.10 Expectation

This principle was adapted from the original Appeal. If we want a viewer or user to love a character, then it should be beautiful and gentle. If we are creating an authoritative robot, it should have more dense and stiff movements. Even if one wants to make viewers and users feel pity for a character (such as an anti-hero), then the character's motion and behaviour should generate that feeling, through clumsy and embarrassing behaviours.

Figure 4.16 shows two characters performing the same kind of behaviour, but one of them is performing as a formal character like a butler, while the other is performing as a clumsy character like an anti-hero. In this case the visual appearance of the character was discarded. However, if we had a robotic butler, we would expect him to behave and move formally, and not clumsy.

The expectation of the robot drives a lot of the way users interpret its expression. It relates to making the character understandable, because if users expect the robot to do something that it doesn't (or does something that they are not expecting) they fill fail to understand what they are seeing.

Wistort refers to Appeal as 'Delivering on Expectations' [164], and his arguments have inspired us to agree. He considers that the design and behaviour of a robot should meet, so if it is a robotic dog, then it should bark and wag its tail. But if it is not able to do that, then maybe it should not be a dog. The Pleo robot[2] for example, was designed to be a toy robot for children. So the design of it as a dinosaur works very good, as it does not cause any specific expectation in people - as people do not know any living dinosaurs, and as such, they don't know if Pleo should be able to bark or fetch, so they don't expectation him to be able to do any of that.

4.1.11 Timing

Timing can help the users to perceive the physical world to which the robot belongs. If the movement is too slow, the robot will seem like it is walking on the moon.

However, timing can also be used as an expression of engagement. Some studies have revealed a correlation between acceleration and perceived arousal. A fast motion often suggests that a character is active and engaged on

[2] `www.pleoworld.com` (accessed January 12, 2019)

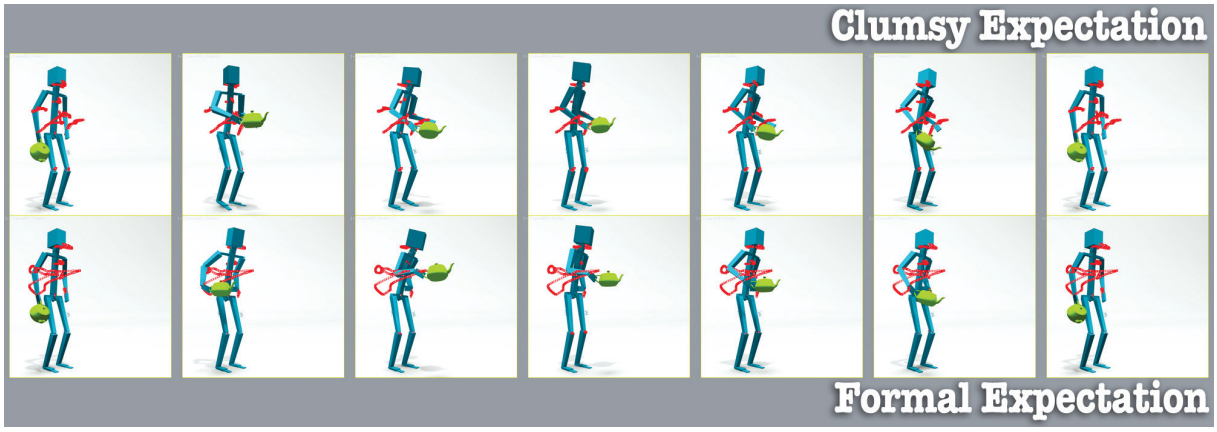


Figure 4.16: An animation sequence denoting the principle of Expectation. The red marks represent the trajectory of the most relevant joints. Notice how the clumsy version balances the teapot around instead of holding it straight, and waves around its left ar instead of holding it closer to its body, delivering a feeling of discourtesy.

what it's doing [95, 31].

Being able to scale the timing is useful to be able to express different things using the same animation, just by making it play slower or faster. In Figure 4.17 we get a sense that the top character is not engaged as much as the lower character, because we see it taking longer to perform the action. It may even feel like the character is bored with the task. In the fast timing case we are showing less frames of the same animation, to give the impression of it being performed faster. In reality, that would be the result, as a faster paced animation would require less frames to be accomplished using a fixed time-step.

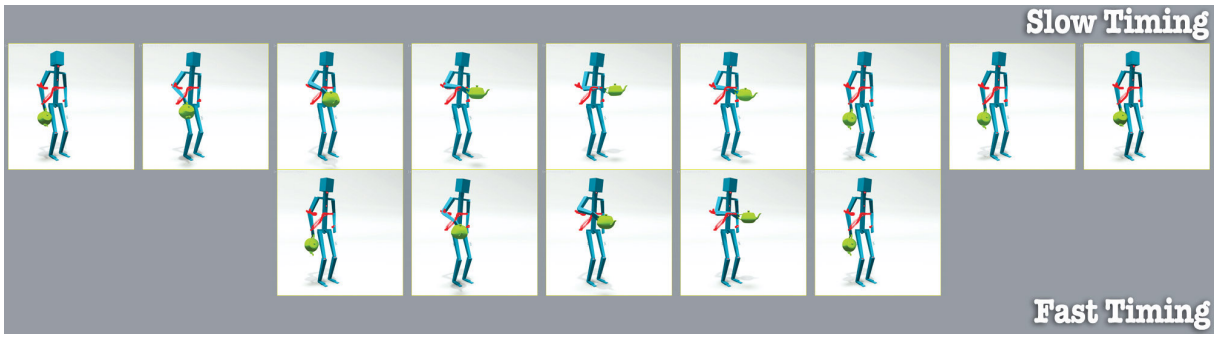


Figure 4.17: An animation sequence denoting the principle of Timing. The red marks represent the trajectory of the most relevant joints.

As a principle of robot animation, timing is something that should be carefully addressed when synthesizing motion e.g. using a motion-planner. Such synthesizer will typically solve for a trajectory that meets certain world-space constraints, while also complying with certain time-domain constraints such as the kinematic limits that the robot is allowed to perform. In many cases, a very conservative policy is chosen, i.e., the planner is typically instructed to move the robot very slowly in order to keep as far away as possible from its kinematic limits. However, such a rule may be adding some level of unwanted expressiveness to the motion. We therefore argue that when using such planners it is important to consider, within the safety boundaries of the robot's kinematic limits, ways of generating trajectories that can exploit the time-domain in a more expressive way.

4.1.12 Follow-Through and Overlapping Action

This principle works like an opposite of anticipation. After an action, there is some kind of reaction - the character should not stop abruptly.

We should start by distinguishing these two concepts here. *Follow-through* animation is generally associated with inertia caused by the character's movement. An example of follow-through is when a character punches another one, and the punching arm doesn't stop immediately, but instead, even after the hit, both body and arm continue to move a bit due to inertia (unless it is punching an 'iron giant'). Overlapping is an indirect reaction caused by the character's action. An example of overlapping is for example the movement of hair and clothes which follow and overlap the movement of the body.

Using follow-through with robots requires some precaution because we do not want the inertial follow-through to hurt a human or damage any other surroundings. Follow-through might also cause a robot to loose balance, so it seems somewhat undesirable. Many robot systems actually will try to defend themselves against the follow-through caused by its own movements, so why would we want it?

In first instance, we consider that follow-through should better not be used in most robots, especially for the first reason we mentioned (human and environment safety). However, when it can be included at a very controlled level, namely on pre-animated motion, it might be useful to help mark the end of an action, and as such, to help distinguish between successive actions. Unlike anticipation, however follow-through is much more likely to be perceived by humans as dangerous, because it can give the impression that the robot slightly lost control over its body and strength. We would therefore imperatively refrain from using it on any application for which the perception of safety is highest, such as in health-care or assistive robotics.

Overlapping animation depends mostly on the robot's embodiment and aesthetics. It might serve as a tip for robot design, by including fur, hair or cloth on some parts of the robot, that can help to emphasize the movement [169]. As such, we find no need to include overlapping animation into the animation process of robots per se, because whatever overlapping parts that the robot might have, should be 'animated' by natural physics. Therefore if one wishes to use it, it should be considered as an animation effect that is drawn by the design of the robot's embodiment, and thus should be developed initially at the robot design stage.

4.2 Dimensions of Kinematronics

We introduce here the concept of *Kinematronics*, which refers to all the high-level mechanical and electronic systems that allow a robot to portray animate expression either kinematically (through physical movement) or electronically (through screens, lights and sounds). The term is derived from *kinema*tics, which would refer only to the mechanical, physical components, and is composed with the concept of "elec*tronics*" to include the non-mechanical forms of expression.

Robots may take many forms and shapes, and provide various means of both interacting with the world, and of conveying expressivity. We start by defining an *expressive Degree of Freedom (DoF) (degree of freedom)*, further referred to merely as a DoF, to be a one dimensional expressive channel that can be individually controlled through a given range or set of values. Each expressive DoF in a robot can be controlled individually during interaction in order to convey a significant and intentional expression. Based on the set and types of DoFs a robot has, and their

individual and aggregate role on providing expressivity, we have defined four different dimensions of kinematronics:

Stationary Expression refers to motion performed by DoFs that are purely mechanical and that do not yield any intentional movement in space. Examples of such expressions are facial and postural expressions. The term *stationary* is chosen because these are mostly found in stationary robots such as desk-top robots, which do not move around by themselves. We do include full-body postures into this type of expression, as long as they are not meant to move the robot in space. An example of that would be a standing humanoid robot, which enacts a full-body emotive posture. While its legs could be used for walking, i.e., for spatial function and expression, when they are used in non-locomotive expressions we do consider them to be acting as part of the stationary expression dimension.

Spatial Expression refers to motion that moves the robot around in space. These are typically accomplished by either wheels or legs, but can also be performed by rotors, as in a quadcopter drone. Note that besides walking, a legged robot, for example, may also have the ability to perform controlled movements in height, allowing it to e.g. climb up stairs, jump, or crouch. Comparing to the the stationary expression dimension, the accounted number of DoFs may seem lees intuitive than for the other dimensions, as it does not related to the number of legs or wheels that the robot has, nor to their articular structure. A legged or wheeled robot can move in 1D, 2D or 3D, while also being able to perform motion that rotates about a given set of axes. For example, a 1D-capable robot could be able to move either back and forth, or side to side. A 2D-capable robot could move back-and-forth, and additionally either rotate left and right (yaw), or strafe to the sides, or travel up and down. A 3D-capable robot can typically move in 2D plus rotate about the vertical axis (Yaw). This dimension therefore accounts for the total number of axes about which the robot can perform spatial movement, be them translational or rotational axes. It can thus account for zero to 6 DoFs, given the robot's ability to travel along its local X, Y or Z axes, or to perform yaw, pitch or roll movements. Please refer to figure 4.18 for any further clarifications.

Display Expression refers to expressions portrayed through some form of electronic light display. This can include simple monochrome LEDs, multi-color/RGB LEDs, a monochrome (LCD) screen or an RGB screen. Ultimately it can also include some kind of light projection system. If no layer of expressive control is defined for the display, we consider each individually controlled LED to be one DoF (even if it is multi-colored), and each individual screen/projector to also be one DoF (regardless of its pixel resolution). However, if the LEDs are disposed in a particular expressive way, such that they all relate to the same expressive channel, that should always be controlled as a whole (e.g. each of NAO's eyes is composed of 8 LEDs), then we consider them all to be a single, aggregated DoF. Similarly, of a robot necessarily includes a particular type of expressive display application, such as a face, then we consider the display element to have as many DoFs as that application. Note that in a case where e.g. the application allows to individually control the opening of each eye, that would amount to 2 DoFs. If one can control the opening and frowning of each eye, then it has 4 DoFs. If however the application has only a set of pre-defined expression, without any further control, then we consider it to have only one DoF, which corresponds to the discrete list of expressions. This type of specification for display expressions allows us to abstract from the technical aspect of how the displays and lights are physically implemented, and instead specify the type and amount of expressive signals can be

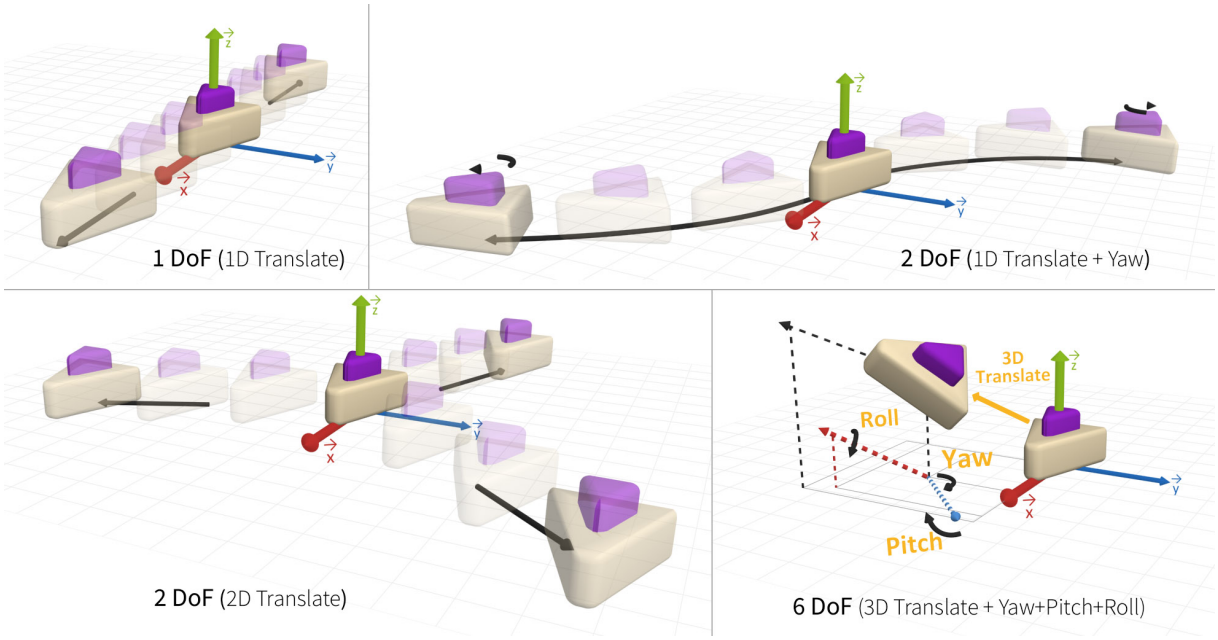


Figure 4.18: The Spatial Expression of Kinematronics. ***top-left***: movement in a single direction represents 1 DoF. ***top-right***: movement in one translational direction plus one rotational direction (in this case, Yaw), amounts to 2 DoFs. ***bottom-left***: movement in two translational directions also amount to 2 DoFs. ***bottom-right***: the most complex 6-DoF example, in which translational movement can be performed in 3D, and rotational movement can also be performed along the three rotation axes. Using intrinsic rotations is recommended, i.e., the coordinate system for the rotations is attached to the moving body and therefore changes after each elemental rotation. Elemental intrinsic rotations are performed in the order Yaw-Pitch-Roll.

individually portrayed through it.

Audible Expression refers to any audible form of intentional expressivity that a robot may have, from simple beeps, to 4 or 8-bit audio effects (sampled or generated), or a more sophisticated speech system. Speech may either be pre-recorded (from humans), pre-synthesized, or synthesized during interaction using a TTS. Outputting speech will typically require a more modern 16- or even 24-bit audio output system. Similarly to the case of the display expression level, we consider that each individual audio player/controller accounts to one DoF. That means that the speech system is one DoF, and any other audio-output adds as many DoFs as the number of audio signals it can control and play simultaneously.

In Figure 4.19 we can see examples of how some several robots would be placed within the kinematronics dimensions. In particular, taking the humanoid *NAO H25*[3] robot as example, we see it contains at least 25 stationary DoFs that can be used for expression. Although its legs may be used for locomotion, there are many cases in which they are used purely for expressive postures. Regarding spacial expression, NAO is capable of 3D motion, given that it can walk forward and backward, strafe sideways, and also perform yaw rotation. As to display expression, and while in total, the robot has many individual LEDs, we consider the amount of display-expression DoFs to be 5: one for each eye and ear and one on the head. Finally, for audible expression, NAO is capable of both talking and playing audio files. While it physically contains two speakers (one on each ear), what matters expressively is that its audio-player typically allows to play only one file/sound at a time. Be it music, expressive or warning sounds, they will all be played through the same controller. Therefore, it contains 2 audible-expression DoFs: the TTS, and the

[3] `http://doc.aldebaran.com/2-5/family/nao_h25/index_h25.html` (accessed January 12, 2019)

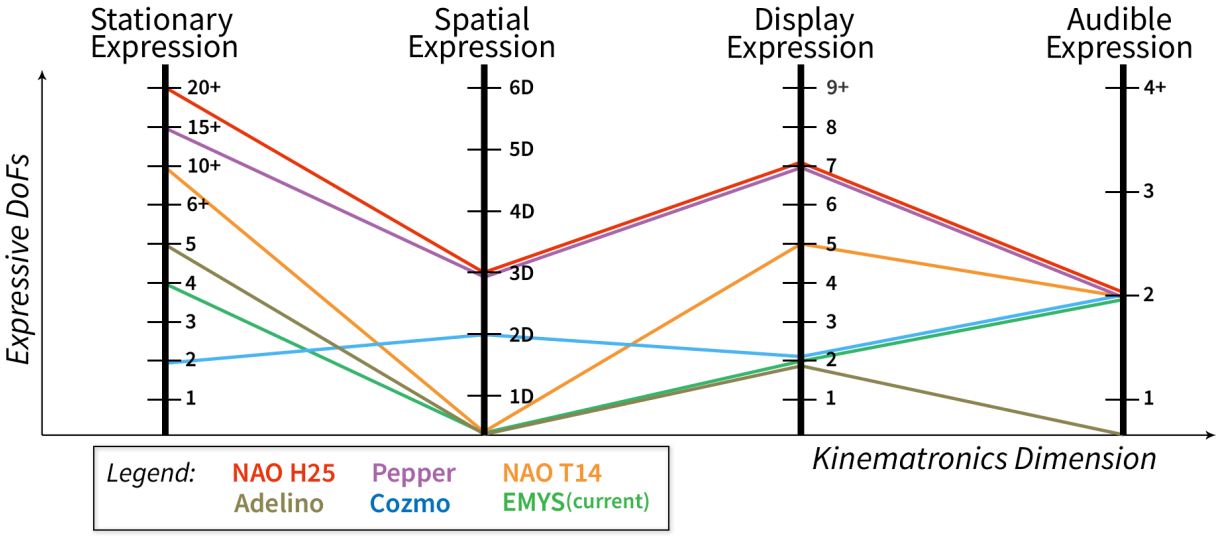


Figure 4.19: The four kinematronics dimensions, along with an illustration of how several existing robots would be represented.

audio-player. The figure also compares the NAO T14[4], Pepper[5], Adelino[6], Cozmo[7], and the latest version of the EMYS[8] robot.

It is extremely important to note that these dimensions do not portray how expressive a robot is or can be. Due to design factors, a robot with e.g. few static expression DoFs such as the EMYS or the Adelino, may be considered more expressive than a high-DoF robot such as the NAO. The purpose of these dimensions are solely to enumerate and provide a specification for the various expressive channels that can be found in robots, and does not provide any hints for comparing the overall expressiveness between them.

4.3 The *Nutty Workflow* for Robot Animation

In order to implement social robots that are based on the concept and processes of robot animation, one must properly introduce these into the design and development workflow. In this section we introduce general concepts on how a system architecture should be laid out and used, which is presented as the *workflow* for the design and development process. The workflow presented here are deeply inspired on the work developed previously with the Nutty Tracks animation engine (Section 6.1), which was used as a sandbox to explore and develop new robot animation techniques for interactive applications [121, 42, 157]. As such, we refer to these as the *Nutty Workflow* and the *Nutty Pipeline*.

The Nutty Workflow presented here aims specifically at allowing the type of animation capabilities mentioned throughout this thesis. As such, it should not stand as a general workflows for the whole field of HRI and social robotics. Instead, it presents the elements that should (or are suggested to) be present to achieve highly animate social robots, that exhibit the illusion of life, and whose design and development was carried out with animation theories and practices in mind. Further modifications should be carried out in order to accommodate any other

[4] `http://doc.aldebaran.com/2-1/family/nao_t14/index_t14.html` (accessed January 12, 2019)
[5] `http://doc.aldebaran.com/2-5/home_pepper.html` (accessed January 12, 2019)
[6] `https://vimeo.com/232300140` (accessed January 12, 2019)
[7] `https://www.anki.com/en-us/cozmo` (accessed January 12, 2019)
[8] `https://emys.co/` (accessed January 12, 2019)

requirements.

4.3.1 Concept Design

First, if developing a new robot, its concept design must carefully consider all the expressive capabilities and the kinematronic dimensions needed. We will not deeply explore this concept there, as it is also subject of study in other works. In particular we refer to the work by Hoffman & Ju which explores the initial stage of designing a robot with its expressive movement in mind [96]. This stage should include both hand-drawn concepts, along with 3D animated concepts, and even pre-visualization prototypes that allow the designers and developers to virtually simulate how the robot would behave during specific use cases of interaction with humans. Such pre-vizualization can be developed using game-development engines such as the Unity[9] or the Unreal Engine[10]. This initial concept stage will help to inform developers both about the aesthetical design of the robot, on its kinematic structure, such as number of joints, and range of motion, and also about the use of display expression elements. Sound design [170] can also be explored at this stage, and can be used both on rendered 3D animations of the robots, and on the interactive pre-vizualization. If the robot will be performing speech, it is also a good idea to pre-visualize how it will look with the robot at this stage, as that may seriously impact the design of any facial features that should become animated while the robot is speaking.

4.3.2 The Nutty Workflow

Figure 4.20 illustrates the *Nutty Workflow*. We split the workflow in two main areas: the creative development and the technical development. The idea is that both are intrinsically part of the model and should holistically be considered as a whole. In the creative development area, we find most of the behaviour-authoring related to the

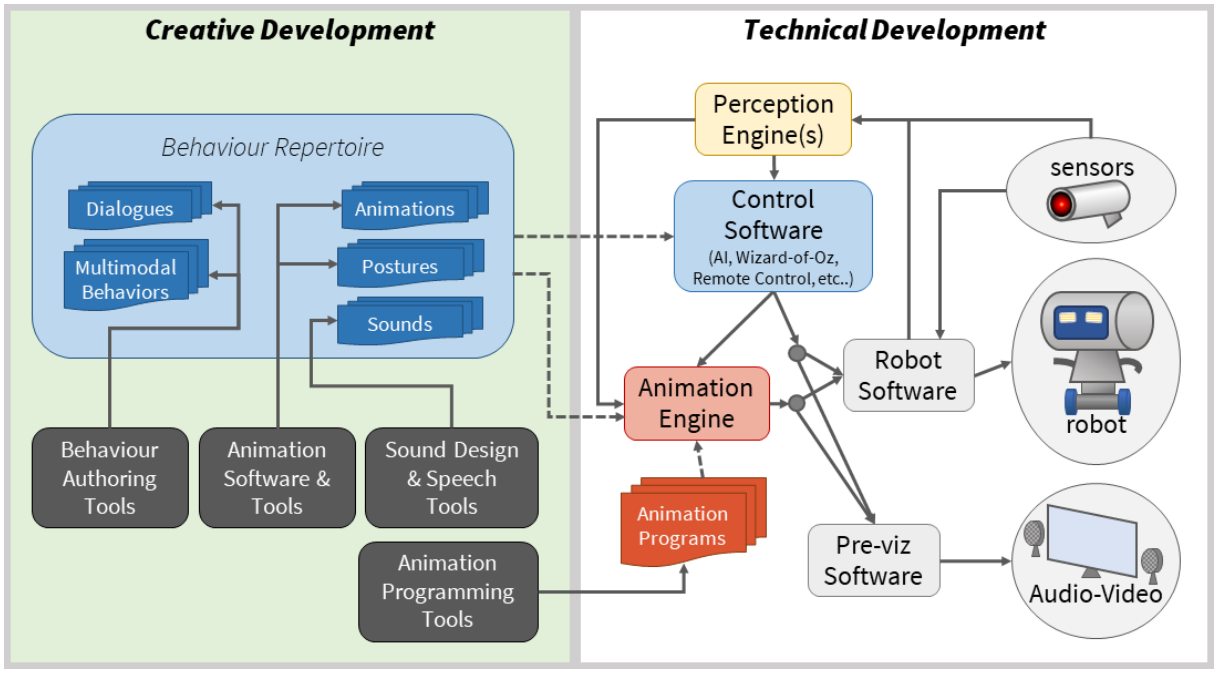


Figure 4.20: The *Nutty* robot animation workflow.

[9] `https://unity.com` (accessed January 12, 2019)
[10] `https://unrealengine.com` (accessed January 12, 2019)

social robot, including the development of pre-designed animations and postures, sound design, dialogues and composite multi-modal behaviours, which allow to sequence and synchronously play back a set of e.g. animations and postures, along with dialogue and sound effects. This area is expected to include non-technical developers such as animators or psychologists. As such it is important to carefully consider the type of tools used, to make sure they can produce properly specified assets that can be used further in the software. We will addressed and elaborate on such tools later in section 5.3.

The technical development area consists of the software architecture that is typically expected for a social robot. That includes, on the hardware part, both the **Robot** and **Sensors**, along with the lower-level **Robot Software** that controls and communicates with both. Note that while the robot will likely contain sensors already, other external sensors may be used, such as external cameras (RGB or RGB-D) for object and user tracking and recognition, or even for localization of the robot. As such it is useful to include a dedicated **Perception Engine** that can handle the input signals and translate them to symbolic, meaningful inputs for the Control Software and the Animation Engine. The **Control Software** is illustrated as a single component, but may be split into various sub-components depending on the application. This should handle the actual application-domain knowledge and control, which allows the robot to perform a given task or application. Alternatively, it can consist of remote control tools, such as a Wizard-of-Oz, or a tele-operation panel. The output of Control Software should be discrete and well-specified commands, given to either the robot software directly (e.g. shutdown, reset, etc..), or to the Animation Engine.

Because the focus of this workflow is robot animation, we do place the **Animation Engine** as a separate component. This engine should be able to handle all the commands that control the various kinematronic abilities of the robot. Depending on the aim of the application, it may have various levels of complexity. There are explained further in Section 5.2.

Note that the animation engine has the ability of running **Animation Programs**. These programs differ from a static animation file, in that they contain a sequence of rules that allow the generation, transitioning and blending of various expressive modalities, along with the computation of ad-hoc motion such as the ones that are produced through inverse kinematics or path planning (Section 5.2). While a more traditional architecture would delegate such techniques to the actual robot software, we claim that including all motion control in the animation engine allows to seamlessly use **Pre-Viz Software** in place of the real robot during much of the development. In particular, such Pre-Viz aims at allowing the creative developers to work on the robot's behaviour and expressivity, in an interactive way, in order to ensure that the final behaviour of the robot during an interaction will match the intended, authored behaviour, as close as possible.

Chapter 5

Robot Animation in Practice

5.1 Building Autonomous Socially Expressive Robots using SERA

Throughout our history of creating various human-robot interaction (HRI) applications, for different purposes and featuring different robots, we have designed and developed multiple interaction scenarios and software tools to aid us in augmenting both the quality of the interaction and richness of expressivity of the different robots used. Figure 5.1 provides a high-level illustration of how most of these scenarios have been built.

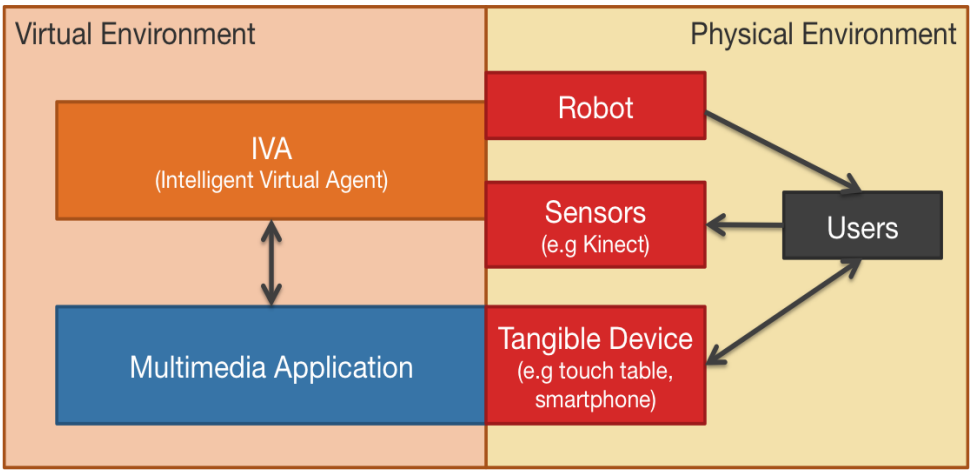


Figure 5.1: The composition of our typical HRI scenarios.

In order to streamline and promote the design and development of reusable components, we have created SERA, which is an architecture and set of tools for creating autonomous socially expressive robots (ASERs) [42]. The SERA ecosystem was created following on the SAIBA model which is very popular within the virtual agents community [37]. Figure 5.2 shows the original SAIBA model, while Figure 5.3 illustrates the general components of the SERA model. Colouring of the components establishes a relationship between both figures. In overall, our architecture aims at providing a reusable structure and collection of modules, that can work for different scenario applications and robots.

The **Intention Planning/Decision Making** (DM) layer contains components that perform the decision making

Figure 5.2: The SAIBA model for virtual agents [37].

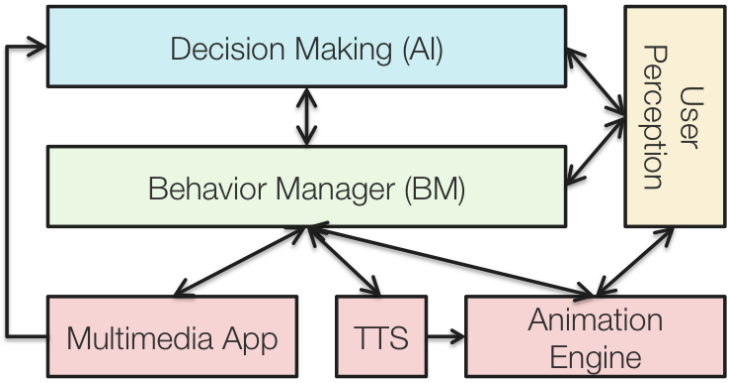


Figure 5.3: The SERA model. In compliance with SAIBA, the AI is the Intention Planning level; BM is the Behavior Planning level; All the others (including User Perception) are the Realization level.

and is mostly scenario-specific, as it models the high-level knowledge intrinsic to the interaction scenario or application.

The **Behaviour Planning/Manager** (BM) layer builds high-level behaviours based on intention-directed instructions generated at the higher level. These behaviours can be slightly generalized, as different characters and scenarios may share some common behaviour mechanisms.

We generally decompose the realization level into a Text-to-Speech (TTS) engine, realization of animation, and some multimedia application through which the user can interact with the system and receive feedback from it.

User Perception has not been traditionally included in the SAIBA model. However, on previous work adapting that model to HRI, we have include a transversal Perception layer that runs across all other levels [171], providing both high and low-level representations of the interaction environment. Our experience has shown it to be useful both to provide high-level perception of the user such as facial recognition, emotional state recognition or gestural actions, to the AI, and also lower-level perceptions such as simple face detection or sound direction estimation, to be used both for the generation of some types of behaviours at the DM and BM components (e.g. rapport), and for adaptation of behaviour at both the BM and Animation components (e.g., tracking a user's face). Actions selected by the user in the multimedia application can also be interpreted by the AI as perception of user actions (e.g. clicked a button, selected an option).

In particular, when dealing with robots, a component such as the *Animation Engine* tends to be developed specifically for the robots' motor control systems, as these differ greatly from robot to robot. This specificity of the realization layer for robots thus poses as a problem in abstracting high-level behaviours in a way that scenario- and behaviour- components can be used across various robotic platforms and interactive applications. We addressed this problem by creating the Nutty Tracks animation engine, further described in Section 6.1.

We will further refer to a SERA Character, or just a Character as a set of components that have been built and are architecturally laid based on SERA, acting together in order to function as an ASER. Although a Character could also be virtual, in the scope of this thesis we will not enforce such distinction unless it is required.

It is important at this point to discuss how the Autonomous feature of the robot is given especially by the components used for Decision Making. Throughout our development of full scenarios or Characters, it has been a trend to follow a two- or three-stage iterative user-centred design. While the final result is an autonomous robot, throughout its development, we may go through several milestone prototypes that are typically semi-autonomous and partially tele-operated.

The following sections of this chapter start by presenting our ASER development methodology and then some of the main tools that have been developed as part of SERA. Thalamus, presented in Section 5.1.2 is a component integration framework which we have been using to allow all the modules and tools to easily connect and communicate across multiple computers and operating systems. Skene (Section 5.1.3 is a Behaviour Manager that connects with most of the other modules in a character. Nutty Tracks, in Section 6.1 is a symbolic animation system that allows the ASER to mix all the different behaviour modalities generated during an interaction, and can produce them for different embodiments and robots.

5.1.1 The SERA Development Methodology

Our methodology for developing autonomous socially expressive robots is generally composed of two or three states. Figure 5.4 illustrates the complete three-stage methodology. Depending on the nature of the scenario and its application, we can optionally skip some of the stages. Given, for example, an HRI application that is directed to school children or elders, it is important to consider all the stages, and to include the experts (e.g. teachers, caregivers) in the process. However for a typical HRI scenario aimed at entertainment, if would generally be sufficient to start developing at Stage 2. Further simple, small-scale applications such as the student projects develop during classes of their Masters degree courses. would most likely consider solely Stage 3. In such cases one must also consider a different designation for some of the steps in Stage 3 (e.g. Refinement becomes actual Development).

Stage 1 - Mock-up Prototype

The first stage is generally considered when the application requires us to work alongside experts. In that case we start by establishing the foundations of the interactive scenario and activity, as in what the target users will get from it.

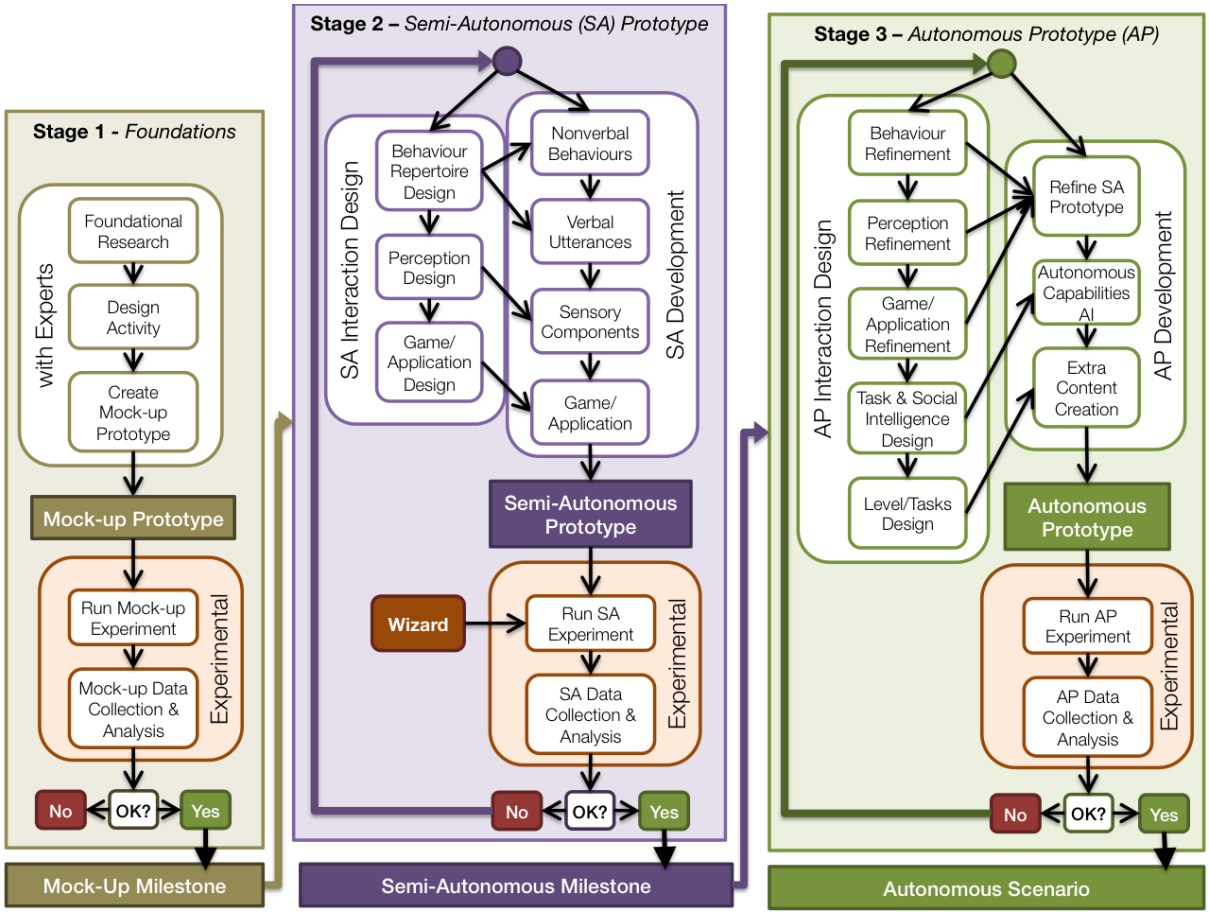


Figure 5.4: The SERA-based multi-stage ASER development methodology that we have typically followed on previous HRI scenarios.

In order to understand how humans typically address a given task or application, we can build a mock-up prototype and use it to run a human-human (H-H) experiment. This mock-up prototype should look as close as possible to the task or application that we are addressing, but without including any robot at all. For example, if developing a robotic tutor for class rooms that is able to play a given game with children using a tablet or touch-table, we may start by setting up such application, or application prototype on a table, and have children interact with it accompanied by a peer or a teacher. That way we can analyse the H-H interaction that happens while the student is playing the game, and collect the type of utterances and specifications for a gaze model that is appropriate to the target application.

Stage 2 - Semi-Autonomous Prototype

The prototype developed in Stage 1 along with the data collected is used to inform Stage 2. We split this stage between Interaction Design (ID), and Development, where ID can be performed by non-technical professionals such as psychologists, animators or designers.

In this stage we develop a first set of behaviours for the robot, both verbal and non-verbal, through which it should be able to interact with the target users in the target application. When the first Stage was not required, the behaviours are developed either based on literature, on previous analysis or observation, or in some cases (e.g. entertainment applications), they can be developed intuitively or in an artistic fashion. This is also where we develop

most of the required perceptual components, and a nearly final version of the game or application that can also be controlled by an external AI agent (and not only through direct user input). It is important to note that, when using a robot to interact with users through an application running on a tablet or any other virtual device, those virtual applications may have to be modified or developed in order to allow the AI agent to control it, e.g. by adding commands that allow a button to be pressed or a message to be shown without the user having to interact directly with the application. In many situations the robot may also take an active role in the activity, and as such, it should be able to initiate it, terminate it, or perform actions on it just as if it was a real person interacting. This type of remote control of the application must be carefully considered and designed so that the users understand that it is the robot who is controlling the application. If the robot has hands, having it point at the application in synchronization with the actions can help to convey this. In any case there should always be a visual highlight on the screen whenever the robot is controlling the application or talking about it, so that the user knows where to look at.

This set of behaviours, perceptual capabilities and application are then evaluated using a remotely controlled semi-autonomous prototype of the robot. Although this is typically referred to solely as a Wizard-of-Oz (WoZ) setting [172], we stress the semi-autonomous factor of our WoZ practices. In a semi-autonomous WoZ, the Wizard is used mostly to replace a higher-level Perception and Decision Making components. However the robot can run most of the task and interaction autonomously, while the Wizard is provided with higher-level controls that guide the behaviour selection or generation processes, based on what is observed by the Wizard throughout the interaction. Performing an evaluation at this stage allows us to test and refine the robot's verbal and non-verbal behaviour, and its role within the interactive activity, before investing on making the system run autonomously.

At the end of this stage, the HRI system should be almost fully develop, in a way that we could just remove the WoZ module and replace it by the final Perception and AI modules. In order for that final step to be seamless, it is important to integrate the WoZ module just as if it was another component if the system. Therefore, if some kind of messaging mechanism is used to communicate between modules, the WoZ module should already produce and receive the same messages that the final Perception and AI modules are expected to produce, using the same API, so that the final stage will require very few to no changes in its behaviour and code.

Stage 3 - Autonomous Prototype

This stage is about turning the previous prototype into an autonomously controlled one. At this point there should be only some changes left to do regarding the robot's verbal and non-verbal behaviour, and the interactive application. In order to guarantee a proper interaction flow with the autonomous version, it is however expectable that the design and use of some of these components may need to suffer some changes. It is also common to add more content to the prototype at this stage, such as extra levels in a game, given that all the mechanisms required for the robot to interact through it are already created and tested.

Developing the final Perception and AI modules is generally application-specific. Even when using the same Perception sensors such as the Kinect or microphones, different applications may require different higher-level perceptions of a task or of the user. As to the AI, depending on the goal and scope of the project, is may be created as a simple rule-based system, or even include some for of machine learning.

An evolved version of the semi-autonomous WoZ, called *restricted-perception WoZ*, has been presented by Sequeira et al., in which the Wizard monitored the interaction not through a video or audio feed, but through

discrete symbolic messages that were generated by the Character's actual perception modules [173]. Instead of being presented with a live audio and video feed, the Wizard is presented with the discrete information that can actually be collected through the sensors and the Perception module. An example of this is when the robot must react the the user's gazing direction. Because its Perception will have to identify whether the user is gazing, e.g., towards the robot, towards the task, or elsewhere, the Wizard should be presented only with this fact, given what its sensors were able to perceive. This way the Wizard was presented with the realistically limited and imperfect information that can be collected through the Character's sensors, and was forced to perform decisions based on those, instead of being allowed to perform based on a more natural observation from real-time audio and video, through which the Wizard is able to discern the user and the environment in a way that the robot is not.

The main advantage of this type of setting is making it easier to create autonomous behaviours, based on the data collected from the interactions of the previous Stage. If the Wizard's selected actions are recorded based on the discrete information that is produced by the real robot's Perception, then it is possible to train the AI directly using that data.

In this stage it is still crucial to have the Interaction Designers working closely with the Developers, in order to guarantee that the expected interaction is achievable in an autonomous fashion, while finding solutions for limitations that do not sabotage the interaction.

5.1.2 Thalamus

Thalamus is a high-level integration framework aimed especially at developing interactive characters that interact both through virtual and physical components. It was developed in C#/.NET to accommodate social robots into such a framework, while remaining generic and flexible enough to also include virtual components such as multimedia applications or video games running on a touch table [41]. It follows on the concepts of asynchronous messaging middle-ware and on well-defined message structures (based on MOM as ROS does) to provide a seamless plug-and-play-modules functionality (Figure 5.5). However, being a higher level middle-ware (in comparison to ROS) it works "out of the box", without requiring any installation on the host system, and also includes graphical interfaces aimed at developing Thalamus modules as agents. It also aims at being easy to use and to share in an academic and research setting, to be portable, and adequate for collaborative development.

Thalamus breaks the sense-think-act loop by not specifying any particular layer structure. The idea behind it is that a Thalamus Character is an agent built out of agents. These agents are Thalamus modules that exchange perceptions and actions between them, so while any module may actually contain a sense-think-act loop, holistically the Thalamus Character does not. That allows it to simultaneously contain several modules that deal with behavior, or with perception, or even with decision making, as long as the combination of them all produces the expected overall behavior. These Characters can be used seamlessly across embodiments (virtual or robotic) and applications, by just switching or tweaking some of the modules. An example of that is a robot interacting with users through an application running on a touch-table, and using a Microsoft Kinect to track the user's face. Contrary to traditional agents that contain a "body", all the those three components represent the physical interface between the users and the system. User perception is informed by the Kinect, which is independent of both the robot and the touch-table; user actions are perceived by the application (e.g user clicks), and behaviors are both executed expressively by the robot, and task-wise through the application (e.g. the agent can invoke the application to pop-up a screen while the

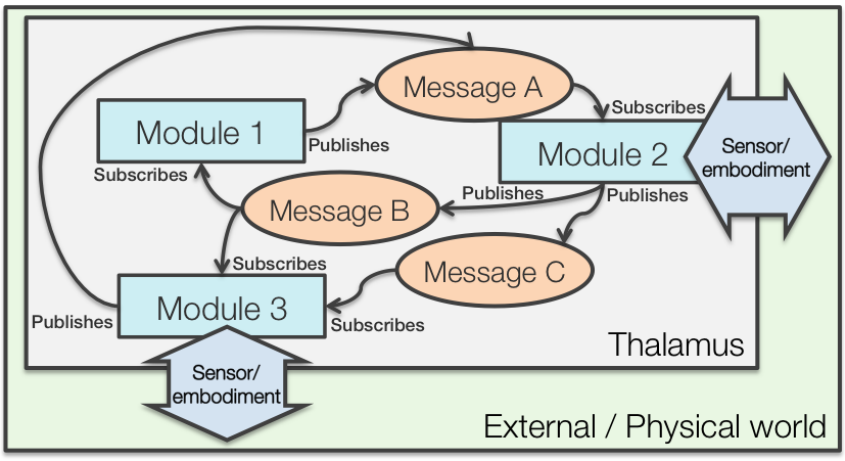


Figure 5.5: Example of how several Thalamus modules coexisting in the same virtual space, exchanging messages through a publish/subscribe mechanism.

robot points at it).

It is important to mention at this point that every component we develop for our system (including the ones on the following sections) is developed in order to function as a Thalamus module. Therefore, our current implementations of SERA Characters are in fact Thalamus Characters, while what we call the SERA ecosystem is the whole set of modules that have been developed and can integrate and coexists within Thalamus, in order to compose specific Characters.

5.1.3 Skene

Skene is a semi-autonomous behavior planner that translates high-level intentions originated at the decision-making level into a schedule of atomic behavior actions (e.g. speech, gazing, gesture) to be performed by the lower levels [171]. It was initially created specifically for the EMOTE project (see Section 7.1.1), with situated robots in mind, that can also interact through multimedia/virtual interfaces (like a large touch-table). As such, it later became a common use on other scenarios besides EMOTE, as the place where most of the other components meet in order to integrate behaviour with the environment. Some of its features are:

- Contain an explicit representation of the virtual and physical environment, by managing coordinates of relevant targets at which a robot can point or gaze at;
- Autonomously perform contingent gazing behavior, such as gaze-aversion and establishing gaze (the opposite of aversion), using an internal gaze-state machine (GSM);
- Gaze-tracking a target marked as a *Person* using the GSM;
- Automatically gaze-track screen-clicks using the GSM (for multimedia application running on touch-tables);
- Maintaining, managing and allowing other components to control utterance libraries;
- Augment the sense of intelligence of the robot, by performing simple back-channelling, and turn-taking with human users.

Regarding the GSM, Skene contains a list of Targets that can be either built-in, loaded from a file, or created in run-time. Target that are not built-in can also be updated in run-time. Each of them are indexed by a key-word, and return a pair of angles representing the horizontal and vertical direction of such target in relation to the robot's embodiment.

A target can either be specified directly through angles, or through screen coordinates (X, Y), or be a procedural, built-in target. In the later case, Skene contains code that generates the coordinates for the given Target (e.g. *Random* target generates random coordinates). Screen coordinates are converted to angles using the Physical Space Manager (PSM), where we can define the position and orientation of the embodiment, relatively to the interactive screen. Therefore, given an (X,Y) point on the screen, it is able to calculate the direction to which the robot should direct its gaze, in order to gaze at that specified points. This is extremely useful and important when the interaction happens around a large touch screen, and the robot is explaining or referring to a particular object or region drawn on it. Some targets can also be set as aliases of other targets. Creating a target called *Tiago*, linked to a built-in target *Person* will allow *Tiago* to inherit all the built-in semi-autonomous behaviours that are associated with *Person*.

The GSM can be controlled using two types of gazing behaviours: Gaze and Glance. The difference between both is that whenever it is instructed to Gaze towards a specified target, that target becomes the new state of the GSM, whereas a Glance will turn the robot's gaze direction towards the new target only temporarily (2 seconds) and will then return to the current gaze state. Note that in any case, the GSM generates lower-level *Gaze* actions which are published to be received by the animation engine. Because the nomenclature is the same, it might lead to some confusion. However internally these represent different actions, as the Gaze action that controls the GSM takes a single string-type parameter (indicating the new gaze target), while the *Gaze* action produced by the GSM contains a floating-point pair that represents a physical direction to gaze at. At the API level the former is actually called GazeAt. However within utterances, we simplify by using solely Gaze, which do refer to the GSM control actions, as described below.

Skene Utterances are the actual representations of the aforementioned intentions and were mostly inspired by the FML-BML pair used in virtual agents and the SAIBA model [37]. They are composed of text, representing what the robot is to say, along with markups both for the TTS, and for behavior execution. The behavior markup can be used to control Gazing, Glancing, Pointing, Waving, Animating, Sound, Head-Nodding and even Application instructions. The following is an example of a Skene Utterance:

```
<GAZE(/currentPlayerRole/)>I'm unsure if it's a good idea to
<HEADNODNEGATIVE(2)> build industries near <WAVE(throughMap)>
the populated areas. <ANIMATE(gestureDichotomicLeft)>
<GLANCE(Eco)> What do you think? <GAZE(clicks)>
```

The behaviors contained in the markup are non-blocking, meaning that while the speech is executed, the TTS engine sends events whenever it reaches a marked-up position, so that Skene can concurrently launch the execution of that mark-up behavior. While this seems like a pliable solution, it actually allows the further Realization components to perform their own resource management. Thus, if for example, the robot needs to gaze somewhere and perform an animation at the same time, the animation engine is the one to either inhibit or blend the simultaneous forms of expression.

We also include replaceable tags in the utterances, so that these may be used as templates, and completed in run-time. Tags are specified by enclosing a word with special characters (which may be custom-changed). Therefore whenever an utterance is invoked to be performed, the invoker must also provide a *Tags Table* that indexes each tag's replaceable word with its current *value* (to be replaced with). In the utterance shown above, */currentPlayerRole/* is a replaceable tag. These can either be used within the spoken text (e.g. to indicate a user's name or score), or within other commands, such as the Gaze command. This allows the utterance to perform behaviours that are only fully specified at runtime, so that the authoring process is less cumbersome. By including commands such as <*Gaze(/currentPlayer/)*>, an utterance can be used at any time, and will include a gazing behaviour that depends on the current state of the task.

The Skene utterances we have used were developed mostly by well informed psychologists that take part in the development cycle as interaction designers. In order to facilitate such collaboration, Skene Utterance Libraries are stored and loaded directly as Microsoft Excel Open XML spreadsheets[1]. Such feature hugely facilitates the interaction designers to collaborate between them and with the technical development team by authoring such files using online collaborative tools such as Google Spreadsheets[2].

Most of what we consider to be semi-autonomous behaviours of the robot are triggered and managed by Skene. These behaviours, described below, were built into this component as they have shown to be useful across different scenarios.

Gaze-Tracking If the gaze target in Skene it set to a target of type *Person*, it will generate *Gaze* commands towards the tracked persons' direction, every time the coordinates of the Person are updated. These coordinates should be updated by an external Perception module that can detect the person, and publish a specific message that contains the coordinates of the person's head.

Gaze-breaking Whenever the gaze target is set to a *Person*, Skene will generate a short Glance to *Elsewhere* from time to time, in order to reduce the sense of fixation. *Elsewhere* is a built-in target that generates random directions, but only upwards.

Politeness is a feature that acts as a very simplistic turn-taking mechanism. If there are microphones connected to some Perception module that can detect whether or not a person is speaking, this feature hold Skene from triggering new utterances while a person is perceived to be speaking. This way people can engage in conversation with the robot, and even if an AI triggers an utterance in the middle of a person's sentence, Skene will hold it until the person finishes.

Questions were considered by allowing any utterance to be marked by the authors as being a question. In that case it is expected that after the robot performs it, the users might answer back. Therefore, after performing a question, the robots waits for a certain amount of question-wait seconds (e.g. 7 seconds) before performing the next utterance, even if it had been invoked immediately. This feature is complemented by the Politeness feature, as after those question-wait seconds, if the person is still responding to tue question, the robot will keep waiting until the person is finished. Whenever during the mentioned question-wait period, a person is detected to start speaking, Skene assumes that the person is replying to the question. Therefore,

[1]XLSX: `http://fileformat.wikia.com/wiki/XLSX` (accessed January 12, 2019)
[2]Google Spreadsheets: `https://www.google.com/sheets/about/` (accessed January 12, 2019)

the utterance authors may also specify a *backchanneling* category from which Skene will randomly take an utterance to perform it immediately after the person finishes speaking. With proper authoring, the use of these features can easily allow the robot to seem like it's understanding the users, even if no speech recognition or actual dialogue management is used at all.

5.1.4 Other SERA modules

In order to implement fully interactive scenarios, various other modules were also developed. We won't go into full detail, but outline some of the ones that we consider most relevant.

Kinect modules

The Microsoft Kinect has been one of our major components for User Perception. We have used modules both for Kinect v1 (initially) and then for Kinect v2 after the latter was released. It was used mostly for face-tracking, so that the Behaviour Manager and Animation Engine can be informed where to look at when performing gaze-tracking. Additionally we have used it to detect face-direction, in order to have a hint of where the users are gazing at - however face-direction is not sufficient for that process given that user may be facing a given direction with their face, while gazing at another one using their eyes. For proper gaze-direction detection one would have to use another system that can detect eye-gaze direction. Because the Kinect also includes an array of microphones we have also used it in some situations for speech-direction detection, and from there, to detect the active speaker when multiple users (faces) are detected. Although it not a very reliable option, it was used in situations where we did not want to require users to use a lavalier microphone, such as in public demonstrations and events. Another useful detection that can be made using the Kinect (although we did not use it in our scenarios) is the pointing-direction, by taking the users' detected skeletal information, and using the direction of the active arm. However we have found that the Kinect performs skeleton and facial detection separately, and in cases where the users are sitting, face-detection works, while skeleton detection does typically not.

Sound Detector

The sound detector modules takes the input of a pair of audio-inputs, such as two lavalier microphones, which can be connected to the computer using a stereo input audio device (such as a small mixing desk), or even a portable recorder such as the Zoom H4n or Zoom R6. It was used in user studies in which we did not want to rely on the Kinect for active speaker detection, and where there was some initial set-up time for the participants. For public events, placing and adjusting the lavalier microphones would be too impractical, so in those cases we would use the Kinect.

Speaker Rapport

This module takes the active-speaker detection information (provided either by the Kinect or the Sound Detector, as both produce the same events), and generates a Gaze action towards the active user. It also performs volume mimicking, i.e., it takes the measured decibels of the active speaker's sound, and attempts to match it, so that the robot will speak louder when the users are excited and speaking louder, and lower then they are more relaxed and

speaking lower. The Gaze action does not include coordinate information - instead it instructs to gaze towards the possible active speaker possibilities (in our case either Left, Right, Both or None). The Behaviour Manager will be the one to *know* where the Left or right users are located, and direct the configured Gaze action to the Animation Engine. When the active speaker is None then it will gaze at a random point. When it is Both, then it will alternately gaze between each of them with a fixed time interval (e.g. 2000 ms).

Media Player

For scenarios in which participants were required to watch a video with the robot, we developed the Media Player module, which allows to play media files full-screen in a computer that is co-located with the participant. The requests to play or stop the media would all come from the scenario's AI module.

5.2 The Nutty Pipeline for Programmable Robot Animation Engines

The Nutty Pipeline was formulated to inform the design, development and execution of the programmable robot animation engine. Just like the Nutty Workflow (Section 4.3), the pipeline presented here is deeply inspired on the work developed previously with the Nutty Tracks animation engine (Section 6.1), which was used as a sandbox to explore and develop new robot animation techniques for interactive applications [121, 42, 157].

Such an animation engine in *Nutty* terms, is a program that continuously runs a sequence of steps at a given rate, in order to produce on-line motion for a robot, based on interactive parameters specified by an AI or tele-operation module, and on user- and environment-based perception data. The **Nutty Pipeline** lies within the animation engine, and configures the steps that run on each animation cycle. The choice of those steps specifies how the motion is effectively produced, given a set of inputs, rules, and various types of motion generators. The concept of the programmable animation pipeline is deeply inspired by programmable graphics pipelines such as the one provided by OpenGL [174]. It means that the actual execution pipeline is not fixed, but instead, can be programmed to specify both the execution layout, the steps that will run, and how they are parametrized. It allows an animation engine to be used with different embodiments and applications, by introducing the new concept of *Animation Program*. Figure 5.6 illustrates how the Nutty Animation Pipeline fits within a *Nutty*-based animation engine.

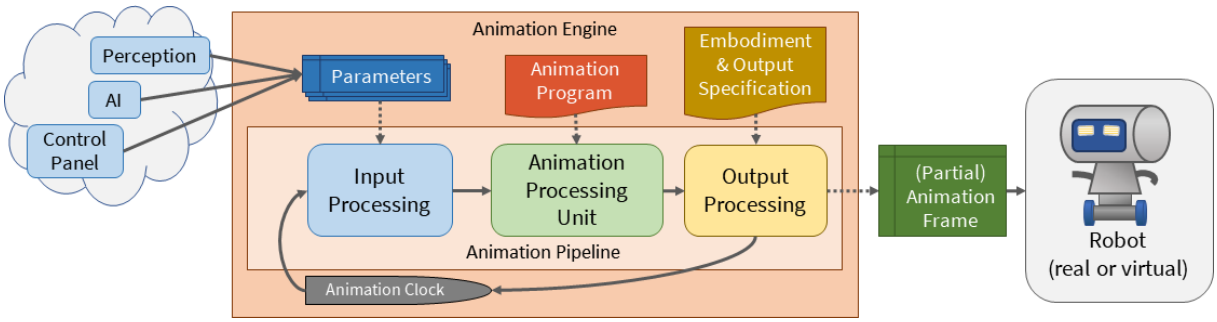


Figure 5.6: A *Nutty*-based animation engine, including the *Nutty Pipeline*. At each clock cycle, the Input parameters along with the selected Animation Program are provided to the Animation Processing Unit, which outputs a (Partial) Animation Frame containing the motion parameters for each programmable DoF.

The **Input** to the pipeline consists of parameters that are provided by other components such as the *AI* or the *Perception Engine*. Those input parameters can be very diverse taking several forms such: gaze-target coordinates;

expressive posture to exhibit; a pre-designed animation to play; an array of emotional-state values, and even particular custom commands such as *reset posture*, or *activate idle-motion*. There is no fixed specification for the animation pipeline input, which may need to be designed and developed for each particular situation.

The **Output** of the pipeline generates an *Animation Frame* that is compliant with the currently selected embodiment and output module. Although the embodiment and output are typically enforced to work together, it is important to distinguish them. The embodiment specification defines the available DoFs and their layout, or hierarchical structure. The output typically connects with either the Robot Software or the Pre-Viz's API, in order to render the animation frame, either on the physical embodiment or on its virtual representation.

An **Animation Frame** (AF) is a data structure, containing both header information and a matrix of motion parameters for each programmatically animatable DoF. The header may contain information such as the embodiment designation and the frame's delta-time. For each DoF, the motion parameters may have various data types, depending on the kinematronic dimension of the DoF. For non-integer numeric values, it may be as simple as a single, absolute set-point (no derivatives), or include 1^{st}, 2^{nd} or 3^{rd}-order derivatives (velocity, acceleration and jerk). However it may also contain discrete or enumerate values, which are more appropriate for e.g. a display-expression component with pre-defined expressions. We also distinguish between an Animation Frame and a **Partial Animation Frame** (PAF) in that the partial animation frame may contain only part of the whole list of DoFs (or in some cases, even none - an *empty* animation frame). This allows the pipeline to output only the signals that have been modified in each cycle, allowing to control different DoFs at different rates, and to perform blending and other operations using only a selected set of DoFs. When we refer to an AF is, it always contains parameters for all the DoFs (i.e., a *full* animation frame), while a reference to a PAF means that it may contain all or only part of the DoFs, and even be an empty frame, with no DoFs (which therefore produces no effect).

The central component of the Animation Engine and Pipeline is the **Animation Processing Unit** (APU), which executes an **Animation Program**. In the Nutty pipeline, an Animation Program takes a similar role as a *Shader* program in the *OpenGL* pipeline [174]. The APU can be developed to run at different levels of complexity, depending on the requirements of the robot-animated application, as illustrated in Figure 5.7.

The main building-block of the APU is called an **Animation Block** (AB). Multiple variants of ABs are created for different purposes. Each of these blocks takes in a set of input parameters, and generates a PAF through a specific method such as playing an animation file, or generating a motion signal through a mathematical formula. We further distinguish an *operator* AB as one that takes in a PAF that already contains a signal and modifies it, versus a *source* AB, which provides a source for the signal and generates it. In many cases the AB will also manage an internal state, such as in the case of an animation file player, for which, given the delta-time as input, the AB calculates the new time-position within the animation, and outputs the respective frame. That allows each AB to control how the signal is produced in the time-domain, along successive cycles of the animation engine. Given that they output PAFs and not necessarily AFs, an AB may also be some sort of single-dimensional motion generator such as a sine-wave or 1D Gaussian noise-generator. The pre-loaded Animation Program will specify the type of APU that is required, describe the required ABs, and specify how they are laid out into sequences, layers and stages.

The following list provides an overview of various complexity levels for APUs. A given animation engine can be developed to support only up to a specified level of complexity, or support all of them. Later, the AF will let the engine know what kind of layout is required to be set-up.

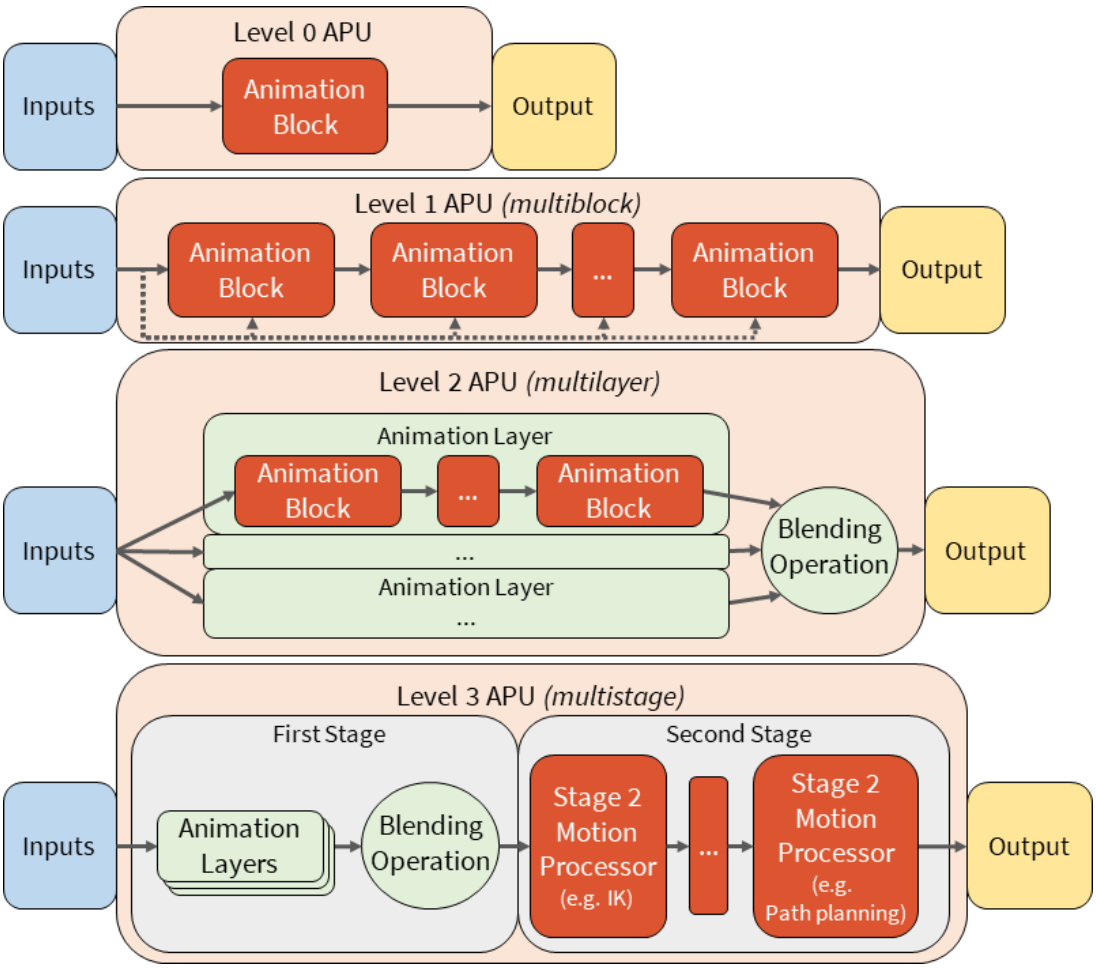


Figure 5.7: The four types of *Nutty* APUs.

The Level 0 APU is the simplest form of APU, and contains a single Animation Block. Conceptually, a Level 0 APU is also interchangeable with an AB, as both contain a single execution step.

The Level 1 APU supports multi-block processing, or a sequence of Level 0 APUs. Each AB may output to another AB and therefore it allows for more complex animation, that is achieved by sequential composition of ABs.

The Level 2 APU supports multi-block, multi-layer processing. Each sequence of AB blocks composes a single layer and is equivalent to a Level 1 APU. The various layers are blended using a specified Blending Operation, in order to produce a final, single PAF.

The Level 3 APU supports multi-block, multi-layer and multi-stage processing. At the moment we define only two stages. The **First Stage** consists of a Level 2 APU. The **Second Stage** allows to include more complex and intensive motion-generation processors such as inverse kinematics (IK) or path-planning. The Stage 2 processors are meant to be used as post-processing steps, and should be applied to several - or all - of the DoFs simultaneously. Nutty Tracks [121] is an example of a Level 3 programmable animation engine[3,4].

Note that depending on the requirements of the animation engine, one may create other forms of APUs, such as a multi-stage, multi-block APU that does not supports layers, or a multi-layer, single-block APU that does not support sequential composition.

5.3 Animation Tools for Social Robots

When including creative artists such as animators into the development workflow, one of the first question that arises is the tools that the artists can use to author and develop expressive behaviour for the robot. Typically those artists are commissioned to produce only pre-authored animation files that can be played back by the animation engine. This may be achieved by either developing a custom-build GUI that allows them to directly develop on the system's tools, data types and configurations, or to allow the artists to use their familiar animation tools such as 3dsmax[5], Maya[6], SideFX Houdini[7] or even the open-source Blender software[8]. These existing animation packages allow to export animation files using general-purpose formats such as Autodesk FBX[9]. That requires the animation engine to support loading such formats, and to convert them into the internal representation of pre-animated motions. Alternatively, and as most of those software support scriptable plug-ins, one may develop such a plug-in that allows to export the motion data into a format that is designed specifically for the animation engine.

Upon our introduction of the programmable animation engine, and of animation programs, it also becomes necessary to understand how the animators can contribute to such animation programming, alongside with their participation in the motion design.

[3] `https://vimeo.com/67197221` (accessed January 12, 2019)
[4] `https://vimeo.com/232300140` (accessed January 12, 2019)
[5] `https://www.autodesk.com/products/3ds-max/overview` (accessed January 12, 2019)
[6] `https://www.autodesk.com/products/maya/overview` (accessed January 12, 2019)
[7] `https://www.sidefx.com/products/houdini` (accessed January 12, 2019)
[8] `https://www.blender.org` (accessed January 12, 2019)
[9] `https://www.autodesk.com/products/fbx/overview` (accessed January 12, 2019)

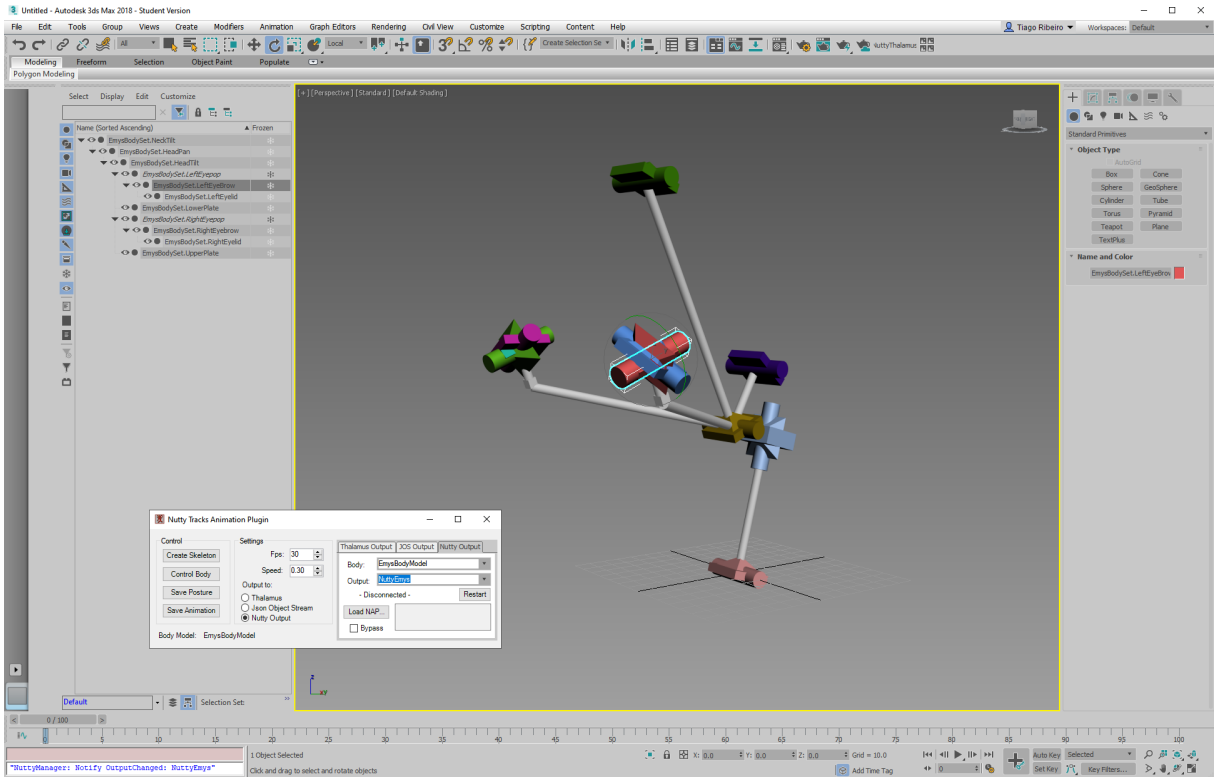


Figure 5.8: A screenshot of the Nutty Tracks plug-in for Autodesk 3dsmax, illustrating the skeletal animation rig created by the plug-in. An animator can generate this rig through the simple click of a button, and then use the plug-in to export the final animation to a Nutty-compatible animation file.

5.3.1 Animation Design Tools and Plug-ins

We argue that for simple cases, developing an e.g. FBX import for the actual animation engine run-time environment is a good choice. In this case the learning curve for the animators is almost inexistent, given that they will be working on their own familiar environment. They will only need to adapt to specific technical directions such as maintaining a properly named and specific hierarchy for the joints and animatable elements, so that those can be properly imported later on. When the nature of the project or application does not allow to rely on third-party, or proprietary software, then the only option may be to develop a custom animation GUI, which poses as the most complex and tedious one. However our feeling has been that the creation of plug-ins for existing, third-party animation software provides a good balance between development effort, usability, user-experience and results.

The creation of plug-ins for existing animation software includes the same advantages and requirements as in the first case, of developing an animation-format importer for the engine. Animators will be familiar with the software, but may have to comply with certain technical directions in order for the plug-in to be able to properly fetch and export the motion data. Figure 5.8 shows an example of the Nutty Tracks plug-in for Autodesk 3dsmax. By having the EMYS embodiment already loaded in the Nutty Tracks engine, the plug-in can create an animatable rig for the robot, through the click of a single button, based on the embodiment's hierarchical specification including rotation axes, joint limits, etc. Optionally it may even include the actual geometry of the robot for a more appealing experience. From here on an animator may animate each of the gizmos that were created for each of the robot's animatable DoFs, using his or her typical workflow and techniques.

However, the development of such a plug-in also allows to augment the creative development workflow, by

adding visual guides directly into the viewports of the animation software, in order to represent technical constraints that are required specifically for robots, such as kinematic ones (e.g. velocity, acceleration, jerk limits). Figure 5.9 shows an example of a plug-in developed for Autodesk Maya, to show the *trajectory-helper* of a given mobile robot platform, which highlights the points in the trajectory that break some of the robot's kinematic constraints. In this case, green means that the trajectory is within the limits, while the other colors each represent a certain limit violation, such as maximum velocity exceeded (orange), or maximum acceleration exceeded (pink) or maximum jerk exceeded (red). Based on this visual guide, the animator knows where the trajectory must be corrected, and is able to readily preview how the fix will look like, while making any further adjustments to the motion in order to ensure the expected intention or expression is properly conveyed without exceeded the physical limits of the robot.

Other useful features may be to perform automatic correction of such constraints, while rendering the result directly within the animation environment, thus allowing the animators to fix the motion that results from enforcing such constraints, in a more interactive way. From what we have gathered however, animators are typically not happy to have a tool that can change and control their animations. Instead, the preferred option is to keep the artist-animated version of robot untouched by the plug-in, and to create an additional copy of the same robot model. This copy, which we call the *ghost*, will, in turn, not be animatable or even selectable by the animator, but instead, will be fully controlled by the plug-in. Therefore, when the animator is previewing the playback of its animation, the plug-in will take that motion and process it in order to enforce the kinematic limits. The resulting corrected motion is however applied only to the ghost, which therefore moves along with the animated robot. If at any point, the animated motion did exceed the limits, the ghost will be unable to properly follow the animated model due to the signal saturation, which allows the animator to have a glimpse not only of where the motion is failing to comply with the limits, but also how it would look like if the limits were enforced. In some cases the animator might actually feel that the result is acceptable, even if the originally designed motion would report limit violations on a trajectory-helper solution such as the one of Figure 5.9. Note that in the case of the *ghost-helper* technique, whenever the final animation is exported, it should be exported from the *ghost* robot, which contains the corrected motion, and not the animated robot which does not.

In summary, the two major robot-animation features we have presented, and that can be provided through the use of animation software plug-ins, are the *trajectory-helper*, as presented in Figure 5.9, and the *ghost-helper*, described in the previous paragraph. Depending on the animator's preferences, and the scripting capabilities of the animation environment, either one or both of the features can be used. The ghost-helper seems to provide a more agile solution, as the animators aren't required to fix all the limit violations. As long as they accept the motion provided through the ghost, the problem is considered to be solved, thus allowing them to complete animations quicker than using the trajectory-helper. The trajectory-helper however allows an animator to better ensure that all the points of the trajectory are smooth and natural, and especially that the automatic correction (achieved e.g. through signal saturation) will not introduce any other unexpected phenomena. This feature is especially important when animating multiple robots[10], to ensure that each of the individual auto-corrections do not place the robots in risk of colliding.

Without the ability to preview or at least evaluate the animated motion directly within the animation environment, the animators would need to jump between their software, and a custom software that solves and reports on those

[10] `https://gagosian.com/exhibitions/2018/urs-fischer-play/` (accessed January 12, 2019)

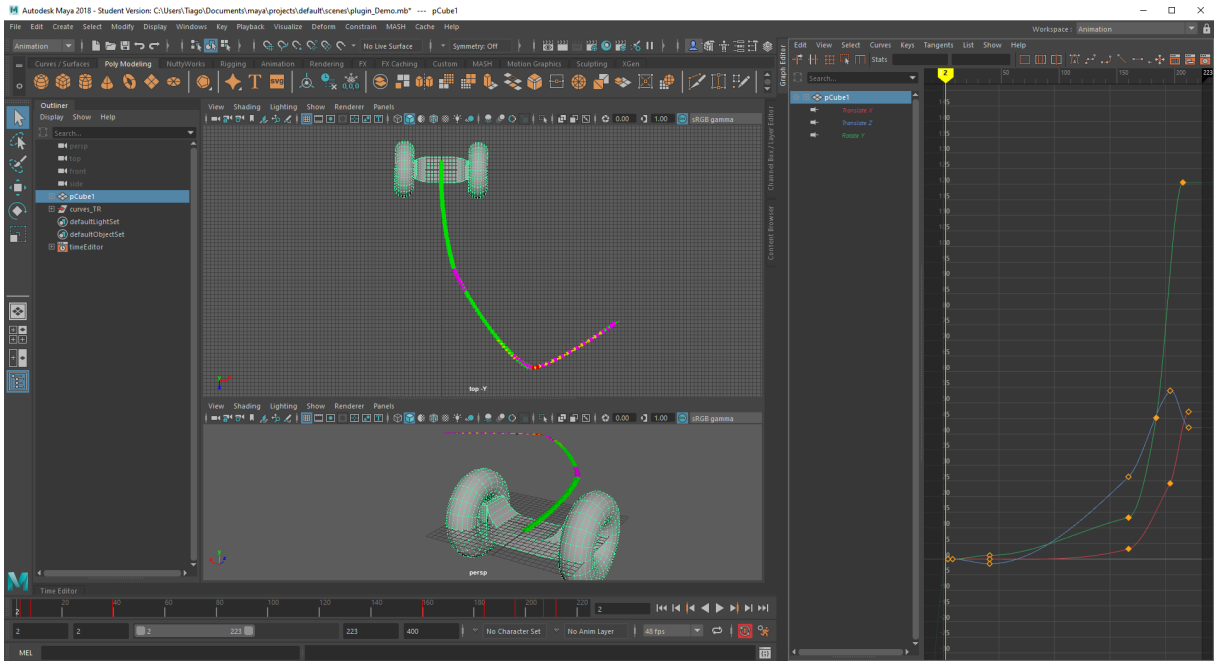


Figure 5.9: A screenshot illustrating the robot-animation trajectory-helper feature implemented through a plug-in into Autodesk Maya. This feature draws the motion trajectory as a path directly into the scene of the animation software, and highlights the points of the trajectory that break any of the robot's kinematic limits.

issues, while providing typically a mediocre or even no visual feedback on what is happening, and what needs to be fixed. Besides making it a more complex workflow, that option also hinders and breaks the animator's own creative process.

Finally, an additional feature that can be developed through plug-ins for existing animation software is the ability to directly play the animations through the robot software or interactive pre-visualisation system. This allows the animators to include testing and debugging into their workflow, by being able to see what will happen with their animations once they become used during interaction with the users and the environment.

5.3.2 Animation Programming Tools

Animators working with social robot application are required to learn some new concepts about how motion works on robots, in order to identify what can or cannot be done with such physical characters, as opposed to what they are used to do in fully virtual 3D characters. Besides having to adapt to certain technical requirements when building their characters and animation rigs, they may also need to learn how to interact with some other pieces of software that will allow them to pre-visualize how the designed motion will look on the robots during actual interactions.

At some point the character animators will acquire so many new competencies and knowledge that they become actual *robot animators*, an evolution of animators that besides being experts on designing expressive motion for robots, may also have learned other technical skills as part of the process. One such skill is what we call animation programming. The difference between a non-robot-programming animator, and a programming-robot animator is akin to the difference between a texture artist and a shader artist (or lighting artist) in the digital media industry. The texture artist is a more traditional digital artist that composes textures that are *statically* used within digital media. A shader artist is able to take such textures, or other pattern-generators, and configure the shaders (i.e.,

programs) to adapt and change according to the environment parameters and applications. The shaders are, in that sense, *programmable* textures. Similarly, and animation programs are *programmable* animations.

Animation programs can, at a very basic level, be specified by some kind of mark-up code. However, we believe the best option to be taking inspiration from currently existing tools. Both Autodesk's Slate material editor[11], and the Unreal Material Editor[12], are well-established artist-friendly shader-programming interfaces. Houdini[4] is also known for its visual graph-based visual effects programming system. Pure Data[13] allows visual, sound and performance artists to develop their own musical instruments, visual effects processor, or any other kind of interactive system, using a visual block-graph paradigm that allows to simultaneously run the program while also allowing it to be composed, all in real-time. As such, we argue for the creation of similar, artist-friendly, animation-programming editors.

These new animation programming tools can be built from scratch as standalone GUI application (e.g. Nutty Tracks), or using game development tools such as the Unity Engine[14], which allows for the scripting of new interface tools. In this case, because a game engine such as Unity3D also provides 3d visualization and animation tools, it could be extended by an animation programming tools in order to become a fully-fledged animation designing, programming and pre-visualization tool.

Nutty Tracks provides an example of how such an editor may be presented[15]. Its programmable animation GUI is also shown in Figure 5.10 and further details about it are further described in Section 6.1. It was conceptualized to allow an animator to load and pre-visualize how animations and expressive postures designed in another software (e.g. 3dsmax) will look like when procedural layers of motion are added, such as ones that generate idle-behaviour, user-face tracking, or inverse kinematics. Such output motion is processed by the Level 3 APU in Nutty Tracks, and could not be properly visualized within the typical animation design software (which are based on a timeline). However the process of composing and tweaking the animation program using animation blocks follows a workflow that is similar to the one found on other artist-friendly applications that inspired us.

Despite such effort, it will still be the case that such an animation program editor will pose as a truly novel tool for the animators, with a steep learning curve. An animator may e.g. be familiar with the concept of an animation layer, which does not match the one used in the visual animation program editor. The idea of composing programmable animations using operator- and generator-blocks may have a paralell with certain motion control nodes found in some animation software, but the way they are used and composed may also not seem intuitive or obvious for the traditional 3D animator. As such, it is required that these tools are developed with a user-centered design perspective, in close collaboration with the end-users, who are the actual animators, and to ensure the GUI provides an understandable translation between the animator's mindset, and the underlying mechanics and pipeline of the animation engine.

[11] `https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2017/ENU/3DSMax/files/GUID-7B51EF9F-E660-4C10-886C-6F6ADE9E8F56-htm.html` (accessed January 12, 2019)

[12] `https://docs.unrealengine.com/en-us/Engine/Rendering/Materials/Editor/Interface` (accessed January 12, 2019)

[13] `http://puredata.info` (accessed January 12, 2019)

[14] `https://www.unity.com` (accessed January 12, 2019)

[15] `https://vimeo.com/67197221`(accessed January 12, 2019)

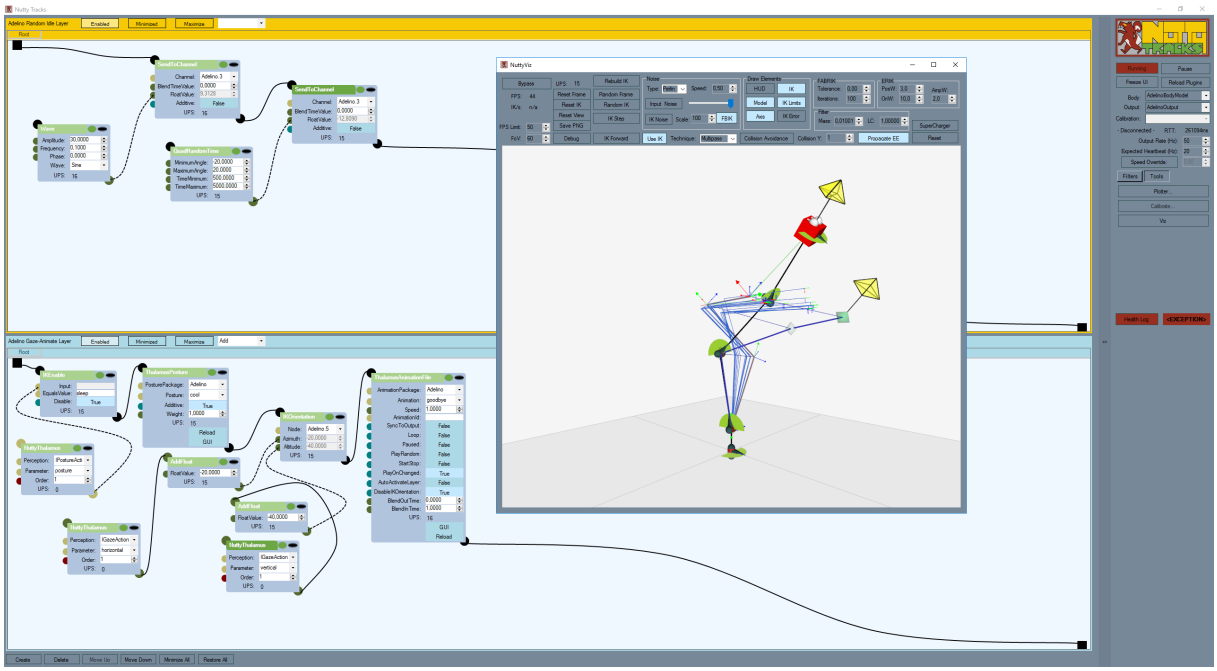


Figure 5.10: The Nutty Tracks GUI, used for animation programming in a multi-layer, multi-block visual editor. Within the figure, we see several different Animation Blocks. Some of them take PAFs as input and/or output (black connection points), while others provide colored connection points for single-dimension signals, which are color-coded depending on the type of signal they carry (e.g. floats, integers, strings, etc..). It additionally includes an Inverse Kinematics interactive visualizer which allows an animator to tweak some of its parameters, in order to adjust the generated motion to the robot's kinematic capabilities.

Chapter 6

Robot Animation Technology

6.1 Nutty Tracks

Nutty Tracks (Nutty) is a symbolic animation engine based on CGI methods that allows to animate both virtual and robotic characters [121]. It implements the Nutty Pipeline as a Level 3 APU (Section 5.2) and is simultaneously a design-time and run-time environment, i.e., it is used both for designing and programming animation, as well as to execute it in real-time during interaction.

Using Nutty provides us with high flexibility regarding the design, blending and modulation of animations on any robot. It allows to use professional animation tools (e.g. Autodesk 3ds Max[1]) to design animations and postures, and provides a generic translation layer between the character's animation parameters and the actions and parameters that arrive from other components in the system.

One of the principles of Nutty is to work on animation at a symbolic level. This means that while the system is aware of the structural hierarchy of the robot, its animation isn't processed at the level of the actual joints, but on symbolic channels, which may represent joint motion or some other signal (similar to [104]). These symbolic joints can actually be mapped to a real robotic joint, or to a set of joints, thus also allowing to work as an aggregated joint (e.g. we can animate a 1-DoF joint called VerticalGaze which is later decomposed into several real motors of the real robot's neck).

The composing of animation programs in the Nutty Tracks GUI follows a box-flow type of interface greatly inspired in other programming tools commonly used by artists, such as the Unreal Engine [2], Pure Data[3] or SideFX Houdini[4]. Figure 6.1 shows the Nutty Tracks GUI.

We recall here the schematics of the Nutty Pipeline as Figure 6.2 and illustrate the Nutty Tracks APU on Figure 6.3. Note that within Nutty Tracks, NAP stands for *Nutty Animation Program*. The Body Model is the entity that contains the *Embodiment's specification*, while the Output Plugin is a separate component that can stream the frames through different transports (e.g. TCP, JSON-TCP), or different interfaces (e.g. Arduino). The input control to Nutty Tracks is provided through Thalamus, which integrates Nutty Tracks into a SERA environment (Section 5.1.2).

[1] 3ds Max: `http://www.autodesk.com/products/3ds-max/overview` (accessed January 12, 2019)
[2] Unreal Engine: `http://www.unrealengine.com/` (accessed January 12, 2019)
[3] Pure Data: `http://puredata.info/` (accessed January 12, 2019)
[4] `https://www.sidefx.com/products/houdini` (accessed January 12, 2019)

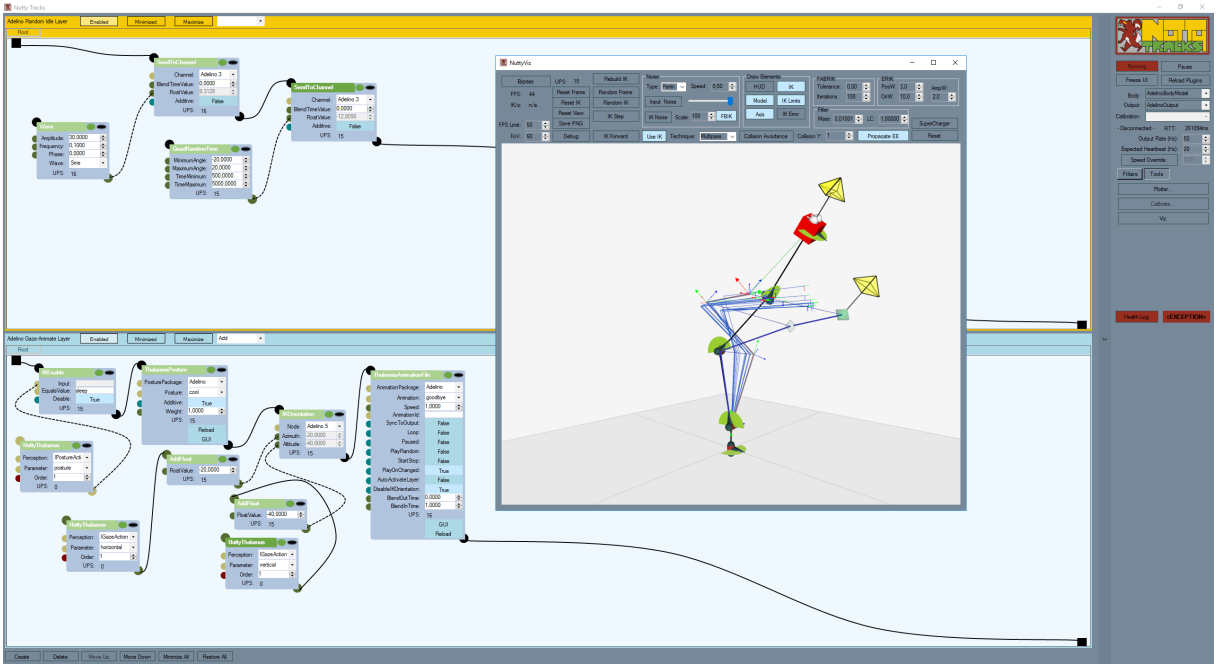


Figure 6.1: The Nutty Tracks standalone GUI, used for composing animation programs, and to execute them in both a virtual window (for diagnostics) and on the real robot.

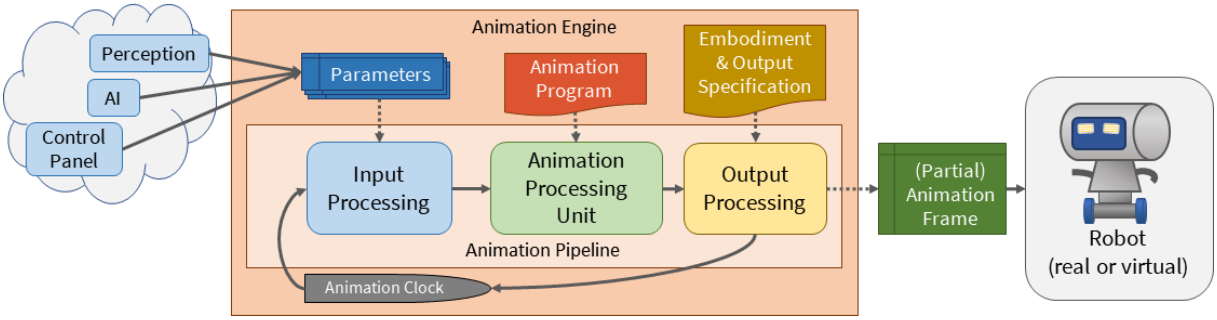


Figure 6.2: (reiteration of Figure 5.6) The *Nutty Pipeline*. At each clock cycle, the Input parameters along with the selected Animation Program are provided to the Animation Processing Unit, which outputs a (Partial) Animation Frame containing the motion parameters for each programmable DoF.

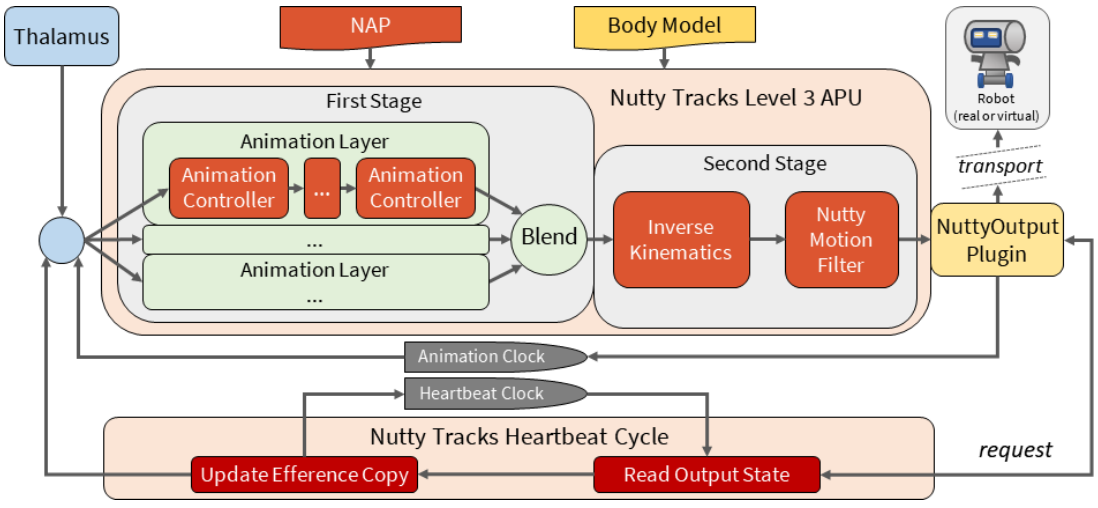


Figure 6.3: The Nutty Tracks Level 3 APU.

The following is a list of the main features and contributions of Nutty Tracks:

a) **Provide a body-agnostic animation representation**

Animation is represented in a generic animation frame type we call the Animation Buffer (Ani-Buffer). The Ani-Buffer is explained in more detail further in section 6.1.4.

b) **Provide symbolic degrees-of-freedom**

By symbolic, we mean that all the DoFs that are animated are just containers of animation data and are not yet assigned to any specific virtual or hardware articulation. Abstracting DoFs from the skeleton when animating robots has been previously suggested by [104]. Therefore, a symbolic DoF may either contain motion data for one specific joint, for an aggregation of joints, or for a non-motor expressive channel such as a LEDs brightness, or the morph weight of a morph target in a virtual face. The idea behind using symbolic DoFs is to animate certain expressive channels, and not to specific parts of the animated body. A clear example would be the Gazing channels. A basic embodiment would be able to gaze vertically and horizontally, which means that there are two gazing DoFs. However, some embodiments may have several articulations for the VerticalGaze (e.g. humanoid characters). Nonetheless, what actually matters expressively is the overall vertical angle that is applied to all the gazing articulations. Instead of having to consider how many articulations a character's neck has, we can animate just one high-level angular component which in rendering will be decomposed into those actual articulations. This allows symbolic gazing animation to be used on any character that is able to perform gazing behaviour, regardless of the number of articulations it has for doing so.

c) **Modular layered animation controller composition using box-flow**

One of the most useful features gained from using a generic animation representation format is that reusable Animation Controllers (ACs) can be developed to process motion signals regardless of the embodiment that is being animated. These can then be composed in a controller-chain form, and also in layers so that different parts of the NAP can be composed independently, and activated and deactivated depending on what is needed. The composition of the controller chains in the GUI follows a box-flow approach, i.e., the user adds ACs to a layer and then uses click-and-drag between connection points in order to connects the ACs (more details in Section 6.1.7).

d) **Control animation parameters from an AI**

Nutty Tracks does not intrinsically provide an integration with any type of AI engine. Instead, this is performed by developing a plug-in for Nutty that connects to that AI. What Nutty provides is that the ACs are defined with controllable parameters that can be linked to other ACs. Taking as example an AC that generates a sine wave given an amplitude and frequency, this sine wave can be used to control some DoF of a character. The amplitude and frequency parameters can either be specified in design-time, or be linked to some other AC that received input from an AI. In our implementation we implemented a Nutty-Thalamus plugin that allows such information to be provided to Nutty Tracks as Thalamus messages. Note that an AC is like a blackbox that outputs some kind of signal, and may receive some kind of input. An AC that is part of an AI plug-in would be a box with no inputs (in Nutty) but that internally receives messages from that AI and translates them to some signal type and outputs that value. Further detail on the types of signal used by ACs and AC-Parameters are provided in section 6.1.7.

e) **Body-agnostic Output**

All the previously mentioned goals sum up to the final designation of supporting any kind of embodiment. By providing a generic animation representation, and being able to compose animation programs out of reusable controllers, the final animation frame remains in the generic format. Although during execution the system works with a specific embodiment configuration, the main statement on body-agnostic output is that the pipeline does not enforce any specific restriction on the type of embodiment that can be used. After just developing a BodyModel for each embodiment and an appropriate output plug-in, the same animation system and animation controllers can be used to create different NAPs for each embodiment and context. Moreover, by providing mechanisms that allow the ACs to be controlled from an external AI, the whole animation system can be reused along with other components of the overall AI agent, in different applications. The BodyOutput component is detailed further in section 6.1.2

f) **Provide embodiment proprioception to the Animation Controllers**

The Heartbeat mechanism was introduced in order to provide the animation engine with the possibility of performing dead reckoning. It is useful especially on initialization, as it provides an integrated mechanism to know what is the current state of a robot's joints and thus avoiding a sudden abrupt movements in start-up. It can also be used later by Animation Controllers that are created to act based on body feedback. The Heartbeat response from the embodiment is expected to contain the sensor-measured values of each of the joints. Optionally it can also provide e.g. the current measured centre of mass if the embodiments has sensors to provide so, or other data that the Output plugin and embodiment are able to provide.

6.1.1 Execution

Nutty Tracks starts by loading a specified NAP (Section 6.1.3), and setting the specified NuttyOutput and BodyModel (Section 6.1.2) which contains the representation of the selected character's embodiment in terms of available degrees-of-freedom (DoFs) and its hierarchy. The cycle executes the Nutty Animation Program (section 6.1.3) in its APU. The NAP specifies one full iteration of the animation cycle, and contains a set of layers with chains of Animation Controllers (section 6.1.7) that will generate and process the motion data into a partial animation frame represented as an Animation Buffer (section 6.1.4). After each cycle, the final Animation Buffer is sent to the embodiment via a NuttyOutput plugin. The Animation Cycle can run faster than the specified output rate. Therefore in order to not overflow the output, the Output component contains a Choker that limits the output rate to a fixed number. In order to provide smoothness on robotic embodiments, this rate should be at least 30Hz [104]. It also contains a DirtyFilter, which only outputs the frame if it has been modified in the last iteration and a Repeated Value Filter (RVF) which abstains from re-outputting values that have not changed since the last cycle.

The Heartbeat Cycle (on the left) runs in parallel. This cycle was added in order to add additional support for robotic embodiments. While in a virtual embodiment one can render the character immediately with any joint configuration, robotic embodiments work differently. Instructions are sent to the a servo controller to command the servos (joint motors) to move to a particular position/rotation, with a specified torque (or speed). Then the servo acts on its own. The interpolation of the joint between two given angles is actually performed internally by the servo. Servos therefore generally provide sensors which can be used to read the current position/rotation of

a joint. After instructing a servo to move to a particular angle, one can use the sensors to verify not only when, but if it has actually achieved the desired angle. Hoffman has previously relied on dead reckoning to optimize the bandwidth of communication with the servos [136]. This technique was adapted from aviation and mobile robots navigation control. Instead if polling the servos' sensors after each instruction, they are polled at a slower rate. Between consecutive instructions, the servos are assumed to have moved according to the motion they were instructed to perform - this assumed output may be referred to as the efferent copy of the output. Upon sensor polling, the assumed rotations are corrected. In our pipeline, this polling happens at a fixed rate, slower than the Animation Cycle (e.g. 5Hz). The retrieved state is then updated in the Animation Cycle and made available to Animation Controllers.

6.1.2 BodyModel and NuttyOutput

The purpose of the BodyModel is to provide a description of the Animation Channels (further detailed in section 6.1.5) available in a given character embodiment. It is defined externally and loaded in runtime as a plug-in. It is tightly linked with the concept of symbolic DoFs, as this is the component that informs the system on what the embodiment's available DoFs are, and is later used by the NuttyOutput or external animation module to translate those DoFs to the actual embodiment's articulation. The BodyModel as seen in **figure 6.4** represents a given embodiment identified by a **Name**, a hierarchy of **BodySets** and a list of **Enslaving** rules. Each BodySet is an actual list of Animation Channels which represent DoFs of an animatable character. The purpose of having a collection of BodySets in the BodyModel instead of just a collection of DoFs is towards portability of animations across embodiments, as long as those all consider the same reference frame. Each BodySet is like a "namespace" of DoFs. This allows that animation files created for DoFs belonging to a given BodySet may be used with other BodyModels that contain the same BodySet (even if in overall, they represent a different character embodiment). An example of this would be to consider a generic Gazing BodySet that contains two Channels: VerticalGaze and HorizontalGaze. The Enslaving rules are specified to map DoFs from generic BodySets to the DoFs that are specific to the embodiment.

In order to provide a better understand of the BodyModel, **figure 6.5** shows how this would look for the EMYS robot. It contains three BodySets (one EMYS-specific, and two generic ones), along with how each of the generic DoFs should be mapped to EMYS's specific DoFs. Internally within the BodyModel, these rules also contain the calculation methods for distributing values through the enslaved DoFs.

The common workflow of development for each new embodiment would be to create a plug-in that contains both BodySets, a BodyModel and a NuttyOutput component. An example of the creation of a BodyModel is further detailed in Section 6.1.8. As to developing a NuttyOutput, that is tightly connected with the type of output desired, such as connection to an Arduino, to a TCP server, to ROS, or even back to Thalamus, and can therefore be developed

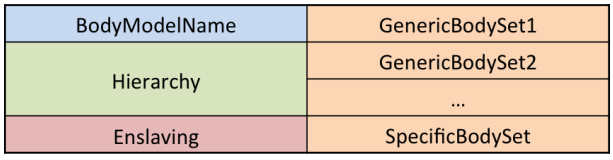


Figure 6.4: The contents of a BodyModel.

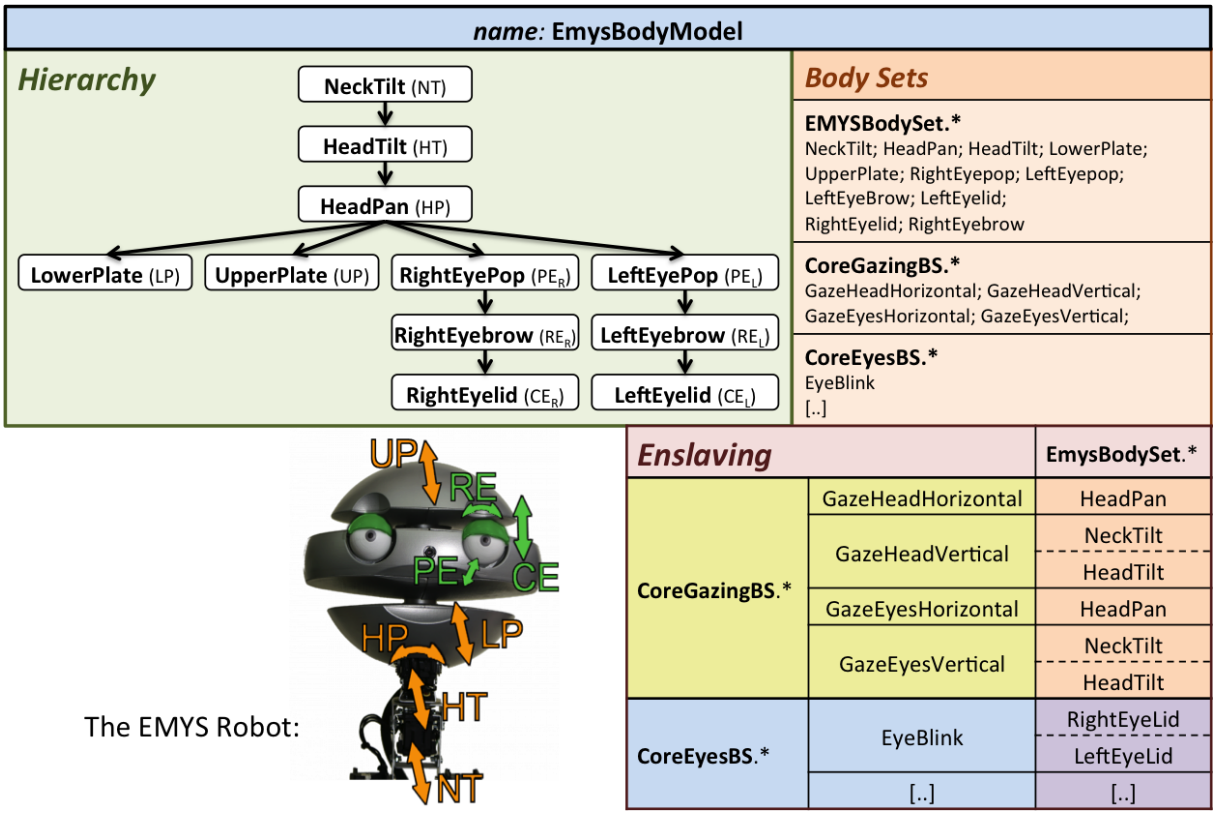


Figure 6.5: The BodyModel for the EMYS robot (with some example generic BodySets and correspondent enslavings).

by any software engineer according to those needs. In any case they are the most re-usable component of the pipeline. Generic BodySets may be distributed in separate libraries in order to be shared by different embodiments.

The NuttyOutput just implements two operations. **Animate**, given an Ani-Buffer which is result of a NAP interation, translates the Ani-Buffer using the BodyModel information, and renders it on the embodiment (or sends it to some external rendering system). **HeartBeat** polls the embodiment for its the current sensor-based state and provides the result back to the system. In case of most virtual characters, the HeartBeat can just return the Ani-Buffer that was provided to the last Animate operation.

6.1.3 Nutty Animation Program (NAP)

A NAP is the sequence of instructions that generates and processes animation data on each frame. The concept of animation as a program was inspired in computer-graphics (CG), where material shaders are programs that create and process the appearance of CG objects. In particular, inspiration from artist-friendly visual tools such as the Unreal Editor's Material Editor (mentioned in section 5.3.2) encouraged the creation of an animation system based on small modular pieces that could be composed according to different purposes and outputs. Just as in CG one can use texture generators (e.g., gradients, Perlin Noise) or pre-designed textures as input, in animation one can either load pre-designed animations or use motions generators (e.g. sine wave, Perlin Noise). And just as with image date, animation data can be processed and composed using operators and filters.

In runtime, the NAP serves as a *recipe* or configuration for Nutty Tracks to set-up the APU. It contains the definition of the Layers and the chain of Animation Controllers in each layer, along with the initial values for each

of the ACs (further detailed in Section 6.1.7). It may also contain other information such as Motion Filter settings, IK settings, or what BodyModel and Output to use, and any further configuration for them (e.g. COM port, TCP address and port, etc..). Having all that information in the NAP allows Nutty Tracks to be fully initialized through a script, and would also allow it to be ran as a *headless* APU (with no GUI), although the *headless* version of Nutty has not been implemented at date.

The creation of a NAP in Nutty Tracks follows a box-flow approach, so that the user can visually create the controller chains for each layer using the GUI, while immediately visualizing the output that it produces **while** it is being created (as happens with the Material Editor in the Unreal Engine[5]). A NAP can then be saved, so that it can later be loaded and used in run-time applications.

6.1.4 The Ani-Buffer

The Ani-Buffer (AB) is an object that contains animation information for a given set of DoFs. Its purpose is to represent a partial animation frame. By partial we mean that it needs not to contain animation info for all of the DoFs that the animated embodiment possesses, both to reduce the amount of frame data send when used with robotic embodiments, and to not overflow the lower-level motor control system with repeated data. As seen in Figure 6.6, this buffer contains some meta-data and a list of Animation Channels that can refer to DoFs of any type of embodiment. The list is filled in a per-frame basis by Animation Controllers (ACs) that can either generate such frame values or process existing values (filled in previously in the same animation iteration by other ACs). The meta-data fields contained in the Ani-Buffer are:

BodyModel to which the given frame applies. On creation, the Ani-Buffer is set to be using the BodyModel selected in NuttyTracks;

Dirty is a flag used to mark if any data was written to the current Ani-Buffer instance (so that it can by skipped otherwise);

DeltaSeconds is a double-precision value representing how many time in seconds has passed since the previous animation iteration;

At each animation cycle, an empty Animation Buffer is created for the first layer, and further fed to the its first Animation Controller (Section 6.1.7).

[deltaTime: 0.033]	EmysBodyModel {EmysBodySet; CoreBodySet}					Dirty: ☑
Channel	Value	Mask	Passthrough	Speed	Speed Mask	ValueType
EmysBodySet.HeadPan	52.0	☑	1.0	0.7	☑	AngleDeg
EmysBodySet.HeadTilt	12.0	☑	1.0	0.6	☑	AngleDeg
EmysBodySet.UpperPlate	0.45	☑	1.0	1.0	☒	Percent
CoreGazingBS.GazeHeadVertical	-24.0	☑	0.2	0.6	☑	AngleDeg
CoreEyesBS.EyeBlink	0.0	☒	1.0	1.0	☒	Percent

Figure 6.6: An example partial Ani-Buffer for the EMYS robot.

[5]Unreal Engine Material Editor: `https://docs.unrealengine.com/en-US/Engine/Rendering/Materials/Editor/Interface/index.html` (accessed January 12, 2019)

6.1.5 Animation Channels

Each Channel represents an animated expressive DoF (based on the specification of Kinematronics on Section 4.2). Along the animation pipeline (before being sent to the embodiment), it contains the following information:

• A double-precision animation **Value**;

• A boolean **Mask** value which states if the Channel was animated in this iteration;

• A double-precision **Passthrough** value between 0 and 1 which serves as an ”opacity” mask in order to facilitate per-channel blending (by default it is 1);

• A double-precision **Speed** value which is defaulted to 1. This provides additional support to robotic embodiments. It is intended to be used as a multiplier of the robotic servo's default torque value (i.e., by using a value of 0.5, the servo should be set to move at half of its default torque speed). This way ACs can also control the torques of a robot's servos along with their rotation. A further extension would also allow to include the full motion parameters of velocity, acceleration and jerk if required

• A boolean **SpeedMask** value which states if the Channel's speed was changed from the default value of 1. *Note: SpeedMask and Mask are marked independently*;

• A FrameType option which is Degree by default and represents the type of Value contained in the channel. Degree means it represent a rotation in degree angles. Other options are Radians, Percent (keeps values between 0 and 1, double-precision) or Valence (keeps values between -1 and 1, double-precision) depending on the needs of the embodiment (i.e., if animating a channel that actually represents a robot's light, Percent may be most appropriate to use).

6.1.6 Ani-Buffer Operators

The Ani-Buffer contains several operators in order to allow Animation Controllers to manipulate it:

SetChannelValue(*X*) fills in the specified channel with the value ***X***. It also marks that channel as active in the buffer's Mask field and sets the Dirty flag.

SetChannelSpeed(*S*) changes the Speed multiplier of the specified channel to ***S***, sets its SpeedMask and marks the frame as Dirty;

ClearChannelValue() resets the specified channel to its zero-values and un-marks its flags. If no other channel is being used, the Dirty flag is unmarked;

Conversion of a channel's value to another FrameType is provided internally from Degree to Radians and vice-versa. Conversion to and from Percent or Valence is performed by the BodyModel in order to consider the values' bounds (such as joint limits);

Retime(*T*) changes the DeltaSeconds value of the frame to ***T***;

Touch() marks the buffer's Dirty flag without having set any values;

Clone() creates a deep copy of the Ani-Buffer;

Multiply(*D*) multiplies all the Ani-Buffer's active Channels by the double-precision ***D*** value;

Add(*A*,*B*) creates a new Ani-Buffer containing the result of adding all the active Channels of two given ***A*** and ***B*** Ani-Buffers. Values, Speed and Passthrough are all added. The resulting Ani-Buffer uses the same BodyModel as ***A***. Channels that exist or are active only in ***A*** are copied to the result. The same applies to Channels that exist in both ***A***

and ***B*** but are only active in ***B***; Channels that exist only in B but not in A are ignored.

Subtract(*A*,*B*) works as Add, but performing subtraction of values;

Override(*A*,*B*) returns an Ani-Buffer that results of copying A and overriding it with the Channels that are active in B. This allows to combine partial frames by overriding instead of addition.

6.1.7 Animation Controllers and Layers

Animation Controllers (ACs) are blocks of code that perform a specific operation, and output either a single or multi-dimensional signal. Multidimensional signals are stored as ABs, while single-dimensional signals can be either floats, integers, strings or booleans.

There are two main types of ACs:

Generators have no main input signal and therefore solely generate an output.

Operators have a main input signal and therefore allow to take the input, process it somehow, and output the result.

Each ACs may also expose several *wired parameters*, which can be of any of the aforementioned types (including AB). Figure 6.7 illustrates the GUI that would be created for an example operator AC, which takes as input an AB, and outputs an AB based on a set of wired parameters. In the GUI we provide color-coded connection points so that the user knows what type of signals can be connected together (heavily inspired by the Unreal Engine editor).

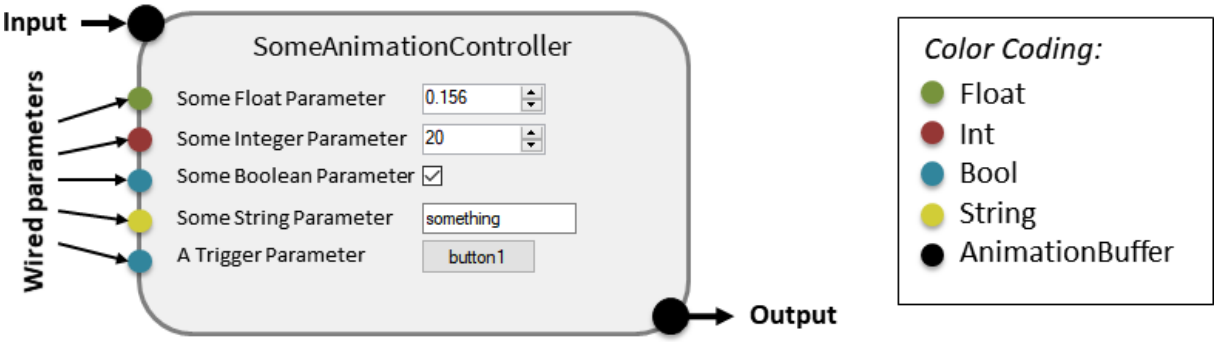


Figure 6.7: An example animation controller GUI with the signal color-coding using in Nutty Tracks.

Additionally a boolean parameter may also act as an On/Off *Trigger* to start or stop performing an action, or to change the output logic of the AC (inspired by the *bang* of PureData[6]). In that case, the parameter is initialized to *False* and an internal flag *active* is also set to *False*. At some moment the input may switch to *True* during one single frame (e.g. when a specific event arrives from the AI), which internally triggers some logic to change the output of the controller accordingly. Upon a change of the *Trigger* parameter to *True*, the *active* flag is also set to *True*, and the *Trigger* parameter is immediately changed back to *False*. Within the GUI of an AC it therefore acts as a button. While the internal *active* flag is *True*, the controller logic changes. If the *active* flag is still *True* and the *Trigger* parameter is again set to *True*, then the *active* flag changes to *False* (as does the *Trigger* parameter), thus changing back the logic of the controller output.

When Nutty Tracks is launched, the available Animation Controller types are loaded as plug-ins, and as such, can be created separately from Nutty Tracks depending on our needs, and shared within the community. Therefore

[6]Pure Data: `http://puredata.info` (accessed January 12, 2019)

each Nutty Tracks instance may actually contain a different set of available Animation Controllers, and thus exhibit different features from another instance, depending on the local plug-ins. This feature turns Nutty into not only a highly flexible animation software for robots, but also a highly extensible one.

When a NAP is loaded, Nutty Tracks creates the Animation Layers specified in the NAP and instances all the ACs and connections in each layer. Animation Controllers (ACs) are connected into a chain of execution (Figure 6.8) that generates and composes animation either procedurally or using animations and postures that were pre-designed (e.g. with Autodesk 3ds Max).

These chains of ACs are further composed into a hierarchy of layers that can be activated and deactivated during interaction in order to either blend or override their animated degrees-of-freedom[7].

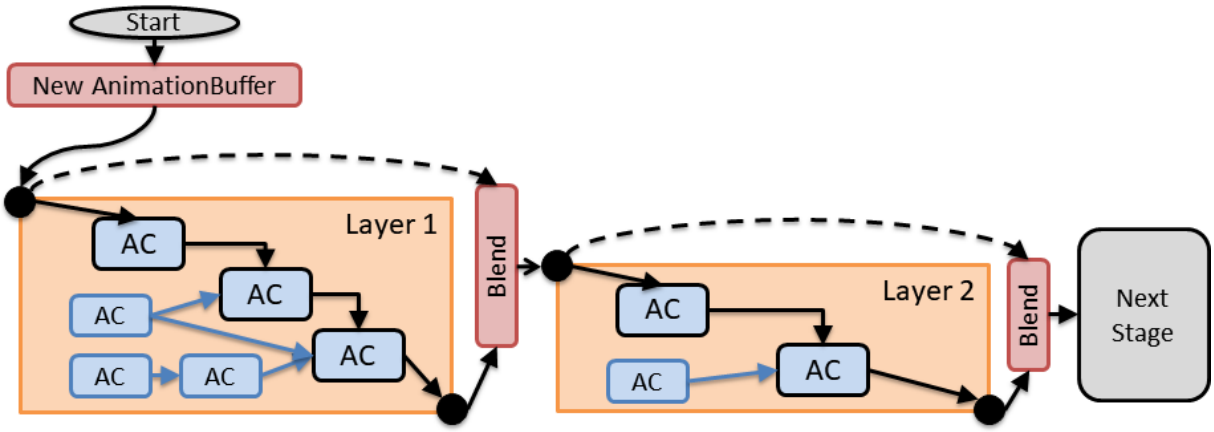


Figure 6.8: The structure and flow of a NAP.

Each layer starts with an empty Ani-Buffer (or an efferent copy if the Heartbeat is being used), which is fed to the first AC of the layer. Each AC processes the input AB and outputs the result to the next AC, or until it reaches the end of the layer's AC chain. The output of each layer is blended with the previously accumulated blended AB following either an addition logic, or an override logic. In any case, only the *dirty* channels are blended. The result of blending each two layers (the accumulated blended AB) is then fed as input to the next layer, i.e, except for the first layer, each one is initialized with the blended result so far. The main chain of ACs is the one that connects the layer's start point until its end point (outlined in black).

Besides receiving an AB as input (mandatory for main chain ACs), an AC may optionally have some of its parameters controlled by other ACs (in the figure outlined in blue), which compose the secondary AC chains.

The following is a list of Animation Controller examples. Many other ACs can be (and have been) implemented as plug-ins to Nutty Tracks depending on the requirements of the target application.

Generators

- **Float**:*float*(**value**:*float*) outputs ***value*** as a constant float signal.
- **RandomFloat**:*float*(**min**:*float*, **max**:*float*) outputs a random float value between ***min*** and ***max***.
- **IntStep**:*int*(**step**:*int*, **stepSize**:*int*) outputs the integer value ***step*** $\times$ ***stepSize*** (similar to what **MultiplyInt** would produce, except that this AC works as a generator, i.e., it has no inputs, only parameters,

[7]Nutty Tracks: `http://vimeo.com/67197221` (accessed January 12, 2019)

and can therefore be used at the beginning of a secondary chain).

- **Wave**:*float*(**amplitude**:*float*, **frequency**:*float*, **phase**:*float*, **shape**:*string*) generates a *sine* or *square* wave signal based on the value of ***shape***, with parameters ***amplitude***, ***frequency*** and ***phase***.

Operators

- **AddFloat**:*float*(**input**:*float*, **value**:*float*) outputs ***input***+***value***.
- **MultiplyFloat**:*float*(**input**:*float*, **value**:*float*) outputs ***input***×***value***.
- **AbsFloat**:*float*(**input**:*float*) outputs the absolute value of ***input***.
- **IfFloat**:*float*(**input**:*float*, **test**:*float*, **ifTrue**:*float*, **ifFalse**:*float*) outputs ***ifTrue*** if ***input***=***test***, or ***ifFalse*** otherwise.
- **StrConcat**:*str*(**input**:*string*, **other**:*string*) outputs the result of concatenating ***input*** and ***other***.
- **StrChanged**:*bool*(**input**:*str*) outputs *True* whenever the ***input*** value changes between two frames, otherwise outputs *False*. This logic is useful to control *Triggers*.
- **WriteToChannel**:*AB*(**input**:*AB*, **channel**:*string*, **value**:*float*, [**speed**:*float*]) invokes the **SetChannelValue** on ***input*** to set ***channel***'s value to ***value*** and optionally its speed to ***speed***.
- **AddAniBuffer**:*AB*(**input**:*AB*, **other**:*AB*) adds the values of each channel in ***other*** to the ***input*** AB (and returns the resulting AB).
- **Posture**:*AB*(**input**:*AB*, **posture**:*string*, **additive**:*bool*, **weight**:*float*) outputs an AB containing the values specified by the channels in ***posture***, either added to the ***input*** or overwriting them depending on the value of ***additive***, using ***weight*** to control the opacity of the blending.
- **Animate**:*AB*(**input**:*AB*, **animation**:*string*, **speed**:*float*, **loop**:*bool*, **StartStop**:*bool*, **pause**:*bool*) selects ***animation*** and starts playing it from the start whenever the ***StartStop*** *Trigger* is activated. The output advances the frame of the ***animation*** based on the animation's clock value multiplied by ***speed***. When the animation ends, it may either terminate, or ***loop*** until the ***StartStop*** is triggered again. The animation may also ***pause***, thus outputting the same frame of the animation. Whenever no animation is playing, it acts as a bypass, i.e., the ***input*** is directly routed to the output.

6.1.8 Nutty Plugins for each robot

In order to control a new robot, a specific NuttyOutput and BodyModel plugins may have to be developed. By fitting into Nutty Tracks as a plugin, they are loaded during execution, allowing the user to select which output (and robot) should be used. The BodyModel contains the robot's hierarchical structure, along with parameters that specify each joint's axis of rotation and limits. It also contains the code that translates and executes a generic Nutty Animation Buffer into the robot's control API. The referential used in Nutty matches the one used by OpenGL[8], and joint rotations are generally specified as floating point degree angles. The zero-pose (when all angles are set to zero) is considered to be having the robot facing straight, *neutral* and forward.

[8] `http://www.opengl.org/` (accessed January 12, 2019)

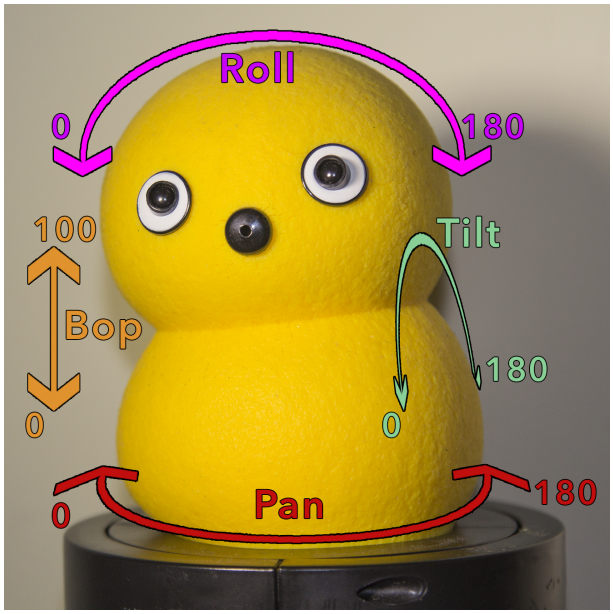


Figure 6.9: A real Keepon robot and range of execution of its Arduino-hacked servos.

Creating a Nutty Output Plugin for each new specific robot requires some work and expertise. However once it is created, it can be reused throughout all future projects. Moreover, any plugin for any robot can be shared with the community. The main advantage is, of course, that one might not need to develop the plugin for a robot if it is already available locally or in some public repository. The second major advantage is that in case of a robot's API upgrade, only this plugin needs to be replaced, while all the animation data and logic programming remains.

Nutty-Keepon example:

We take as example the development of the Nutty-Keepon BodyModel plugin. Keepon was chosen because it is a well-known and very simple robot. The first step was to understand how the Keepon is controlled in its own API. It is especially important to outline what units and reference system it uses. The Keepon used in our system was modified with controllable servos[9] and connects to a computer using an Arduino[10] board. Each servo is controlled by specifying a target position which is represented by an integer value ranging from 0 to 180 for the Pan, Roll and Tilt servos, and 0 to 100 for the Bop servo (Figure 6.9). For that we created an ArduinoOutput plug-in that could take the animated values and send them to the Arduino through a USB-to-Serial connection.

Because in Nutty Tracks animation is normally specified as degree angles, the Nutty-Keepon's robot representation sets all zero-angles (0°) to correspond to servo values of 90 for Pan, Roll and Tilt. As to Bop, it was kept as a value ranging from 0 to 100, representing a percentage. To test and verify this, a virtual version of the Keepon was made using Autodesk 3ds Max for modeling, and Unity3D[11] for real-time rendering (Figure 6.10). Nutty Tracks can also be used to control this virtual version by setting the used BodyModel to the Keepon (which is loaded from the Keepon plugin), while using as output a built-in frame streamer based on JSON[12], which send frames via TCP sockets from Nutty Tracks to some Nutty-JSON-frame client (in this case the Virtual-Keepon application), instead

[9] Keepon Hack: `http://hennyadmoni.com/keepon/` (accessed January 12, 2019)
[10] Arduino: `https://www.arduino.cc/` (accessed January 12, 2019)
[11] Unity3D: `http://unity.com/` (accessed January 12, 2019)
[12] JSON: `http://www.json.org/` (accessed January 12, 2019)

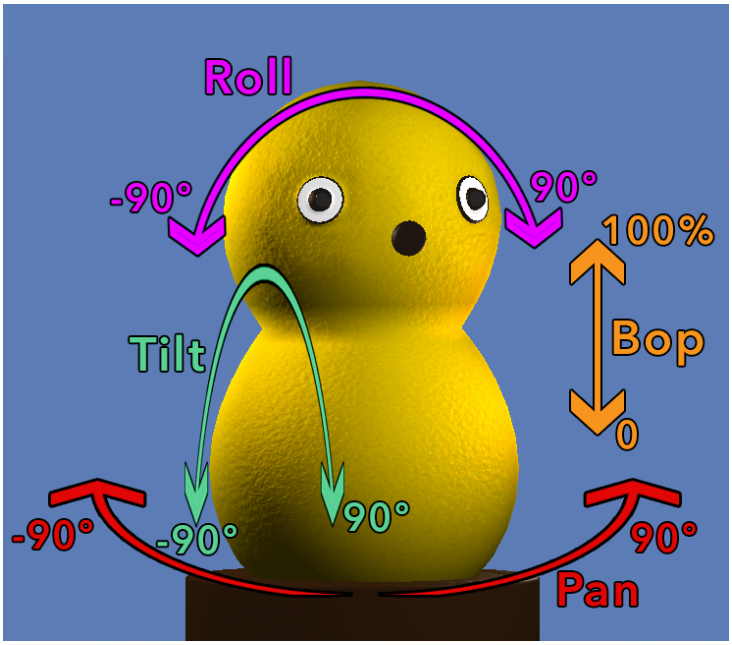


Figure 6.10: A screenshot of a Virtual-Keepon built in Unity3D with the angular range of movement of its degrees of freedom. Note the mapping from the real values in Figure 6.9 to the angular coordinates used in Nutty Tracks.

of outputting them to the real robot through the ArduinoOutput.

Animatable CGI model of the robot

We used Autodesk 3ds Max as a host animation software to load a different type of plug-in version of Nutty Tracks.

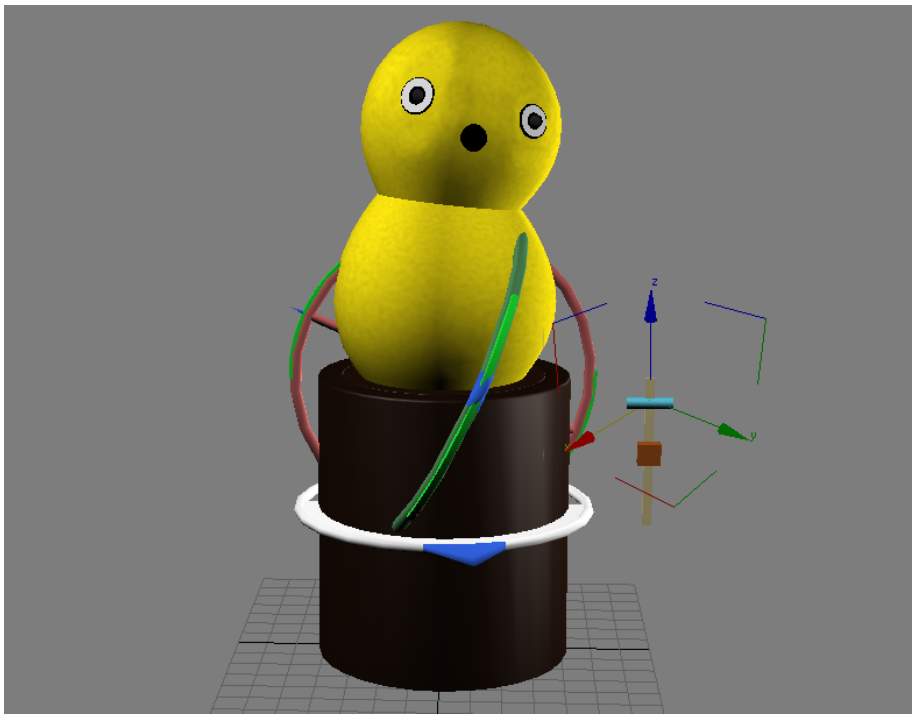


Figure 6.11: The animatable CGI Keepon robot in Autodesk 3ds Max.

This should not be confused with the Nutty plug-ins described on the previous section.

While the Nutty Tracks environment works by loading plug-ins, it can also become a plug-in itself, to a host animation software such as 3ds Max. That means that instead of Nutty Tracks being ran as a standalone application, it is programmed in Maxscript[13] to run as a 3ds Max plug-in.

Because 3ds Max already provides a complete set of modelling and animation tools, an expert animator was able to create a proper animation rig along with a 3d mesh of robot's embodiment with a deformable modifier in order to more accurately represent how the poses and animations look in the real robot (Figure 6.11). Also, because the actual Nutty Tracks engine is running within 3ds Max, the animator can animate while watching the result rendered on the real robot in real-time[14]. From 3ds Max, the Nutty Tracks plug-in allows to export both static postures and animations by baking the selected timeline to an animation file format that was specifically created to hold Nutty Tracks-loadable animations.

6.2 The Nutty Motion Filter (NMF)

When we take and adapt methods or techniques from CGI animation to robots, it is common to run into a particular pitfall regarding the generated motion signal. In CGI, objects can move around freely with no physical or kinematic constraints. As such it is *easy* to elaborate techniques that produce various kinds of motion, and to shape the motion into the expected end-results, following on simple interpolation techniques, and even using stepped motions (ones that are discontinuous). The fact is that virtual motion is, by nature, discrete, so it is always rendered in discontinuous steps, no matter how small those are, even if any derivatives are also calculated.

In robotics however, the motors are physical and therefore enforce certain kinematic constraints which, if not met, may result in errant and jerky motion. If one attempts to render a stepped motion on a robotic servo, the resulting movement will necessarily be continuous, moving from its initial position all the way to the final position, no matter how fast that motion might be. Even if we *try* to ignore it, inertia and other external forces will always be playing a part on the resulting motion. Therefore motion generated for robotic use must comply to different norms than the one produced for purely virtual applications.

In particular, a motion signal generated for a robot should be at least C^2 continuous, i.e, containing a derivative of order 2 or more. For servos and motors that power articulated structures in simpler robots, such a C^2 signal is typically enough (i.e., motion explicitly contains a limited acceleration component). In the case of more complex robots, and in particular for motion in space, the motion signal that drives e.g. the path of a robot must be C^3 (i.e., containing jerk, which is the derivative of acceleration), or even more (jounce is the 4th order derivative of motion, i.e., the derivative of jerk). Furthermore, in addition to angular limits, the mechanics and motors used will typically enforce a physical limit on each of the derivatives' value, which, if violated, may result in either physical damage, or in disorderly motion.

Figure 6.12 illustrates a simple motion signal with 3^{rd} order derivatives between two positions (-6 and 6), along with the limits of each of the derivatives. The signal input is referred to as the set-point, and in this case is a stepped signal, which is the most basic type of signal that can be used for motion control. Recall that a stepped signal

[13]Maxscript: `https://help.autodesk.com/view/3DSMAX/2017/ENU/?guid=__files_GUID_F039181A_C072_4469_A329_AE60FF7535E7_htm` (accessed January 12, 2019)

[14]Nutty Tracks with Keepon: `http://vimeo.com/155593476` (accessed January 12, 2019)

[14]`https://en.wikipedia.org/wiki/Jerk_(physics)` (accessed January 12, 2019)

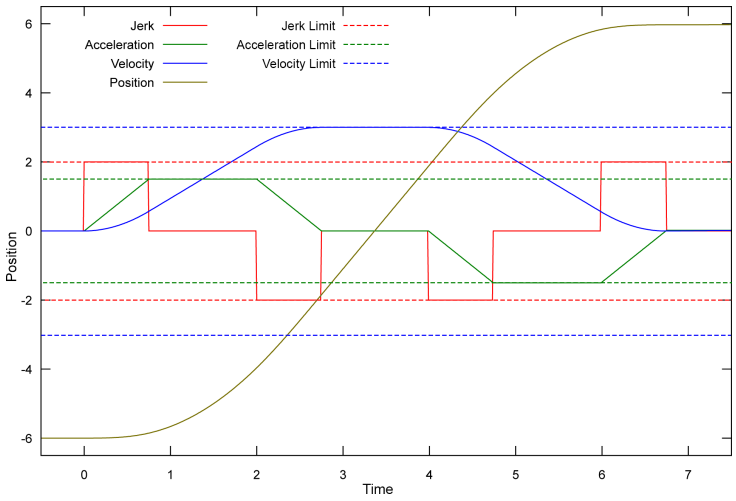


Figure 6.12: A diagram illustrating jerk, acceleration and velocity of a C^3 continuous motion signal that moves from -6 to 6.[14].

is very undesirable for robots, but is, in fact, the type of signal that is typically produced by a CGI application. A CGI application typically runs at a high frame-rate (e.g. 60Hz), which generates motion in small steps of $\frac{1}{60}$ seconds, which therefore becomes unnoticeable on screen. Therefore, it is generally not required to calculate all the derivatives that ensure the smoothness of the motion. If such a stepped signal is, however, applied to a robotic servo, it is likely to cause a lot of audible noise, along with jittery motion, given that, despite the stepped input, the motor will in fact have to move through the intermediate positions between the current one and the set-point, and that achieving that motion (velocity) will lead it to accelerate and de-accelerate between each step. Despite this issue, various authors in the field of HRI have actually used simple position-based motion controllers to control small, expressive robots motion controllers [104, 136, 22]. As long as the generated motion is *slow* enough, guaranteed to *seem* smooth, and produced at a rate of at least 30Hz, the jittery effect may become mitigated or at least acceptable.

Throughout our work we have, at times, took that same, simplistic approach. However in the long term, we feel the need for a proper motion filter that can be used as a bridge between any discontinuous, stepped motion such as the one typically produced in CGI, and the C^3 continuous motion required by robotics. Furthermore, because we place such a strong focus on the character animation aspect, we also considered it desirable to have a motion filter that would allow some kind of tweaking, in order to adapt the resulting motion not only to the robot's embodiment, but also to its character's traits, such as mood, personality or emotion.

The Nutty Motion Filter (NMF), presented in this section, solves that problem by:

- Taking as input any C^0 motion, in real-time, in a sample-by-sample basis (i.e, one sample at a time), in irregular time intervals;

- Outputting a C^1, C^2 or C^3 signal corresponding to the input one saturated by its velocity, acceleration and/or jerk limits, at a steady output rate;

- Providing character parameters that can be tweaked to shape the motion produced, namely in terms of smooth in/out, smooth damping, or if and how the motion should produce follow-through such as overshooting or controlled damped oscillation;

In addition to the contribution contained in this section, we have also made available an online simulator of the Nutty Motion Filter[15], which will allow the reader to further test and visualize the motion produced by the NMF, using various different parametrizations and trajectories.

6.2.1 NMF Definition

The Nutty Motion Filter is defined as the function $X(x(t), t(i), s)$, where $x(t) : \mathbb{R}_0^+ \rightarrow [P_{min}, P_{max}]$ is the motion signal history, i.e., the previous positions that were output from the filter. The parameters P_{min} and P_{max} represent the minimum and maximum values respectively. In e.g. a hinge joint, these would represent the angular limits of the joint. $x(0)$ is the initial position of the signal and must be specified. The function $t(i) : \mathbb{N}_0 \rightarrow \mathbb{R}_0^+$ (shortened to t_i) represents the time at each sample i, such that $0 \leq t_{i-1} < t_i$, and $t_i - t_{i-1} = \Delta t$, where Δt is a fixed time-step, calculated from the sample rate R, such that $\Delta t = \frac{1}{R}$. The sample rate should be chosen based on the requirements and capabilities of both the robotic and computational systems, and must be at least equal to the desired output rate. Therefore 30-100Hz are typically acceptable sample rates. Note that from this definition, i refers to the current sample, and therefore the current time is represented by t_i, while the time of the last sample is t_{i-1} and so on. The set-point s is the new target position, and is used to calculate the *induced velocity* $\dot{x}(t_i)$ as specified in Equation 6.1.

Finally, $x(t_i)$ represents the output that will be computed of the filter at the current time (not in the history yet), while s therefore represents the input. As such, $\dot{x}(t_i)$ must be calculated from s instead of $x(t_i)$.

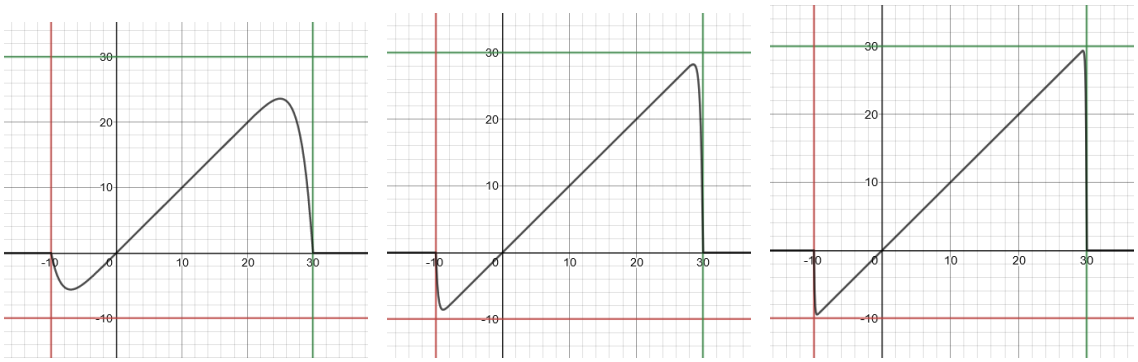
$$\dot{x}(t_k) = \begin{cases} \frac{s - x(t_{i-1})}{\Delta t} & , if\, k = i \\ \\ \frac{x(t_k) - x(t_{k-1})}{\Delta t} & otherwise \end{cases} \tag{6.1}$$

We start by dealing with the problem of limiting the position output of the motion using Equation 6.2. This output saturation function $\Omega(\dot{x}, x, P_{max}, P_{min}, \beta)$ takes the induced velocity and the current output position and prevents the induced velocity from moving the signal beyond the minimum and maximum values P_{max} and P_{min}. As seen in the equation, the induced velocity is reduced by Ω as the current output position approaches either the minimum or the maximum limits, while through the central portion of the motion range, the velocity is untouched. This approach differs from a hard limiter on the output (clamping), by providing some control over the motion before it reaches the position limit. Instead of causing a hard break, we can induce a de-acceleration up to a complete stop, when the output motion is approaching its limits. However the saturation is only applied when the induced velocity moves the signal *towards* the limit, i.e., if the current position is above its center (given by α), then the velocity is only saturated when it is positive, and if the current position is below the center, the velocity will only be saturated when it is negative. Without this remark, the velocity would become stuck at zero upon hitting the edge of the motion range, as this saturation function would not allow it to move away from the it. The β parameter controls the exponent of this de-acceleration, thus allowing to control how close to the limit the output is allowed to get before being saturated. As β increases, the saturation becomes more similar to a hard clamping function. The effect of different values for the β parameter is illustrated in Figure 6.13.

[15] `http://www.tiagoribeiro.pt/nutty/motionfilter.html` (accessed January 12, 2019)

$$\Omega(\dot{x}, x, P_{max}, P_{min}, \beta) = \begin{cases} \dot{x} \cdot \left(1 - \left(\frac{x - P_{min} - \alpha}{\alpha}\right)^{2\beta}\right), & if (x > \alpha \;\&\; \dot{x} > 0) \mid (x < \alpha \;\&\; \dot{x} < 0) \\ \dot{x}, & otherwise \end{cases} \tag{6.2}$$

$$\alpha = \frac{P_{max} - P_{min}}{2}$$



(a) Output of Ω using $\beta = 5$. (b) Output of Ω using $\beta = 20$. (c) Output of Ω using $\beta = 100$.

Figure 6.13: Comparison of the output saturation function Ω given the minimum and maximum limits of $[-10, 20]$, using three different exponents $\beta \in [5, 20, 100]$.

Additionally we define the derivative saturation function $\lambda(x) : \mathbb{R} \to \mathbb{R}$. This saturation function is individually applied to each of the motion signal's derivatives in order to enforce the physical limits that are imposed by the robot's embodiment, i.e., enforce that their absolute value does not exceed a given value limit k. This function may e.g. apply hard limits (described in Equation 6.3 for exemplification purposes), or provide smooth limits as described in Equation 6.4, where $k \in \mathbb{R}_0^+$ is the absolute limit value, such that $|\lambda(x, k)| \leq k, \forall x \in \mathbb{R}$. The latter one (Equation 6.4) was chosen for the NMF, as it progressively saturates the input signal while it is approaching its limit, in order not to induce a hard break when the limits are reached, thus alleviating the motion oscillation that would be introduced through the use of the hard limiter.

In fact, using the tanh-limiter, the real limit is never reached, given that the input would have to be infinite for it to happen ($\lim_{x \to \infty} tanh(x) = 1$). Being based on the hyperbolic tangent, this saturation function produces a signal that is also continuously differentiable (contrary to λ', which is C^0).

$$\lambda'(x, k) = min(k, max(-k, x)) \tag{6.3}$$

$$\lambda(x, k) = \frac{k}{2} \cdot \tanh(x / \frac{k}{2}) \tag{6.4}$$

Using the equations of motion directly to calculate the final motion upon saturating the signal would still lead, however, to some oscillation, especially in our case, where the filter digests set-points in real-time, unknowingly of when the set-point and motion will come to a rest. The length and amplitude of such oscillation would depend on

the filter order and the physical limit values.

We have however devised an additional velocity transfer function $H(v)$ (also referred to as stabilization function), presented in equation 6.5 that softly brings the motion to a rest once it starts to approach its latest given set-point. The transfer function is applied to the *saturated induced velocity*. This stabilization function uses two hyper-parameters $\{\sigma, \rho\}$, representing *smoothness* and *responsiveness* respectively, that allow to tweak the filter, changing how quickly it responds and how much it is allowed to oscillate. We call these the *character parameters*, as different configurations for them will shape the motion differently. As such we argue that they can be used to model different character traits, even when the same physical limits are enforced. The *smoothness* parameter σ will ease out the oscillations. However, depending on other filter parameters such as the physical limits, fully easing out might become too slow and make the motion seem too muddy and flat. That is where the *responsiveness* parameter ρ comes in, which allows to precipitate the easing out, so that it may still be smooth, but faster, and thus, more responsive. While these concepts of *smoothness* and *responsiveness* may seem antagonistic in the context of a motion signal, they will be better explained further through illustrative examples.

$$\mathrm{H}(v) = \frac{v}{2} \cdot \left(\tanh\left(\left(\frac{|v|}{1-\rho} \right)^{1-\sigma} - \pi \right) + 1 \right), 0 \leq \sigma \leq 1, 0 \leq \rho < 1 \tag{6.5}$$

Based in the output saturation function Ω from Equation 6.2, on the derivative saturation function λ from Equation 6.4, and the stabilization function H from Equation 6.5, we present below the final equations for either a C^3, C^2 or a C^1 NMF filter. Recall also the definition of $\dot{x}(t_k)$ from Equation 6.1. Higher order filters can also be inferred, based on these equations.

Equation 6.6 contains the C^3, or 3^{rd} order NMF variant, defined as $\chi_3(x, t, s)$.

$$\begin{aligned}
\chi_3(x, t_i) &= x(t_{i-1}) + \lambda(\psi_3(x, t_i), velocity_limit) \\
\psi_3(x, t_i) &= \dot{x}(t_{i-1}) + \lambda(\frac{\xi(x, t_i) - \dot{x}(t_{i-1})}{\Delta t}, acceleration_limit) \\
\xi(x, t_i) &= \ddot{x}(t_{i-1}) + \lambda(\frac{\frac{v \cdot \mathrm{H}(v) - \dot{x}(t_{i-1})}{\Delta t} - \ddot{x}(t_{i-1})}{\Delta t}, jerk_limit) \\
v &= \Omega(\dot{x}(t_i), x(t_{i-1}), P_{max}, P_{min}, \beta)
\end{aligned} \tag{6.6}$$

Equation 6.7 contains the C^2, or 2^{nd} order NMF variant, defined as $\chi_2(x, t, s)$.

$$\begin{aligned}
\chi_2(x, t_i) &= x(t_{i-1}) + \lambda(\psi_2(x, t_i), velocity_limit) \\
\psi_2(x, t_i) &= \dot{x}(t_{i-1}) + \lambda(\frac{v \cdot \mathrm{H}(v) - \dot{x}(t_{i-1})}{\Delta t}, acceleration_limit) \\
v &= \Omega(\dot{x}(t_i), x(t_{i-1}), P_{max}, P_{min}, \beta)
\end{aligned} \tag{6.7}$$

Finally, equation 6.8 contains the C^1, or 1^{st} order NMF variant, defined as $\chi_1(x, t, s)$.

$$\begin{aligned}
\chi_1(x, t_i) &= x(t_{i-1}) + \lambda(v \cdot \mathrm{H}(v), velocity_limit) \\
v &= \Omega(\dot{x}(t_i), x(t_{i-1}), P_{max}, P_{min}, \beta)
\end{aligned} \tag{6.8}$$

6.2.2 Usage and Examples

In order to demonstrate and exemplify the usage of the NMF, we will be defining a set of example filters and example input signals, for which we will then illustrate the transfer function of the example filters along with the output that results from applying them to a given example input signal. Throughout this section, the graphs presented show the position output that is produced by incrementally calculating the filter at each time-step $t \in [0, T_{max}]$, at a 60Hz sample rate (steps of $\frac{1}{60}s$). The figures also display the resulting velocity, acceleration and jerk (when applied). Recall that the filter is calculated on a per-sample basis, and has no look-ahead information on the trajectory (which allows it to be used in real-time applications). Therefore on each moment, the filter knows only what is the current set-point, and what were the previous output positions and derivatives, thus the graphs presented are accurately representative of the output that would be produced by each filter in a real-time application.

Example Filters

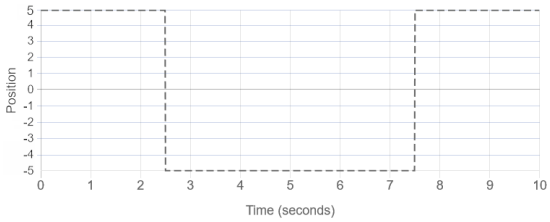
We start by defining a set of example filters in Table 6.1, organized into groups (*Regular*, A, B C, D & E) based on their hyperparameters definition, i.e., the set of character parameters and physical limits. Within each group, there are 1^{st}, 2^{nd} or 3^{rd} order variants, and either may use the Tanh limiter function (Equation 6.4), or the Non-tanh limiter function (Equation 6.3). Each example filter is designated by a name in the format X^{β}_{α}, where α is the order of the filter, and β is its hyperparameter group, followed by the symbol λ' in case it does not use the Tanh-limiter (the λ symbol is omitted otherwise). Although we chose and strongly recommend to use the Tanh-limiter with the NMF, we will be demonstrating both versions in order to illustrate how it impacts the output of the filter. The *Regular* filter represents one in which our stabilizing transfer function H is bypassed, and therefore it contains no character parameters.

Example Input Signals

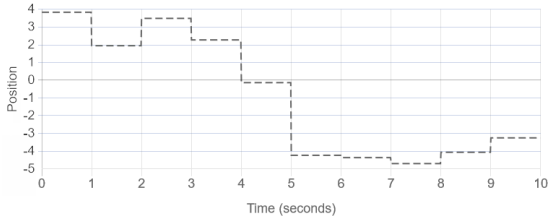
Figure 6.14 shows three different input trajectories that were used to demonstrate the filter across different conditions. The first input Φ_L illustrates a very simple linear trajectory in which the set-point for the position is moved instantaneously from 5 to -5 at time $t = 2.5$, and then back to 5 at $t = 7.5$.

Hyperparam Group	Filter Name	Limiter	Order	Smooth	Responsive	Kinematic Limits Vel.	Accel.	Jerk
Regular	$W_3^{\lambda'}$	Non-tanh	3^{rd}	-	-	20	100	10 000
	W_3	Tanh						
A *slow & smooth*	$X_3^{A\lambda'}$	Non-tanh	3^{rd}	1.0	1.0	20	100	10 000
	X_3^A	Tanh						
	X_2^A	Tanh	2^{nd}					
	X_1^A	Tanh	1^{st}					
B *slow & vivid*	$X_3^{B\lambda'}$	Non-tanh	3^{rd}	0.1	0.0			
	X_3^B	Tanh						
	X_2^B	Tanh	2^{nd}					
	X_1^B	Tanh	1^{st}					
C *fast & vivid*	$X_3^{C\lambda'}$	Non-tanh	3^{rd}	0.1	0.0	90	700	50 000
	X_3^C	Tanh						
	X_2^C	Tanh	2^{nd}					
	X_1^C	Tanh	1^{st}					
D *fast & smooth*	$X_3^{D\lambda'}$	Non-tanh	3^{rd}	0.95	1.0			
	X_3^D	Tanh						
	X_2^D	Tanh	2^{nd}					
	X_1^D	Tanh	1^{st}					
E *fast & smoother*	X_3^E	Tanh	3^{rd}	0.95	0.2			

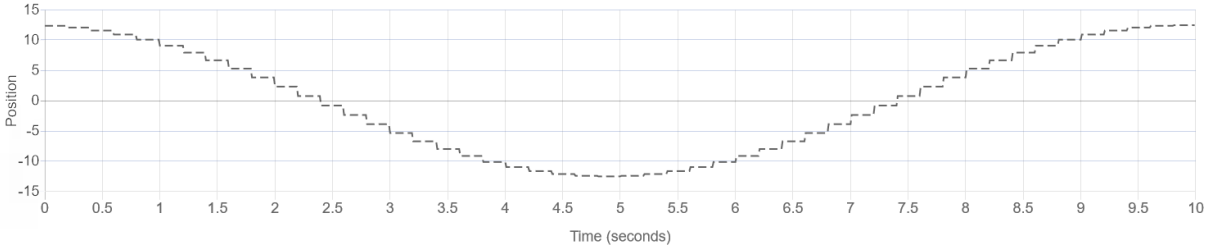
Table 6.1: Definition of filters used in the examples throughout the current section. As a mnemonic, the subscript of the filter name represents its order (1, 2 or 3), while the superscript represents its hyperparameters group (3 distinct sets A, B C D & E), along with an additional λ' in case it uses the Non-tanh limiter. Additionally the Non-tanh filters were shaded to improve readability.



(a) Example *linear* input Φ_L as a highly discontinuous input.



(b) Example input Φ_R as a *random* input.



(c) Example input Φ_C as a stepped *circular* trajectory (in one dimension only).

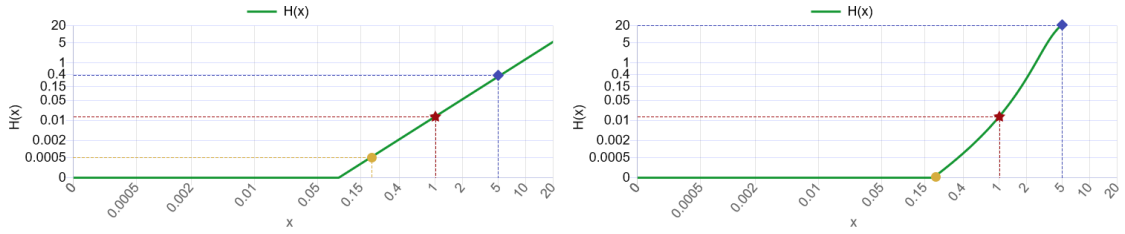
Figure 6.14: Three example input trajectories, used to demonstrate the use of the Nutty Motion Filter.

It is intended to show how a filter responds to a large change in the input signal. The second input Φ_R illustrates a case in which the trajectory set-point is randomly adjusted at each second. It is intended to show how a filter responds both to small and large changes in the input signal. The third input Φ_C illustrates a case in which the final trajectory was a circle. In this case we see only one of the two dimensions of the circular trajectory. This trajectory

however, was discretized into 50 points, thus producing a small step at every 0.2 seconds for the 10-second long trajectory. It is intended to emulate what in CGI would be seen as a smooth signal, but is, however, a stepped input.

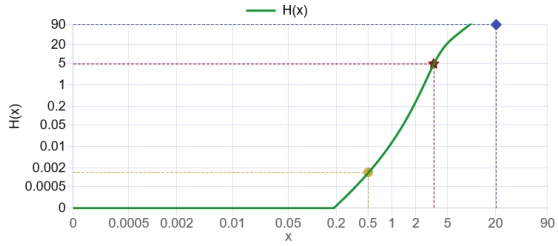
Example Filters' Transfer Function Response

Figure 6.15 contains four plots that illustrate the transfer function for different hyperparameter groups. The top graphs refer to groups A and B, which both share one set of *slower* physical limits, while the lower graphs refer to groups C and D, which share the other set of *faster* physical limits. The transfer function of B and C are actually equal (same character parameters $\{\sigma, \rho\}$), however they consider different physical limits. This is reflected in the plots, as each shows the output of $H(x)$ being $x \in [0, \mathit{VelocityLimit}]$. As such, $x \in [0, 20]$ for groups A and B, while $x \in [0, 90]$ for C and D.

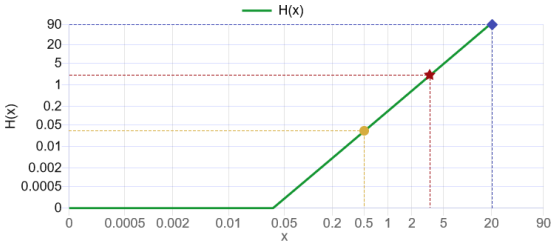


(a) Transfer function of group $A(\sigma = 1.0, \rho = 1.0)$.

(b) Transfer function of group $B(\sigma = 0.1, \rho = 0.0)$.



(c) Transfer function of group $C(\sigma = 0.1, \rho = 0.0)$.



(d) Transfer function of group $D(\sigma = 0.95, \rho = 1.0)$.

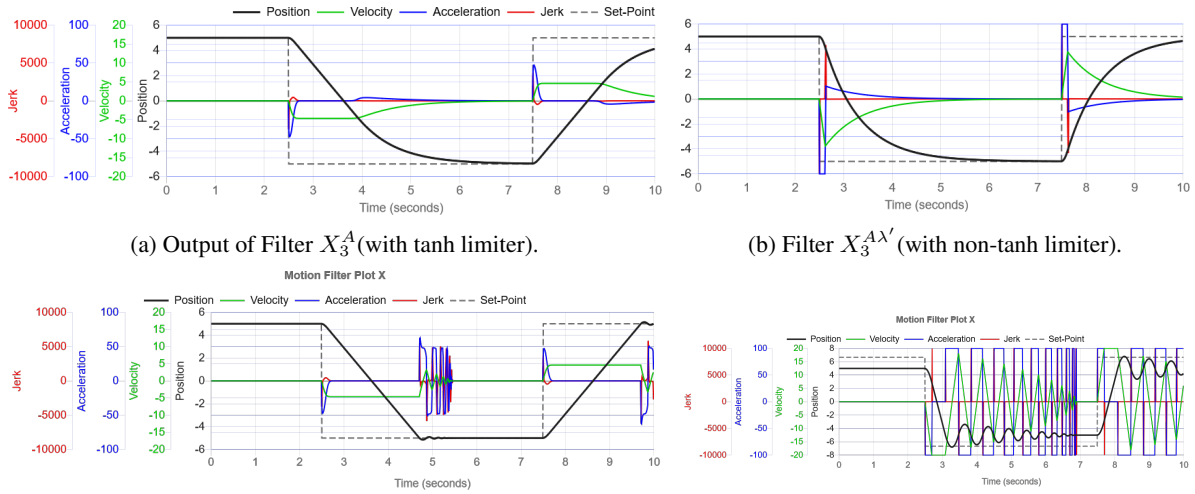
Figure 6.15: Plots of the different transfer functions specified by each hyperparameter group A, B C & D. The domain of these graphs is $x \in [0, \mathit{VelocityLimit}]$, thus corresponding to $[0, 20]$ for A and B, and to $[0, 90]$ for C and D. Also note that filters in groups B and C share the same character parameters, which results in the same transfer function, confined only to a different domain. All graphs include three distinct points in the x axis as an aid to interpret how they differ.

Output Examples using the Nutty Motion Filter

Through this section we will be comparing various graphs in order to illustrate how the filter's response changes both given a different set of character parameters, physical limits and input trajectory. We will also take the opportunity to demonstrate how the tanh-based limiter differs from a non-tanh-based limiter, and even to demonstrate how a given filter would behave without the use of our stabilizer function.

Using the simpler Φ_L input, Figure 6.16 shows on the top left (a), the output of X_3^A, with the tanh-limiter, in comparison with $X_3^{A\lambda'}$, on the top right (b), which uses the non-tanh limiter. The hyperparameters for this filter group (A) make it what we would call a *slow* character, given that the motion takes some time to respond, and then again to become fully stationary when the set-point has rested.

Observing the derivatives' curves, the difference between the two variants becomes clear. In the first case, they never hit their maximum value, and are all smooth, as they are based on the hyperbolic tangent. The output of the



(a) Output of Filter X_3^A(with tanh limiter).

(b) Filter $X_3^{A\lambda'}$(with non-tanh limiter).

(c) Output of Filter $W_3(H(x) = x$, with tanh limiter).

(d) Filter $W_3^{\lambda'}(H(x) = x$, with non-tanh limiter).

Figure 6.16: Comparison of the effect of the tanh-limiter on the output the Nutty Motion Filter. **Top row**: The output of filter X_3^A(a) compared to filter $X_3^{A\lambda'}$(b). **Bottom row**: The output produced by the NMF equations if the transfer function was bypassed, i.e., making $H(x) = x$, while using the tanh-based limiter on the left (c), and a non-tanh based limiter on the right (d). All four plots are produced from the *simple* input signal Φ_L.

tanh-based variant becomes, however, slower, because the velocity was in general, confined to a lower value than in the non-tanh version. This illustrates the implications of the tanh-limiter on the output motion - the system will, in general, produce an output that is slower than physically allowed, by creating what we call a *headroom*[16], that allows to smoothly accommodate cases in which a very large change is induced by the input signal. Due to this feature of the tanh-limiter, we differentiate the maximum value of a derivative, from its maximum sustained value.

Taking as example the first derivative, that maximum sustained velocity will be the absolute value at which the velocity tends to hold as constant (about 5 in Figure 6.16a), in contrast with the real maximum absolute velocity (20 in the same Figure), which is the hyperparameter used to parameterize the filter, and includes the headroom.

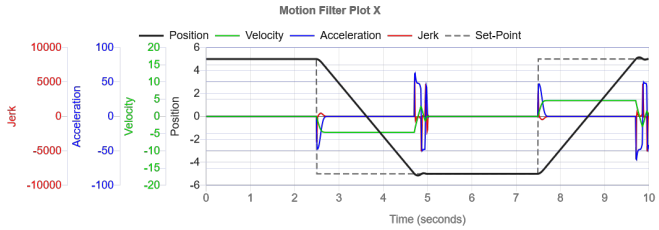
For a reference, on the bottom row we also present the output of the signal that would be produced if we bypassed our transfer function, i.e., making $H(x) = x$. In this case we see on the bottom left (c) that the signal actually responds quickly with some slight oscillation when using the tanh-based limiter, and results in severe oscillation when using the non-tanh limiter (bottom right (d)). Without the transfer function, the output only starts to stabilize after it has reached the set-point. Therefore, we can observe, on the left-side, that the W_3 filter accelerates until it reaches a maximum velocity and continues that trajectory until it reaches the set-point. Only then does it attempt to stabilize the output. Because it was going too fast and even overshot it, some oscillation was produced, which however, was mitigated by the use of the tanh-limiter. However, on the X_3^Afilter we see that the output starts to de-accelerate much earlier in order to allow the output to stabilize smoothly without oscillating. These graphs therefore show that although bypassing our transfer function is a possibility, we would have no control over how fast or smoothly the filter responded, except by tweaking the physical limits, which would be an undesirable requirement.

Figure 6.17 show a comparison of the same input signal using the 3^{rd} order variants of the B, C and D filter groups, again with both their tanh and non-tanh based variants. In this set of examples we have varied the character

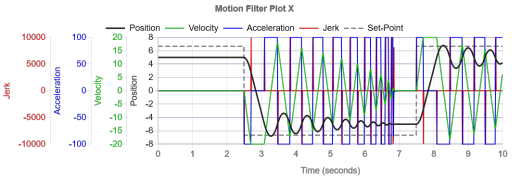
[16]Borrowed from the concept of headroom used in digital audio. `https://en.wikipedia.org/wiki/Headroom_(audio_signal_processing)` (accessed January 12, 2019)

parameters and physical limits. When we parameterize the filter to provide a more vivid response as in filters X_3^B(Fig. 6.17a) and X_3^C(Fig. 6.17c), we do encounter a slight oscillation effect. This oscillation is, however, introduced due to our choice of the parameters, and is therefore a controlled oscillation, i.e., one that would allow the character to exhibit some overshooting and follow-through animation, in order to convey a sense of weight and inertia. If that oscillation is fully undesirable, we may parameterize the filter further to produce a fast and steady response, as seen in filter X_3^D(Fig. 6.17e).

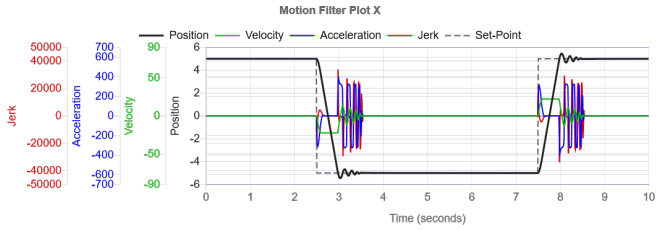
On the right side of the figure, the non-tanh limiter shows a faster response in comparison with the tanh-based limiter version, but then after the output overshoots, it struggles to stabilize the signal quickly, thus leading to the oscillation. It becomes clearer why the tanh-limiter became our choice for the NMF as it allows us to tweak the *shape* of the output signal (as seen on the left), without introducing that *undesirable* oscillation.



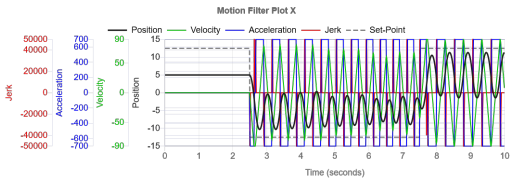
(a) Output of Filter X_3^B(with tanh limiter).



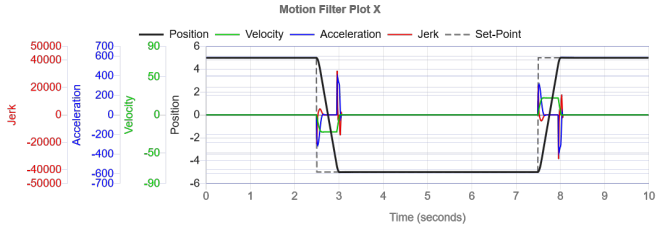
(b) Output of Filter $X_3^{B\lambda'}$(with non-tanh limiter).



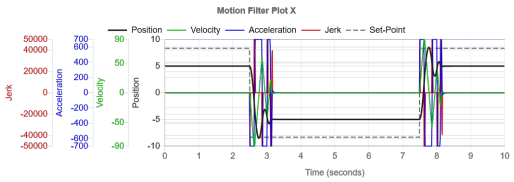
(c) Output of Filter X_3^C(with tanh limiter).



(d) Output of Filter $X_3^{C\lambda'}$(with non-tanh limiter).



(e) Output of Filter X_3^D(with tanh limiter).



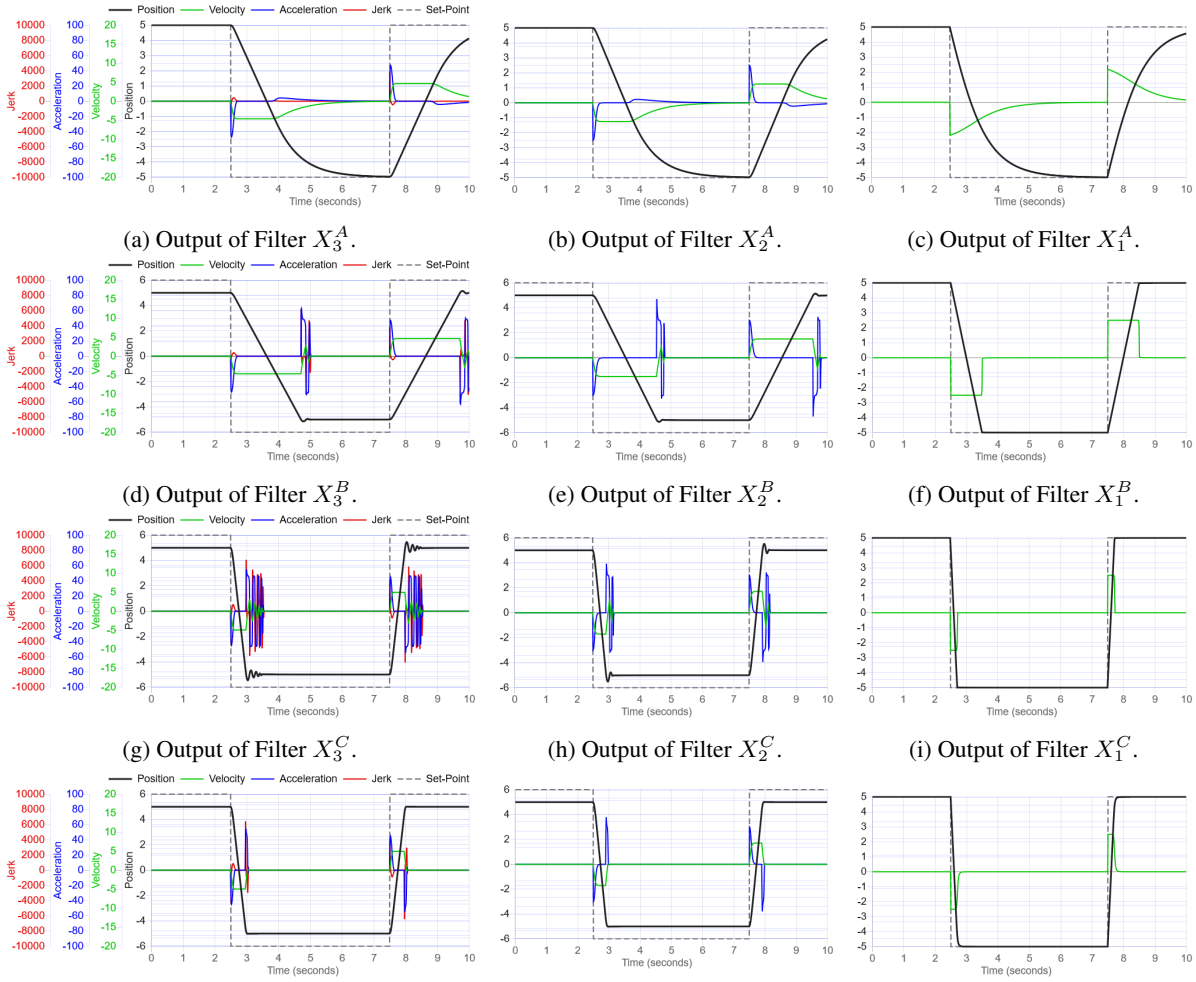
(f) Output of Filter $X_3^{D\lambda'}$(with non-tanh limiter).

Figure 6.17: Output of the 3^{rd} order filter of groups B, C and D, using the *simple* input signal Φ_L.

In Figure 6.18 we can see a comparison between three different filter orders for each of the hyperparameter groups A, B, C and D using the same simple Phi_L input. This figure allows to verify that the filter's response shape remains consistent across different orders, given the same character parameters and physical limits. What also observe that the maximum sustained velocity increases as the filter order decreases, thus suggesting that we should use the least order filter that the embodiment allows, in order to take the best advantage of the embodiment's kinematic capabilities.

In order to better conclude about how various filter parameters respond to different trajectories, we include some additional sets of plots.

Figure 6.19 shows the plot for each hyperparameter group A, B, C and D, using the *random* input trajectory



(a) Output of Filter X_3^A. (b) Output of Filter X_2^A. (c) Output of Filter X_1^A.

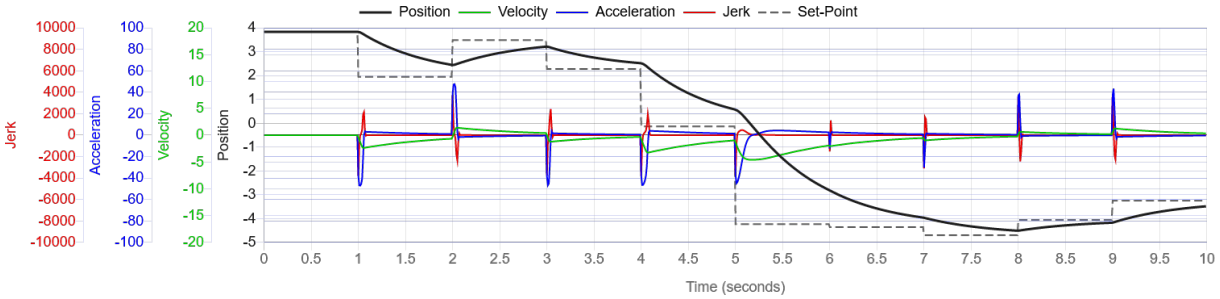
(d) Output of Filter X_3^B. (e) Output of Filter X_2^B. (f) Output of Filter X_1^B.

(g) Output of Filter X_3^C. (h) Output of Filter X_2^C. (i) Output of Filter X_1^C.

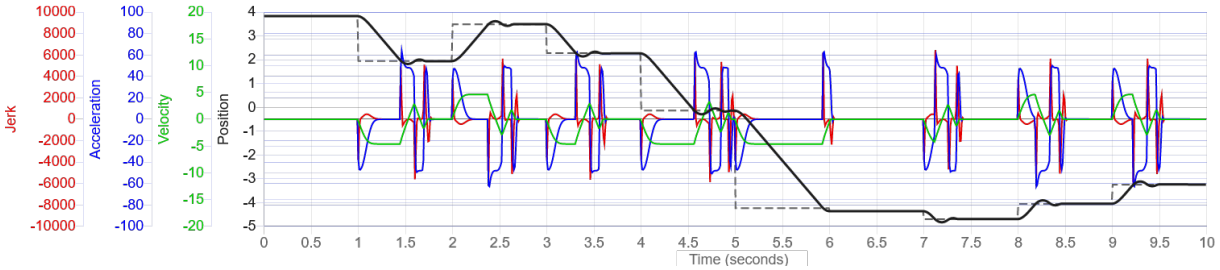
(j) Output of Filter X_3^D. (k) Output of Filter X_2^D. (l) Output of Filter X_1^D.

Figure 6.18: Comparison of the 3^{rd}, 2^{nd} and 1^{st} order filters for hyperparameter groups A (first row), B (second row) C (third row) and D (last row), using the *simple* input signal Φ_L.

Φ_R. In this trajectory we see how each filter responds both to large and small set-point changes. Filter $X_3^A$is too slow to actually reach the set-points through half of the trajectory. Filter X_3^Bperforms better there, but adds some overshooting which can be seen as a small bump. Filter $X_3^C$is too loose, and although it reaches the set-points quickly, it introduces not only overshooting but also some oscillation. Filter X_3^Dillustrates what we consider as a fast and steady response, with nearly no overshooting.

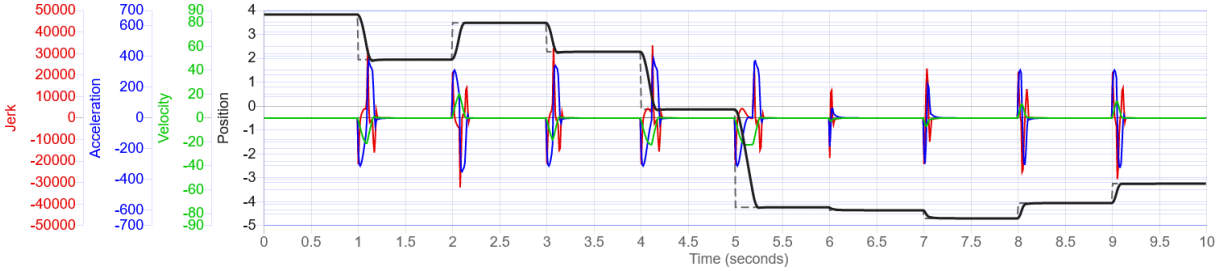


(a) Output of Filter X_3^A using the *random* input signal Φ_R.



(b) Output of Filter X_3^B using the *random* input signal Φ_R.

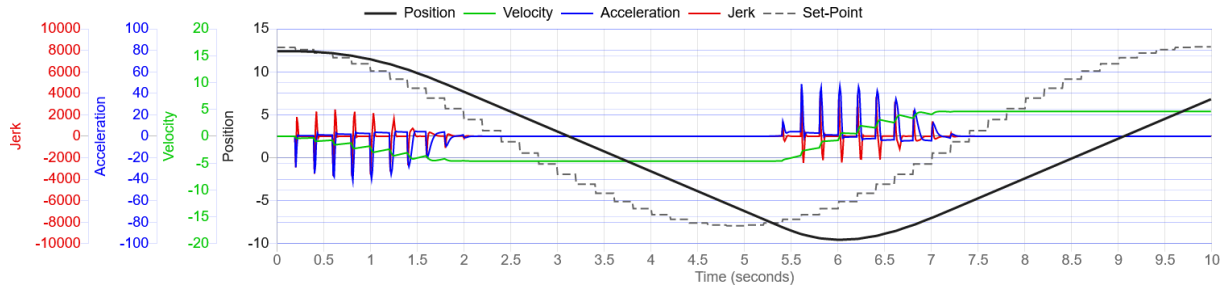
(c) Output of Filter X_3^C using the *random* input signal Φ_R.



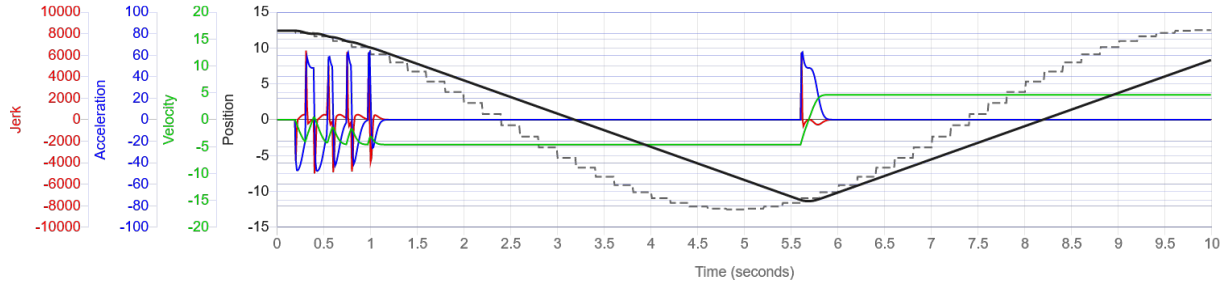
(d) Output of Filter X_3^D using the *random* input signal Φ_R.

Figure 6.19: Comparison of hyperparameter groups A, B, C and D, using the *random* input signal Φ_R.

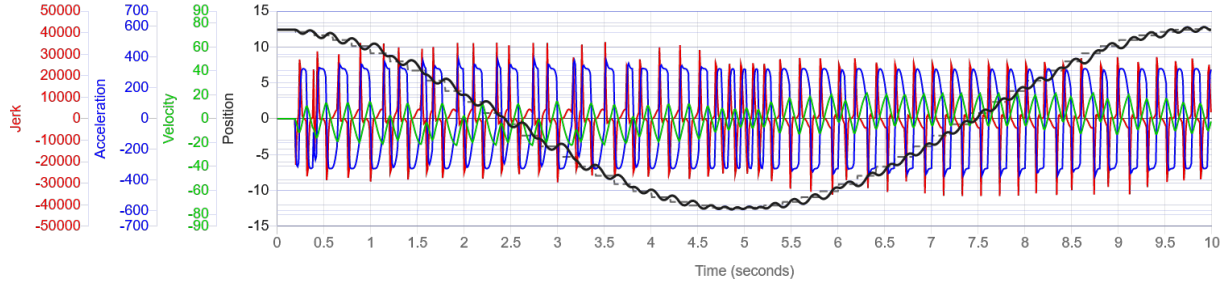
Finally, Figure 6.20 shows the plot for each hyperparameter group A, B, C, D and additionally E, using the *circular* input trajectory Φ_C. These plots how each filter responds to an input signal that is continuously changing in small steps - which in some cases is actually a challenge. Most of the remarks from the previous set of plots (Figure 6.19) also apply here. However we have added the additional X_3^E filter as a version of X_3^D with a lower *responsive* parameter. The purpose of this final set of plots is to show that not only will the selected character parameters depend on the intended shape and smoothness of the signal response, but must also consider how the input signal will be generated and fed to the filter.



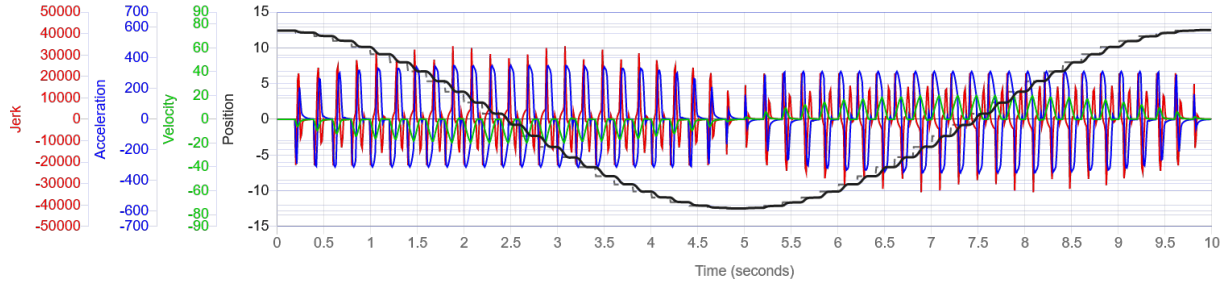
(a) Output of Filter X_3^A using the *circular* input signal Φ_C.



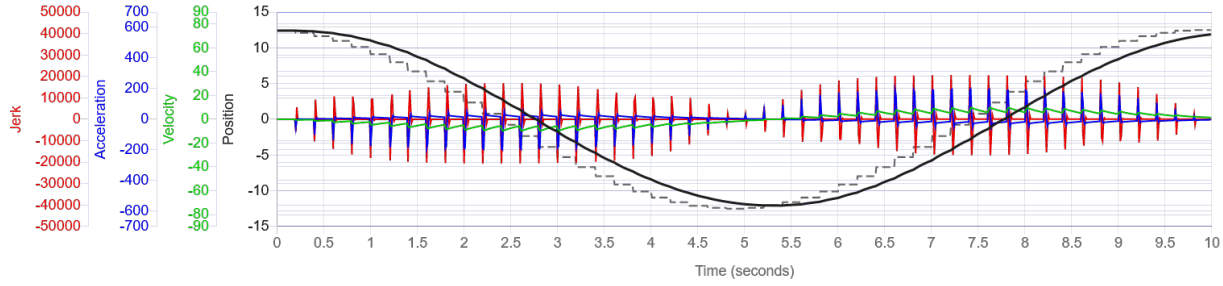
(b) Output of Filter X_3^B using the *circular* input signal Φ_C.



(c) Output of Filter X_3^C using the *circular* input signal Φ_C.



(d) Output of Filter X_3^D using the *circular* input signal Φ_C.



(e) Output of Filter X_3^E using the *circular* input signal Φ_C.

Figure 6.20: Comparison of the hyperparameter groups A, B, C, D and E, using the *circular* input signal Φ_C.

6.2.3 Comments and Remarks

Throughout this section we have presented the Nutty Motion Filter, which uses a composition of various transfer functions to allow an open-loop motion control system to smoothly interpolate and stabilize a given input signal even when that signal is highly discontinuous. Various choices were made which however, should not fully enclose this filter as a sealed solution. We have described the filter as various separate components, although one may implement a more optimized version by combining them in a different way. In addition, various options may be considered regarding e.g. the limiter function. While the tanh-based limiter became our choice, it was a balanced decision, i.e., it provides steady, controllable results, with no tweaking required. For other particular applications, it is important to emphasize that one may choose, explore and develop other types of limiter functions that better suite their requirements.

A final and important remark that must not be left uncommented is on the use of limiters with mobile robots that are operating in 2D trajectories. While it may seem obvious for a person with a strong background in robotics, we want to clarify that in cases where the X and Y directions are controlled separately, they must typically be limited together, in order to provide a limiter on the robot's actual linear velocity (which is made up of x and y) This means that given a 2D vector $[\dot{x}, \dot{y}]$, representing the velocities in both dimensions, one would have to calculate the magnitude of the resulting linear velocity vector, apply the limiter to that resulting vector only, and then proportionally saturate each of the two components, in order to limit both dimensions in a way that the resulting linear velocity does not exceed the specified limit. The same principle would apply for any other degrees of freedom that jointly operate to perform 2D motion (or even more).

6.3 ERIK - Expressive Robotics Inverse Kinematics

One common and basic social robot behaviour that we take as an example is face-tracking, which directs a robot's gazing towards the face of the human with whom it is interacting. For a simple robot, e.g., neck with two DoFs, it is easy to implement face-tracking by extracting a vertical and horizontal angle from the system's perception components (e.g. camera, Microsoft Kinect). These two angular components can directly control the two individual motors of the robot's neck. However this is a very limited conception of face-tracking behaviour, and also a very limited form of gaze control in general. Gazing behaviour can also be compound, by featuring not only face-tracking, but also used deictically towards surrounding objects, and in conjunction with other static or motive expressions (e.g. posture of engagement, nodding in agreement). Furthermore, one must consider that compound gazing behaviour should also be adopted for use with complex embodiments that feature multi-DoF necks, such as industrial manipulators, by considering the manipulator's endpoint to take on the expressive role of being the character's head, i.e. taking inspiration on an animated snake.

Within the goal of this thesis, we are focusing on the possibility of animating a robot such as an expressive manipulator, containing a chain of an arbitrary number of unidimensional degrees of freedom.

Animating such a robot within an interaction with humans would pose at least the requirements of having it able to simultaneously be expressive, while tracking an orientation constraint (e.g. gaze target). This is not however, a trivial problem. While it would be possible to control an expressive posture on the robot through Forward Kinematics (FK), and while an inverse kinematics (IK) algorithm could separately be used to provide it with gaze-tracking

ability, the blending of both an expressive posture (FK) and an IK solution is not possible using a simple operator.

This becomes especially aggravated by the fact that a robot has physical constraints that limit its range of movement, and that the final solution should be in joint-angle space (and not in position-coordinates as can be done with techniques from CGI). Another burden is to ensure that our tools and algorithms would still be able to provide solutions for any type of embodiment, and not just for the one we use.

This problem has posed as a hindrance to autonomous social robots' ability to properly express their underlying intention, when their actions are performed by an articulated structure. We therefore outline a solution to it by creating an animation algorithm capable of blending FK and IK. Such animation algorithm would be solving for two constraints which in most cases, are not simultaneously satisfiable: the expressive posture of the robot, i.e. the configuration of angles for each degree-of-freedom (DoF) that results in a given posture; and the global orientation of the endpoint node, i.e. the configuration of angles for each DoF such that the endpoint node faces towards a given orientation (in world coordinates). Moreover, such algorithm must be fast in order to provide a responsive interaction with humans, the resulting motion must seem smooth and continuous in order to exhibit naturalness, and we also want it to be extensible and adaptable to other embodiments.

Given this challenge, we have defined the following features to be met by such algorithm:

- Simultaneously solve for a given posture configuration and endpoint orientation;
- Prioritize the endpoint orientation constraint by allowing for distortion of the expressive posture;
- Keep the distortion of the target expressive posture as *minimal* as possible. Our definition of *minimal* is that although the expressive posture will not maintain its original form, it should be distorted in a way that its intended expressive meaning still holds;
- Fast solving for real-time applications (e.g. \50 solutions per second on a common computer);
- Per-frame solving to allow continuous tracking of subjects (i.e. in contrast to full trajectory pre-planning);
- Provide continuous solutions, i.e. subsequent solutions should minimize jitter or broken motion in regards to previous solutions;
- Support any configuration of kinematic chains, containing an arbitrary number of single-DoF nodes, rotating each about an arbitrary axis with fixed angular limits;
- Be extensible for full-body solutions, i.e. solving multiple sub-chains for multiple endpoints (e.g. humanoid embodiment with two individually controlled arms);

Our belief is that a solution to this problem will allow to create social robots that are more capable of conveying their social intentions and overall motivation to human interactors, while performing other tasks such as gazing or pointing.

This section presents and describes ERIK, a heuristic inverse kinematics technique that allows a virtual or robotic character with an arbitrary articulated embodiment to convey and hold a given expressive posture while facing a given direction during an interaction. It is able to provide many solutions per second using a standard computer, allowing it to be used for interactive applications.

The effort to design expressive behaviours for interactive characters using ERIK is also minimal. Animators can design single front-facing postures for any given embodiment, which are used as input to the algorithm, with no

pre-computing or offline training required for any new posture or embodiment. The algorithm is then able to take that posture and warp it in real-time, so that a given end-point is facing any given orientation, while respecting the embodiment's kinematic constraints, and while attempting as best to hold the overall shape of the posture.

Furthermore, ERIK was also developed to support its use with robots. As such, its output consists of a list of rotation angles, one for each joint, which can be used either in virtual or robotic applications. The solver computes on a *per-frame* basis in order to easily fit into a typical animation cycle, i.e., it produces one full-body solution at a time, and not a pre-planned motion trajectory.

Figure 6.21 illustrates the work-flow of a Nutty-ERIK system. ERIK can either be used as a component of an Interactive Application such as a game, VR/AR application, or a robotic AI, or alternatively, it can be used as a plug-in for an animation authoring tool. In the latter case, due to its real-time nature, it allows artists to creatively explore the design of expressive postures for real robots in real-time, and directly in the real, physical embodiments. This animator-inclusive workflow follows on the work initially proposed by [22, 121].

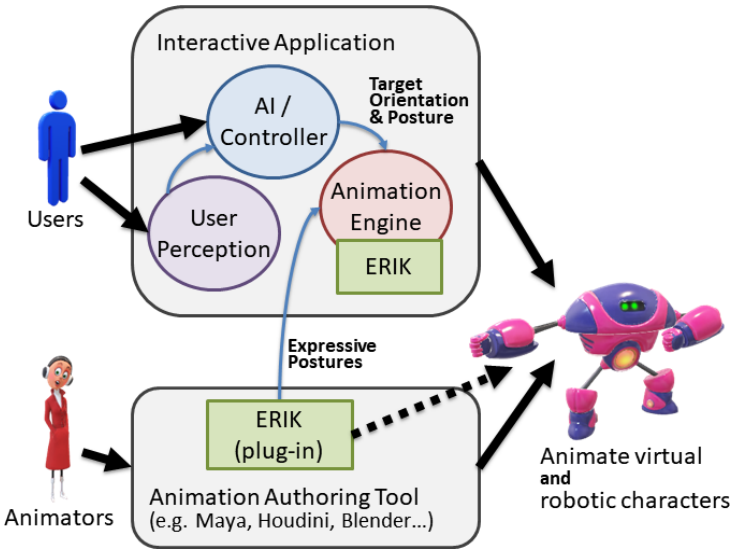


Figure 6.21: The ERIK workflow, illustrated as a particular version of the Nutty Pipeline . Animators can create expressive postures using typical animation tools. Those expressive postures can be selected by an AI or character controller in an Interactive Application to drive an animation engine which, through the use of ERIK, is able to perform the selected posture towards a selected orientation (e.g. from a user-perception component). Alternatively (dotted arrow), ERIK can also be used by a plug-in for the animation tool, to allow live authoring of postures on real robots i.e., the animators are able to test the result of the postures in a given robot directly and in real-time.

ERIK can be used to create tools and character animation engines directed at animation artists, so that they can take a stronger role in the development of autonomous, interactive, computer-animated characters, be them virtual or robotic. By bringing the artists closer to the AI - or the AI closer to the artists - we expect ERIK to prove as a strong technological contribution for the creation of better and more life-like interactive characters, in particular within immersive and emergent applications such as the ones based on VR, AR and robotics.

In many cases, the algorithm will be solving for two constraints that are not simultaneously satisfiable: the expressive posture of the character, i.e. the configuration of angles for each DoF that results in the given posture; and the global orientation of the end-point node, i.e. the configuration of angles for each DoF such that the end-point node faces towards a given orientation (in world coordinates). This means that depending on the character's embodiment, on the target posture, and on the target orientation, the resulting pose may either fully satisfy both goals, or fully

satisfy the orientation goal while partially satisfying the posture goal. Due to mechanical limitations, there will be many cases in which it is physically impossibly to solve both goals. Given that ERIK aims at autonomous characters that interact with humans, it will prefer a pose that complies with the given orientation target (where a human is expected to be), while allowing the posture to fall short of the target expression. This design decision makes ERIK most appropriate for situations in which the characters perform a merely expressive role, where the control of its embodiment is not crucial for safety or successful completion of tasks, such as robotic manipulation of real objects.

From our evaluation, we found that ERIK does in fact solve most cases successfully for both the posture and orientation goals, as long as the embodiment contains enough DoFs to achieve it. ERIK is an iterative algorithm for expressive kinematics that was developed with articulated structures of 1-DoF joints in mind, such as real robots, and in particular, robotic manipulators. It provides a joint model that allows to use techniques initially developed for CGI and not for robotics, such as FABRIK or other IK techniques, which solve for Cartesian (position-based) solutions, instead of angle-based solutions as is commonly used in robotics.

Our algorithm was initially developed towards the problem of expressive gazing, in which a given embodiment, composed of an articulated kinematics chain, is required to orient its end-point towards a target, while also providing expressive control over its posture using expert body knowledge provided by character animators.

Although technically an iterative algorithm, we may also describe ERIK as a multi-phase super-iterative algorithm given that for each set of goals, it solves them iteratively, while using other iterative techniques within each of its iterations. In particular within each iteration it may solve small steps using the popular CCD technique, and will use the custom BWCD technique, which is an adaptation of the CCD algorithm, tailored to simplify some of the steps within ERIK.

6.3.1 From FABRIK to Expressive Robots

The major portion of the algorithm was inspired by the FABRIK technique [89]. While CCD is commonly used in isolation to solve the IK problem required for a given end-point to face a given direction, its solutions suffer from discontinuities and un-natural poses. In this aspect, FABRIK performs significantly better, which makes it more appropriate to be used for expressive motion. However, by operating on the Cartesian level, it cannot ensure reliable orientation constraints. Given a set of parallel, 1-DoF joints as we commonly find in robots, it frequently runs into indeterminations, given that a Cartesian representation of a skeleton can not properly represent induced parallel rotations (i.e., twist). As the authors point out, that results in deadlock situations [89]. They propose that deadlocks can be detected by checking if the distance between the target and the end-point is becoming smaller on each iteration. If not, a deadlock situation is detected. We have imported this concept into ERIK, although we have called these the Nonconvergence cases, for which we provide additional **Nonconvergence Tricks**. Our dealing if the Nonconvergence cases is expressly different, given that under constraints, we must allow the end-point orientation to temporarily move away from the target in some situations, while it is e.g. twisting its root joint to readjust the whole chain to allow reaching the goal, which makes the Nonconvergence detection less trivial. Furthermore, the **Tricks** we apply must consider the fact that we expect to hold the given expressive target posture as best as possible, while in FABRIK, one of the proposed solutions when the target is detected to be out of reach, is solely to place the whole chain in a straight line (which is OK if we do not care for the resulting posture). These limitations have restricted FABRIK's use for robotics, as it was especially formulated for motion-capture of virtual humans, and on

IK problems for position-based targets. Still, FABRIK provides various benefits, such as supporting full-body IK i.e., multiple end-points, non-leaf end-points, closed loops, and prismatic (i.e., sliding) joints. Therefore, we chose FABRIK as the starting point for ERIK so that in the future we may have the chance to replicate and adopt those same features.

6.3.2 BWCD: Backward Coordinate Descent

The BWCD is an IK technique that was specifically created to solve some of the intermediate steps within ERIK. Its execution is similar to CCD's except that execution starts at the root of the chain instead of at the end-point. Therefore the bulk of the warping introduced by BWCD will be concentrated at the bottom of the chain, while CCD tens to introduce it at the top of the chain. The formulation of BWCD was necessary to allow warping postures towards an orientation goal, with preference for having such warping at the root of the chain. That is because by concentrating most of the warping at the root, we expect to maintain more of the shape of the posture through the rest of the chain, up to the tip. Because the warping occurs at the root, which is typically less constrained (such as in a *turret*, or a *pan-tilt* mount), BWCD can return an acceptable solution in a small number of iterations (e.g. <5). Therefore, while being an iterative algorithm, it is fast enough to be used as an internal step within ERIK.

Within ERIK, BWCD is used to operate both on Postures and on Solutions. The **Posture** version solves it in Cartesian space and does not enforce joint rotation limits. The **Solution** version runs in angular space and enforces joint rotation limits.

6.3.3 The ERIK Pipeline

Figure 6.22 shows the main components of ERIK: the inputs **Target Orientation** and **Posture**, the **Joint Model**, the **Warp Posture** phase, the **Solve for Goals** phase and the **Motion Filter**.

ERIK takes in a **Target Orientation**, along with a **Target Posture**, that are to be achieved by the given skeleton, which is the representation of the embodiment's structure, depicted as the **Joint Model**. The **Target Posture** is first naively warped using BWCD, so that its end-point is pointing towards the target orientation. This step, however, breaks the kinematic constraints. Therefore ERIK moves on to the FABRIK-inspired iterative portion that starts by running a **Forward Phase**, and a **Backward Phase** (inspired by FABRIK's own Forward and Backward phases). After the **Backward** phase, the candidate solution exhibits a shape as close as possible to the given **Target Posture**, and respects all kinematic constraints, but its end-point orientation may not match the given **Target Orientation**. Upon testing the candidate solution, if it is within the acceptable parameters, then the solution is returned. Otherwise, the BWCD algorithm is used to orient the solution's end-point towards the given **Target Orientation**. This step will likely cause a slight deformation to the intended posture. If after this step, the new candidate solution is still not acceptable, then ERIK will proceed with a new iteration, starting from the current candidate solution. Before doing so however, it may perform some **Noncovergence Tricks**, in case the algorithm detects that the candidate solution errors are not properly minimizing.

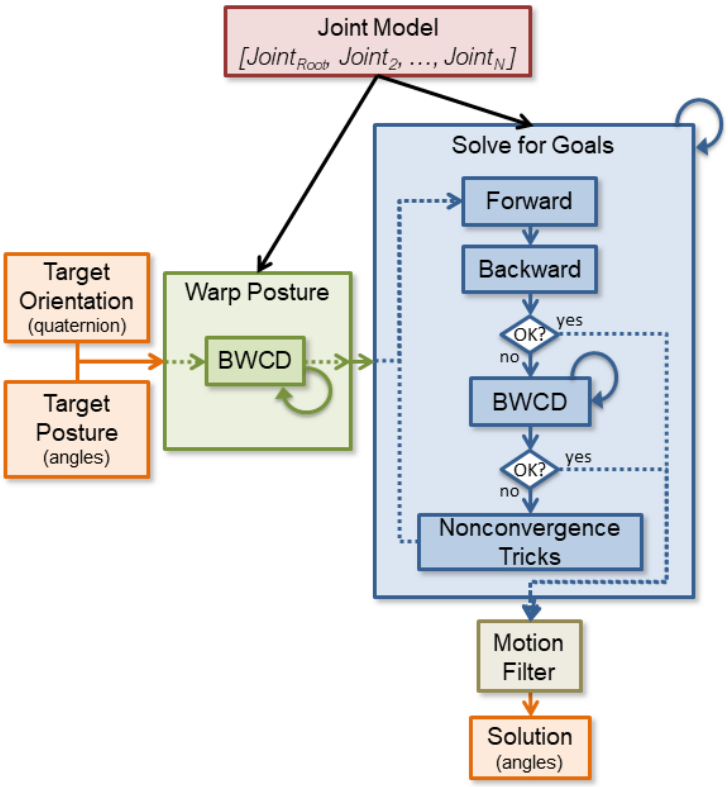


Figure 6.22: ERIK Pipeline. The given Target Posture and Orientation are first warped using BWCD, so that the posture's end-point is aiming towards the Target Orientation, without enforcing joint limits. The result feeds the first iteration of the Iterative portion, which, through various phases on each iteration, returns the final solution. The Joint Model containing the skeletal information and auxiliary operations. The final solution runs through a motion filter to ensure smooth, continuous output.

Nonconvergence Tricks

Upon detection of a non-converging execution, we attempt two approaches, which we call *tricks*, to attempt to get the solution to converge. The first attempt is to add a small offset to the target orientation. It may be the case that the specific target orientation may not be mechanically achievable, and that the algorithm will deadlock trying to achieve it. In that case we attempt to perform a random disturbance of a pre-specified magnitude $\Lambda_{\text{Disturbance}\theta}$ on the Target Orientation, and proceed to the next iteration using the new target.

If the execution comes again to a non-convergence detection, then we attempt to run the CCD technique, using the current intermediate solution as the initial state. This CCD step will likely disturb the expected resulting posture, but will ensure that the end-point is pointing towards the target as best as possible. Given that we take the current solution as the initial state, it is, however expected that the introduced posture disturbance is minimal.

If still this CCD step was unable to provide an acceptable solution, then it is likely that the intermediate solution has become locked due to joint constraints, and that CCD will not be able to solve it. In that case, and only in that final case, will we disregard the target posture, and therefore run the CCD technique again, but starting from the zero-pose.

6.3.4 The ERIK Joint Model and LALUT

In order to allow the use of a FABRIK-based approach with robot-oriented calculus, we started by developing the ERIK Joint Model (EJM) that contains all the required information and operations.

Figure 6.23 shows the unit-sphere EJM space of a joint, where the $\vec{Parent}$ segment is connected to the link's $\vec{Segment}$, which can rotate about a $\vec{RotationAxis}$, within the angular limits of $[Min_\theta, Max_\theta]$.

Vector $\vec{t}$ is a target vector, which specifies the direction where we wish to compute a solution for the joint. Note that the $\vec{Parent}$ was purposely misplaced so that it ends at the origin, to help to visualize this representation as a segment hierarchy, and that all the vectors used are normalized to unit length. Note also that we suggest always considering that the $\vec{Parent}$ is aligned with the $\vec{y}$ of the child's local space, although other conventions can be used. The coordinate axes on the top-right corner of Figure 6.23 should help to clarify the convention in case of any doubt.

The goal of the EJM is to provide answers to the following question: *What angular rotation do I need to apply to the local joint, if I know the joint's rotation limits, and if I know that the Parent joint is a Twister, along with how much it can twist?*

Taking figure 6.23 as example, and note the segment $\vec{S'}$. In order to point the $\vec{Segment}$ to $\vec{t}$, the EJM provides the rotation of α_{swing} on $\vec{Segment}$, resulting in $\vec{S'}$, followed by β_{twist} on $\vec{Parent}$. This would be because the segment wound not achieve $\vec{t}$ through a positive rotation due to its rotational limit Max_θ. Therefore it needs to locally rotate away from the target, and then rely on its parent's Twist capabilities to finally turn to the right direction.

The α_{swing} value is calculated using a pre-computed look-up table which we call the **LALUT**. First, the target direction $\vec{t}$ is turned into a single decimal number we call a latitude λ, which is calculated from Equation 6.9.

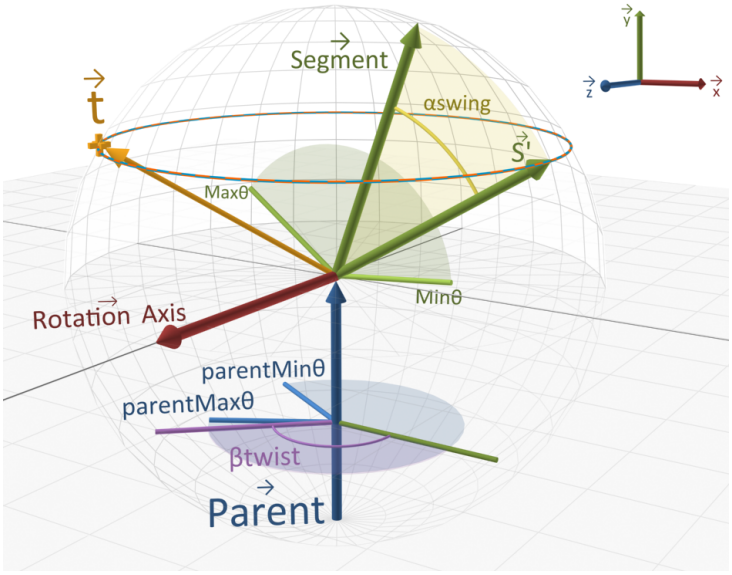


Figure 6.23: The ERIK Joint Model. A joint is defined as having its origin at the tip of its $\vec{Parent}$ segment, and in this coordinate frame, to contain its own $\vec{Segment}$, which can rotate about a given $\vec{RotationAxis}$, within an angle that lies in the range $\{Min_\theta, Max_\theta\}$. In order to achieve a given target t, which is defined in its own local space, it can perform a local rotation of α_{swing}, bringing its segment to S', and then have its parent joint perform a twist of β_{twist} in case the parent is a twister joint.

$$\begin{aligned} \lambda(\vec{t}) &= \sigma(\vec{t}) \cdot \frac{\vec{t} \cdot \vec{P} + 1}{2}. \\ \sigma(\vec{t}) &= sign(\vec{t} \cdot P\vec{O}A) \\ P\vec{O}A &= \vec{R} \times \vec{P} \textit{ (only computed once)} \end{aligned} \tag{6.9}$$

This λ is then used to query the LALUT, which therefore stands for *LAtitude Look-Up Table*. Additionally some auxiliary vectors are computed only once on joint initialization following Equations 6.10

$$\begin{aligned} \vec{OA} &= \begin{cases} \vec{R} \times \vec{S} & \text{, if } \neg(\vec{R} \parallel \vec{S}) \\ \vec{R} \times \hat{\boldsymbol{Y}} & \text{, else if } |\vec{S} \cdot \hat{\boldsymbol{X}}| = 1 \\ \vec{R} \times \hat{\boldsymbol{Z}} & \text{, else} \end{cases} \\ P\vec{O}A &= \begin{cases} \vec{R} \times \vec{P} & \text{, if } \neg(\vec{R} \parallel \vec{P}) \\ \hat{\boldsymbol{X}} & \text{, else if } \vec{R} \cdot \hat{\boldsymbol{X}} = 0 \\ \hat{\boldsymbol{Z}} & \text{, else} \end{cases} \end{aligned} \tag{6.10}$$

Figure 6.24 illustrates the concept of latitude. Given one of the target vectors shown, the latitude will be a number between zero and one, which is inspired on the concept of geographic latitude. The *south pole* corresponds to zero (0.00), while the *north pole* corresponds to one (1.00). The unit sphere is also split in two vertical hemispheres using the plane defined by the vectors $\vec{RotationAxis}$ and $\vec{Parent}$. Given a target vector $\vec{t}$, the hemisphere it lays in is is used to define the sign of the latitude (positive or negative).

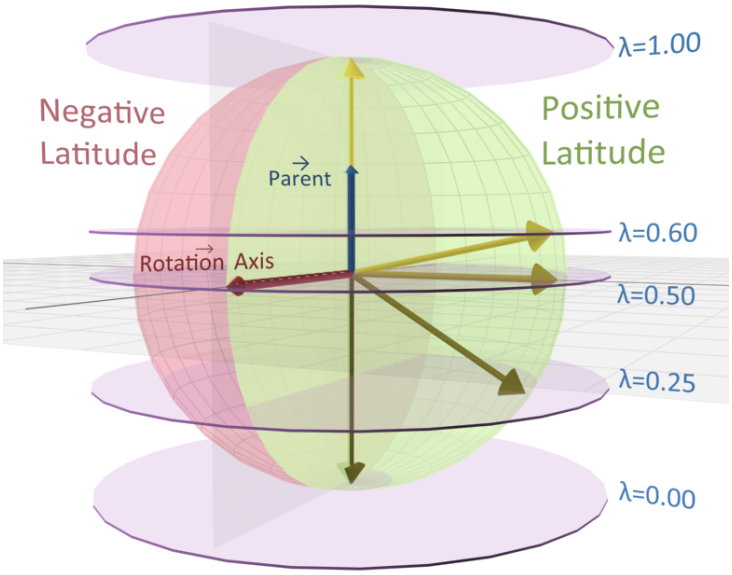


Figure 6.24: The latitude coordinate system. Given a target, represented by the yellow arrows, the corresponding latitude λ is calculated following Equation 6.9. A target pointing at the south pole has a λ of zero (0.00), while one pointing at the north pole has a λ of one (1.00). The λ will be positive or negative depending on if it lies in the right or left hemisphere, which is defined by the plane $\vec{RotationAxis} \times \vec{Parent}$.

The Latitude Look-Up Table

The LALUT serves as a look-up table (LUT), which consists of an indexed array, stored in memory, for which we can associate a value y to an index x. For intermediate values of x that are not present in the table, it should be able to compute the corresponding values of y by interpolation. It is computed only once, on initialization, given that the kinematics of a joint are not expected to change in run-time (i.e., the specific vectors $\vec{Parent}$, $\vec{Segment}$, $\vec{RotationAxis}$ of the system remain the same throughout execution).

This table is computed by iterating a variable a from α_{min} to α_{max}, in small steps (e.g. $\frac{\pi}{180} rad$). The size of the step can be adjusted depending on the needs, with a smaller step requiring linearly more initialization time, but providing higher accuracy. In any case the total execution time should be less than a few seconds on a typical computer.

On each iteration, we rotate $\vec{Segment}$ using a quaternion $Q_{\text{lutStep}} = \text{AxisAngle}(\vec{RotationAxis}, a)$, which produces a new vector $\vec{u}$. We then store the value of a in the LALUT, indexed by its latitude $\lambda(\vec{u})$. Conceptually, this means that the LALUT stores, for a given latitude, the local angle that resulted in it.

Because the LALUT is stored for two hemispheres, it actually contains a positive LUT and a negative LUT. An entry is placed in either the positive one or the negative one depending on the sign of $\sigma(\vec{u})$. Later, for retrieval, the same procedure is followed: Given a target t, a latitude λ and a sign σ are calculated from Equation 6.9. If σ is negative then the negative LUT is queried for λ, otherwise the positive one is queried.

6.3.5 ERIK Parameters and Model Specification

The algorithm relies on a set of Parameters (Π) used to define the execution goals, which are expected to change frequently (even between each solution), along with a set of Hyperparameters (Λ) which should remain unchanged throughout the execution, and are used to configure the algorithm execution. Table 6.2 outlines the main Parameters

and Hyperparameters required for ERIK, along with the symbol by which they shall be represented throughout the document, and especially within Appendix A.1 (Algorithmic Specification).

Table 6.2: Description of Parameters and Hyperparameters of ERIK.

Symbol	Meaning
Π	ERIK Parameters
Λ	ERIK Hyperparameters
$\Pi_{\tau}, \Pi_{\vec{\tau}}$	Target Orientation, Target Direction
Π_{Ψ}	Target Posture
$\Pi_{\Theta_{t-1}}$	Previous Solution
Λ_{Sk}	Skeleton Information (EJM)
Λ_{Sk_i}	i^{th} joint counting from the root[17]
N_{DoFs}	Number of DoFs of the Skeleton
Λ_{ϕ}	Error Function
$\Lambda_{\text{MaxERIKIterations}}$	Maximum iteration count
$\Lambda_{\text{MaxCCDIterations}}$	Maximum iteration count for CCD (and BWCD)
Λ_{foo}	Value of Hyperparameter foo
$\Lambda_{\Xi_{\text{bar}}}$	Extension bar is active
Θ_{ε}	Solution's error value

Table 6.3: List of joint information, given a joint k of a Solution (Θ_k), a Posture (Ψ_k), or a Skeleton (Sk_k). Let Φ represent either a Solution or a Posture.

Symbol	Meaning
$\Psi_{\text{EE}}, \Theta_{\text{EE}}$	A Posture or Solution's End-Effector joint.
$\vec{k_{\sigma}}$	Joint's (child) Segment.
$\vec{k_{RA}}$	Joint's Rotation Axis.
$\vec{k_{OA}}$	Joint's Orthogonal Rotation Axis.
$\vec{k_{POA}}$	Joint's Parent-Orthogonal Rotation Axis.
$\Phi_{k_{\rho}}$	World-Position of joint.
$\Phi_{k_{\theta}}$	Local angle of joint.
$\Phi_{k_{Q}}$	World-Frame (basis) of joint (Quaternion).
$\Phi_{k_{L}}$	Local-Frame orientation transform (Quaternion).
$\Phi_{k_{\Omega}}$	World-frame orientation transform (Quaternion).
$\vec{\Phi_{k_{d}}}$	Direction where the joint segment is pointing at (unit vector).

Besides the Parameters and Hyperparameters, ERIK requires the concept and model of Solutions (Θ), Postures (Ψ) and Links (K). The Solution object is used both for intermediate and candidate solutions, used internally during the execution of ERIK, and also to represent initial and final solutions provided to and by the algorithm. The Posture object is similar to the Solution one, except that it is used to represent a target pose, which may be represented either based on a set of angles, or a set of positions for each joint, and which may or may not comply with the mechanical limits of the Skeleton. In case of Solutions, they contain kinematic information that adheres to the joints' kinematic limits.

Additionally, candidate and final solutions contain an error value Θ_{ε} which represents the result of the error function $\Lambda_{\phi}(\Theta)$. The Skeleton information object contains the set of Links, along with information such as which is the root or end-point joints of the chain.

Both Solutions and Postures contain joint information represented in a similar way. The joint fields used by both are listed in Table 6.3, while some additional algebraic definitions are listed in Table 6.4. The computation of some

of the fields from Table 6.3 is explained in Equations 6.11a–6.11c. Let us clarify that the *orientation transforms* Φ_{k_L} and Φ_{k_Ω} represent the orientation to which the joint's segment is facing, after applying its own local rotation. Also note that an alternative to Equation 6.11c would be to take the $\vec{y}$ axis of Φ_{k_Ω}'s corresponding matrix.

$$\Phi_{k_L} = Q_{\text{AxisAngle}}(\vec{k_{RA}}, \Phi_{k_\theta}) \tag{6.11a}$$

$$\Phi_{k_\Omega} = \Phi_{k_Q} \cdot \Phi_{k_{Q_L}} \tag{6.11b}$$

$$\vec{\Phi_{k_d}} = Q_{\text{AxisAngle}}(\hat{Y}, \Phi_{k_\theta}) \tag{6.11c}$$

Table 6.4: Definition of mathematical symbols used in the algorithms.

Symbol	Meaning
Q_θ, Q_v	Scalar and Vector parts of quaternion Q
Q_M	Rotation Matrix that corresponds to quaternion Q.
Q_k	Axis k of Q's rotation matrix Q_M, $k \in \{x, y, z\}$ (simplification of Q_{M_k}).
$\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}, \hat{\boldsymbol{Z}}$	Unit-vectors in the X, Y or Z directions.

6.3.6 The Error Function

To measure the quality of the solutions produced by ERIK, we established two concurrent error measures, $\epsilon_{\text{Orientation}}$ and $\epsilon_{\text{Posture}}$. These are concurrent measures because in most cases, minimizing one results in not minimizing the other. Through successive iterations, the algorithm attempts to minimize the error function Λ_ϕ (Equation 6.12), which calculates a weighted sum of the two measures. The error threshold $\Lambda_{\text{Threshold}_\varepsilon}$ specifies when the result of the error function is small enough to be acceptable (for which it can successfully terminate and return the computed solution). In all cases, any value that measures error lies within the interval $[0.0, 1.0]$. The orientation error function $\phi_{\text{Orientation}}$ calculates the $\epsilon_{\text{Orientation}}$ for a given solution, while similarly, ϕ_{Posture} calculates its $\epsilon_{\text{Posture}}$.

$$\Lambda_\phi(\Theta, \tau, \Psi, \Lambda) = \Lambda_{\text{OrientationErrorWeight}} \cdot \phi_{\text{Orientation}}(\Theta_{\text{EE}_\Omega}, \tau, \Lambda) + \Lambda_{\text{PostureErrorWeight}} \cdot \phi_{\text{Posture}}(\Theta, \Psi, \Lambda) \tag{6.12}$$

These two error functions are defined in equations 6.13 and 6.14, and further specified in the appendix in Algorithms 5 and 6. The posture error function ϕ_{Posture} measures how different the posture of a solution is, in shape, from the target one. It does so by measuring the local angular deviation between each non-twister solved joint, and target joint, and is designed to punish more for deviations closer to the end-point than closer to the root, which supports our preference.

In equations 6.14 and 6.15, α is a shortcut for the aggravation factor $\Lambda_{\text{ErrorAggravation}}$.

[17] Thus the root joint is Λ_{Sk_1}, the end-effector is Λ_{Sk_N}, and the Superpoint (Section 6.3.8) will be $\Lambda_{\text{Sk}_{N+1}}$.

$$\phi_{\text{Orientation}}(\omega,\tau,\Lambda) = \begin{cases} \min(Z(\tau,\omega), Z(\tau, \text{QAA}(\text{RotVQ}(\hat{\boldsymbol{Y}},\omega),\pi)\cdot\omega)) & , \textit{if } \Lambda_{\Xi_{\text{SymmetricEndpoint}}} \\ Z(\tau,\omega) & , \textit{otherwise} \end{cases} \\ Z(\tau,\omega) = \frac{\min(|\tau-\omega|, |\tau+\omega|)}{\sqrt{2}}. \tag{6.13}$$

$$\phi_{\text{Posture}}(\Theta,\Psi,\Lambda) = \frac{1}{\Lambda_{\text{PostureNorm}}} \sum_{i=1}^{N_{\text{DoFs}}} \begin{cases} 0 & , \textit{if } \text{IsTwister}(\Lambda_{\text{Sk}_i}) \\ \alpha^i \cdot |(1-\frac{1+\Upsilon(\Psi,i)}{2}) - (1-\frac{1+\Upsilon(\Theta,i)}{2})| & , \textit{otherwise} \end{cases} \\ \Upsilon(P,i) = \begin{cases} \|\Lambda_{\text{Sk}_{\text{Root}_\sigma}}\| \cdot \|P_{(i+1)_\rho} - P_{i_\rho}\| & , i = 1 \\ \\ \|P_{i_\rho} - P_{(i-1)_\rho}\| \cdot \|P_{(i+1)_\rho} - P_{i_\rho}\| & otherwise \end{cases} \tag{6.14}$$

The Hyperparameter $\Lambda_{\text{ErrorAggravation}}$ (used in the Equations 6.14-6.15 as α) defines how worse the punishment becomes, as the function calculates deviations closer to the end-point. A value of 1.0 would mean that the punishment is the same across the links. A value of 2.0 means that a given deviation amount at one link would result in twice the error value, one level up the kinematic chain. We can see that the resulting value of ϕ_{Posture} is divided by the $\Lambda_{\text{PostureNorm}}$, which reduces the final sum to a value in the interval $[0.0, 1.0]$. This hyperparameter is calculated once on the skeleton's initialization and given by Equation 6.15.

$$\Lambda_{\text{PostureNorm}} = \sum_{i=1}^{N_{\text{DoFs}}} \begin{cases} 0 & , \textit{if } \text{IsTwister}(\Lambda_{\text{Sk}_i}) \\ \alpha^i & \text{otherwise} \end{cases} \tag{6.15}$$

Depending on the target application, and the embodiment used, one can use different values for the error measure weights, and for the error threshold. We share, as an example, that for a 5-link robotic manipulator aimed at entertainment applications, where expressivity and responsiveness is more important than precision, we achieved good results using an error threshold of 0.04, with a weight of 1.0 for $\Lambda_{\text{OrientationErrWeight}}$ and 0.2 for $\Lambda_{\text{PostureErrWeight}}$. As such, we took these values as a reference when evaluating the algorithm as we will report further in the appropriate section of the document (Section 6.3.10).

6.3.7 The Nutty Motion Filter

The final component of the pipeline is the Nutty Motion Filter, which we refer to as the NMF and has been extensively described in Section 6.2. This piece's function is to interpolate successive ERIK solutions, to ensure that the final produced movement is smooth and continuous. Furthermore, it can shape the motion to make it appropriate for use with robots.

The NMF allows to define limits for the velocity, acceleration and jerk[18] of the signal. Additionally it includes a set of tweaking parameters that can be creatively explored to provide different characteristics to the motion, such as allowing it to respond fast, as in a light character, or respond very slowly and with a lot of inertia, as in a heavy character.

[18] Jerk is commonly used in robotics. It is the derivative of the acceleration. Think of it as the speed at which the acceleration changes.

The motion filter is calculated individually for each joint, at the end of each frame in the animation engine's animation cycle, which is not necessarily synchronized (and should not be) with the ERIK solver engine. We recommend not attaching these given that the ERIK cycle may have inconsistent frame times and drop to a lower rate than is expected in the animation cycle.

The output of the NMF on each frame is given by the function $X(x(t), t(i), s)$, where $x(t) : \mathbb{R}_0^+ \rightarrow [P_{\text{min}}, P_{\text{max}}]$ is the motion signal history, i.e., the previous positions that were output from the filter. The parameters P_{min} and P_{max} represent the minimum and maximum values respectively (e.g. angular limits). Note that each joint may define its own limits and motion parameters for the NMF. $x(0)$ corresponds to the initial position of the joint and must be initially specified. The function $t(i) : \mathbb{N}_0 \rightarrow \mathbb{R}_0^+$ (shortened to t_i) represents the time at each sample i, such that $0 \leq t_{i-1} < t_i$, and $t_i - t_{i-1} = \Delta t$, where Δt is a fixed time-step, calculated from the animation output rate R, such that $\Delta t = \frac{1}{R}$. Note that from this definition, i refers to the current sample, and therefore the current time is always represented by t_i, while the time of the last sample is t_{i-1} and so on.

Finally, the set-point s is the new target position, and is used to calculate the *induced velocity* $\dot{x}(t_i)$. With this consideration, $x(t_i)$ is used to represent the output that will be computed of the filter at the current time (not in the history yet), while s therefore represents the input. As such, $\dot{x}(t_i)$ must be calculated from s instead of $x(t_i)$.

Equation 6.16 contains the explicit definition of the NMF equations. Within them we can find the various motion parameters, which we follow to explain.

The β parameter controls the exponent of the position-limiter de-acceleration, allowing to control how close to the angular limit of the joint the output is allowed to get before being saturated. As β increases, the saturation becomes more similar to a hard clamping function. The use of a soft limiter allows the output filter to avoid overshooting any joint beyond its physical limits, given that in most cases, overshooting at the software's output level would result in a hard break at the *hardware* level. The default value for β is 1.

The $\{\sigma, \rho\}$ parameters both represent *smoothness* and *responsiveness* respectively, and allow to tweak the filter, changing how quickly it responds and how much it is allowed to oscillate. We call these the *character parameters*, as different configurations for them will shape the motion differently. As such we argue that they can be used to model different character traits, even when the same physical limits are enforced. The *smoothness* parameter σ will ease out the oscillations. However, depending on other filter parameters such as the physical limits, fully easing out might become too slow and make the motion seem too muddy and flat. That is where the *responsiveness* parameter ρ comes in, which allows to precipitate the easing out, so that it may still be smooth, but faster, and thus, more responsive.

Please refer to Section 6.2 for more details and examples on the Nutty Motion Filter and the use of its parameters.

$$
\begin{aligned}
\chi(x,t_i) &= x(t_{i-1}) + \lambda(\psi(x,t_i), velocity\_limit) \\
\psi(x,t_i) &= \dot{x}(t_{i-1}) + \lambda(\frac{\xi(x,t_i) - \dot{x}(t_{i-1})}{\Delta t}, acceleration\_limit) \\
\xi(x,t_i) &= \ddot{x}(t_{i-1}) + \lambda(\frac{\frac{v \cdot \mathrm{H}(v) - \dot{x}(t_{i-1})}{\Delta t} - \ddot{x}(t_{i-1})}{\Delta t}, jerk\_limit) \\
v &= \Omega(\dot{x}(t_i), x(t_{i-1}), P_{\max}, P_{\min}, \beta) \\
\dot{x}(t_k) &= \begin{cases} \frac{s - x(t_{k-1})}{\Delta t} & , if\ k = i \\ \\ \frac{x(t_k) - x(t_{k-1})}{\Delta t} & otherwise \end{cases} \\
\mathrm{H}(v) &= \frac{v}{2} \cdot \left(\tanh\left(\left(\frac{|v|}{1-\rho} \right)^{1-\sigma} - \pi \right) + 1 \right), 0 \leq \sigma \leq 1, 0 \leq \rho < 1 \\
\lambda(x,k) &= \frac{k}{2} \cdot \tanh(x/\frac{k}{2}) \\
\Omega(\dot{x}, x, P_{\max}, P_{\min}, \beta) &= \begin{cases} \dot{x} \cdot \left(1 - \left(\frac{x - P_{\min} - \alpha}{\alpha} \right)^{2\beta} \right), & if (x > \alpha\ \&\ \dot{x} > 0) \mid (x < \alpha\ \&\ \dot{x} < 0) \\ \\ \dot{x}, & otherwise \end{cases} \\
\alpha &= \frac{P_{\max} - P_{\min}}{2}
\end{aligned}
\tag{6.16}
$$

6.3.8 The Superpoint

In order for some of the calculations to work on the end-point link, we created the concept of the *Superpoint*. This is a *fake*, 0-DoF joint, used within Postures, that extends the end-point's segment. It allows the End-point to be treated as if it had a child link with 0-DoF. Whenever the Posture's data for the End-point is changed, the data for the Superpoint is also updated, using the rules in Equation 6.17. Also note, by the definitions in Table 6.2 that the Superpoint may be referred to either as $\Psi_{EE_{\text{Child}}}$ or $\Lambda_{\text{Sk}_{N+1}}$.

$$
\begin{aligned}
\Psi_{EE_{\text{Child}_\theta}} &= 0 && \textit{Let } \Psi \textit{ be the posture and EE the Endpoint} \\
\Psi_{EE_{\text{Child}_Q}} &= \Psi_{EE_\Omega} \\
\Psi_{EE_{\text{Child}_\rho}} &= \Psi_{EE_\rho} + Rotate(EE_\sigma, \Psi_{EE_\Omega})
\end{aligned}
\tag{6.17}
$$

6.3.9 ERIK Extensions

Not all embodiments and application pose the same requirements. As such, ERIK was designed with the idea of extensions (Ξ) in mind. Think of extensions as options that you may want to have activated or not, which may change the way the algorithm runs, and thus can result on better outcomes for a given situation (while possibly providing worse outcomes, for a different situation, with different criteria). In that sense, Extensions fall in the category of Hyperparameters, and are therefore contained within those. The extensions we have designed and included in the algorithm on this paper were all found to yield better results given the purpose we define (i.e., entertainment). If your purpose or criteria is different, there is an option to disable such extensions, to modify them,

or even to create new ones. The currently included extensions are:

$\Xi_{\textbf{SymmetricEndpoint}}$ Allows the algorithm to flip the end-point upside down. This is useful if the end-point is symmetric, and can be used both ways. By using such a design, and activating this extensions, the possible solution space doubles, and therefore allows the algorithm to properly solve in many more cases.

$\Xi_{\textbf{AvoidEdges}}$ Instructs the algorithm to avoid positioning joints exactly on its angular limits. In cases where some minor deviation from the goals is accepted, this extensions helps to avoid dead-lock situation when the joint limits are equivalent to singularity-prone angles (such as $\pm\frac{\pi}{2}$).

$\Xi_{\textbf{NonConvOffsetTrick}}$ Allows ERIK to attempt the *Non-converging Offset Trick* when a non-converging execution is detected. This trick applies a small, random orientational offset to the target orientation in cases where the execution has become non-converging. It results in an increase on the amount of cases where the algorithm is able to converge, as long as a minor deviation from the goals is accepted. The deviation applied is defined by hyperparameter $\Lambda_{\text{Disturbance}\theta}$.

$\Xi_{\textbf{NonConvCCDTrick}}$ Allows ERIK to run the CCD algorithm on a non-converging solution, after the *Non-converging Offset Trick* failed to bring the execution into a converging state. It typically results in achieving the orientation goal esier, while allowing the posture goal to become more disrupted (as expected through the direct use of CCD).

6.3.10 Evaluation

Before claiming on the quality and success of ERIK, we are required to run extensive evaluation procedures. Given that the algorithm aims at being used with any embodiment and expressive pose created by animators, we did not want to access if the resulting solutions were able to solve particular use cases, as those should be tailored creatively by such animators in the future. Instead, we realized that we wanted to assure that the algorithm would be able to fulfill an animator's intentions while authoring expressive postures for use with ERIK. Therefore, given an expressive posture, we wanted to test how well the algorithm was able to hold its shape, while orienting its endpoint towards various different target orientations. At the same time, we were concerned with how well the resulting solution effectively aimed at the given target orientation, regardless of the resulting expression. This is because, for interactive, real-work situations, we consider it particularly important to get the aiming right, so that the character is believable, and is able to provide an immersive experience for the user. The expressivity of any particular posture is not, in fact, evaluated. Instead, the evaluation focused on what can be regarded as a meta-expressivity, i.e., given any posture, which an animator would have thought to be appropriately expressive for some purpose, we measure how well the algorithm is able to reach a shape that is similar to the one given by that posture, and that capability is what is evaluated as the expressive goal.

With the purpose of evaluating how well ERIK solves both the orientational goals and the expressive goals, we performed what can be dubbed as a *brute-force* evaluation procedure. This procedure consisted of generating many different expressive postures, and testing how well ERIK is able to solve them for a large set of different orientation targets. All this was done for several different embodiments. It is impossible to cover every possible case through such approach. However we consider that the tested cases are a sufficient reflection of how the algorithm performs

in general, and are representative of both 1) the space of different expressive postures that any animator would possibly produce; and 2) the space of different orientations to which the character might possibly have to face.

Additionally we compared ERIK against an existing technique. In this case we followed the description by Baerlocher on how to solved an IK problem for multiple tasks [73] based on the DLS method. Taking the example of a two-priority problem, the first task, with higher priority, would be the orientation constraint, while the secondary task, of lower priority, would be the postural constraint. The technique was evaluated in the same way we tested ERIK with multiple embodiments, and the results were further included in the same analysis.

Error Measures

Through preliminary experimentations, we decided to established a weight of 1.0 for $\epsilon_{\text{Orientation}}$ and 0.2 for $\epsilon_{\text{Posture}}$, along with an error threshold $\Lambda_{\text{Threshold}\varepsilon}$ of 0.04. The use of these weights states that it is more important to get the orientation goal solved than the expressive posture one. This is because we prefer that the resulting solution is properly aiming at the target orientation, and, because we are aiming at expressive applications, we tolerate that the posture may fall slightly out of shape, as long as it is still within an acceptable amount of disfigurement.

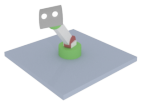
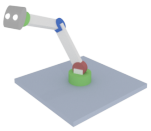
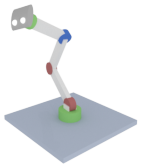
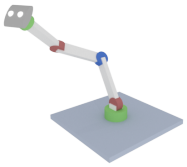
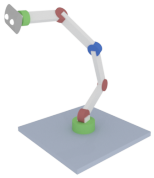
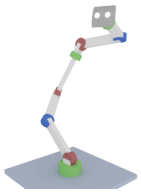
As to the error threshold, while it should be adapted to each embodiment, we found 0.04 to be a decent tolerance to demonstrate and compare the results among different embodiments. In a real-world application, we would have tweaked a different error tolerance for each of the different skeletons.

Evaluation Embodiments

In order to see how the algorithm performed for embodiments with various amounts of DoFs, we established 7 different test skeletons, which are presented in Table 6.5.

It is important to note that we have included skeletons with a low number of DoFs in order to validate that the algorithm behaves as expected, even in such highly constrained situations. Our hypothesis here is that these low-DoF skeletons will yield very poor results, and that by adding more DoFs, or configuring them in different ways and with different angular limits, we can augment the expressive capabilities of the expressive character, which should be proven by yielding better results in the same type of evaluation.

Table 6.5: Definition of test-skeletons used in the evaluation procedure. In the figures, green nodes represent a $\vec{Y}$-oriented rotation axis, while a red one is oriented with $\vec{X}$, and blue with $\vec{Z}$.

Skeleton	# DoFs and rotation axis sequence (root to endpoint)	Angular Range	Illustration
A	3 links Y-X-Y	$[-\frac{\pi}{2}, \frac{\pi}{2}]$ (all links)	
B	4 links Y-X-Z-Y	$[-\pi, \pi]$ (all links)	
C	5 links Y-X-X-Z-Y	$[-\frac{\pi}{2}, \frac{\pi}{2}]$ (all links)	
D	5 links Y-X-Z-X-Y	$[-\pi, \pi]$ (all links)	
E	5 links Y-X-Z-X-Y	$[-\frac{\pi}{2}, \frac{\pi}{2}]$ (all links)	
F	6 links Y-X-X-Z-X-Y	$[-\frac{\pi}{2}, \frac{\pi}{2}]$ (all links)	
G	8 links Y-X-Z-X-Y-X-Z-Y	$[-\frac{\pi}{2}, \frac{\pi}{2}]$ (all links)	

Procedure

Each skeleton was used to test ERIK in various different target postures and target orientations. The target postures were generated by sweeping the angular range of each joint as long as it is not a root twist-joint, or an endpoint twist-joint, with a given resolution, from its min_θ to its max_θ, and combining them to create a large set of postures. Based on our convention, the twist-joints are the ones whose rotation axis is aligned with $\vec{Y}$. In fact, the full set of skeletons has twist-joints both as root and as endpoints, which means that for each skeleton, all joints except these two were swept to generate the target postures. The reason why we exclude these two are that they do not change the actual shape of the posture, and including them would dramatically increase the simulation space.

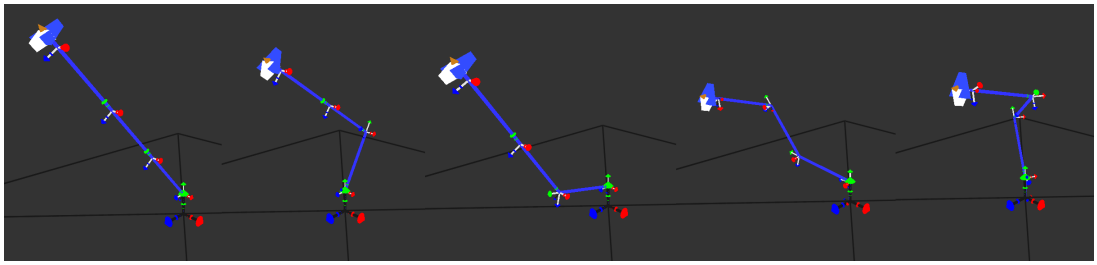


Figure 6.25: Five example test postures for skeleton C, representing the type of generated postures tested.

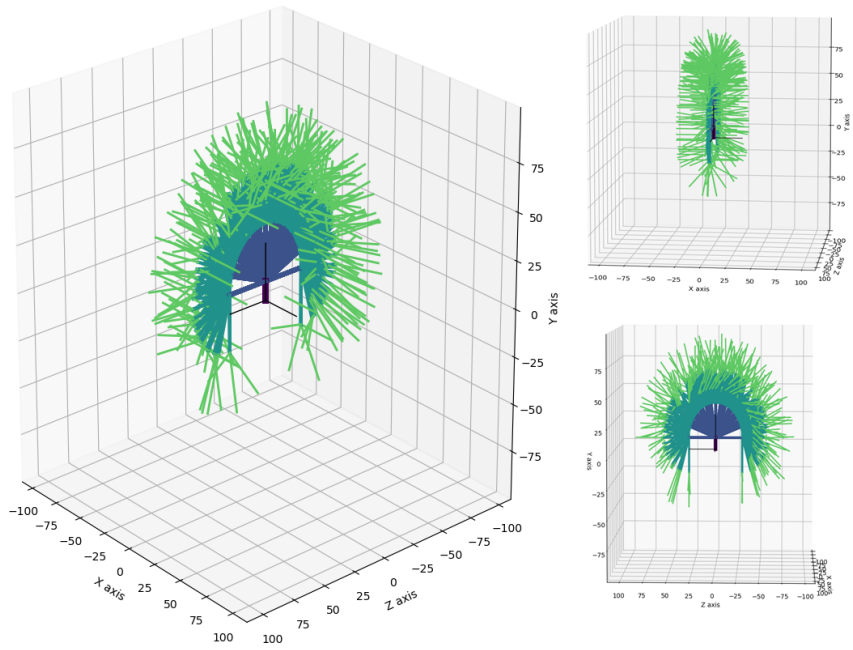


Figure 6.26: The postural simulation space for skeleton C, illustrating 3789 target postures in three different views.

In Figure 6.25 we can see examples of different target postures generated for Skeleton C, with 5 links. The whole postural simulation space for the same skeleton is illustrated in Figure 6.26, where we see each of the 3789 generated postures overlapped. Note that the simulation space contains no rotation on the root joint, as it would merely revolve the posture around the vertical $\vec{Y}$ axis, and thus would not change the posture's actual shape.

Similarly, for the target orientations, we wanted to test the most various orientations in all different directions around the character. For that we swept a horizontal angle α_h, a vertical angle α_v, and a twist angle α_t, all in the range $\{-\pi, \pi\}$. The sets of three angles were then used to generate a large number of target orientations (as quaternions) through the Yaw-Pitch-Roll composition method. It may seem that for α_v, sweeping in the range $\{-\frac{\pi}{2}, \frac{\pi}{2}\}$ would have been enough; however extending the range to $\{-\pi, \pi\}$ introduces additional target orientations in which the target orientation is defined *upside-down*. We wanted to include such cases in the evaluation, to ensure that the algorithm was also numerically capable of dealing with them. As a result, for each of the generated postures of each skeleton, we took a *point-cloud* centered on the robot, each point representing a target orientation (including the roll component). This method allowed us to run the algorithm on a large amount of different parameters, while also taking extra care to ensure that potential failure points, such as angles set to $\pm\pi$, and orientations aligned with any of the coordinate axes, were guaranteed to be included.

Figure 6.27 shows the orientational simulation space as a point-cloud with 7609 points, which was used to run simulations for each posture or each skeleton. Each point illustrates a polar and azimuthal orientation angle, along with a twist that is given by the radial distance from the center. Therefore variously twisted orientation quaternions are tested in the same direction. Both positive and negative twist angles are tested - in this representation, the zero-twist orientations are represented by the points that lie at the center of the point-cloud radius, while positive ones increase towards the exterior, and negative ones towards the interior.

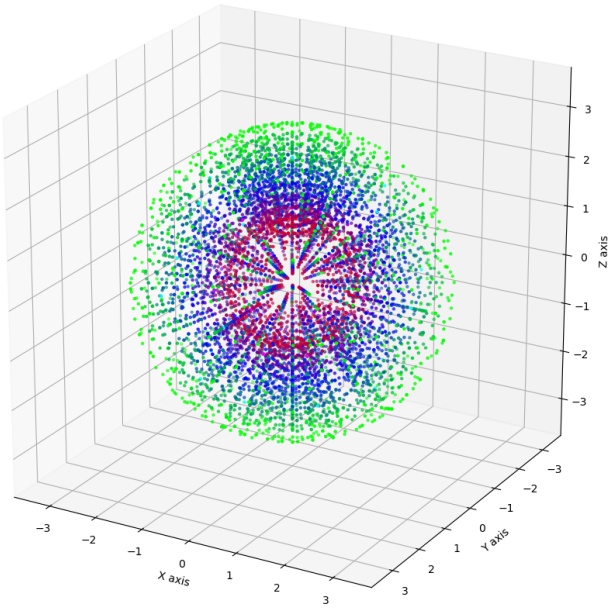


Figure 6.27: Illustration of a point cloud corresponding to 7609 test samples, each representing a different quaternion to be used as the target orientation. Each point represents a polar and azimuthal orientation angle, along with a twist that is given by its radial distance. The colors of the points are modulated from the twist angle (red are negative, blue is zero, green are positive). Please note that the apparent existence of blue or even green dots at the center is an illusion - they are in fact part of target directions that are roughly aligned with the viewing direction.

Comparison with the two-priority DLS

In order to compare ERIK against another existing technique, we chose to use the two-task-priority DLS as described by Baerlocher [73], using Maciejewski's damping factor [77] and the SVD method. These techniques have already been reviewed in Section 3.2.1, and are reiterated in Equation 6.18.

This technique posed as the most appropriate to provide a comparison to ERIK, as it allows us to define two tasks: the orientation task characterized by $J_1 \Delta\theta = \vec{e_1}$, with a high priority, and the postural task $J_2 \Delta\theta = \vec{e_2}$ with a lower priority. Both J_1 and $\vec{e_1}$ are calculated as they would usually be for an orientation-constraint task. The secondary task is meant to keep the joint angles as close as possible to a given target posture Ψ. Therefore we calculate $\vec{e_{2,}} = \Psi - (\vec{s} + \Delta x)$, having $\Delta x = J_1^{\dagger^{\lambda_1}} \vec{e_1}$, i.e., the current solution to the primary task. We recall also that $\vec{s}$ is the initial joint configuration. Therefore the error vector for the secondary task represents the error between the target posture and the posture that results from solving the primary task. As the secondary task aims at solving towards a given posture, its Jacobian matrix J_2 should correspond to the Identity matrix I^n, where n is the number of joints. We add just one correction to it, by setting the value for the 1^{st} and n^{th} joint to zero in case that joint is a twist joint, given that as in ERIK, those do not change the resulting posture's overall shape.

Link	θ (Joint Angle)	α (Twist Angle)	a (Link Length)	d (Joint Offset)
1	0	$\frac{\pi}{2}$	0	10
2	$\frac{\pi}{2}$	0	30	0
3	0	$\frac{\pi}{2}$	30	0
4	$\frac{\pi}{2}$	$\frac{\pi}{2}$	0	0
5	$\frac{\pi}{2}$	0	0	40

Table 6.6: Denavit-Hartenberg parameters (classic) used to run the simulations of DLS on Skeleton C.

$$
\begin{aligned}
\Delta\theta &= J_1^{\dagger^{\lambda_1}}\vec{e_1} + (J_2 P_{N(J_1)})^{\dagger^{\lambda_2}}(\vec{e_2} - J_2 J_1^{\dagger^{\lambda_1}}\vec{e_1}) \\
J^{\dagger^{\lambda}} &= (\sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda_\sigma^2})\mathbf{v}_i\mathbf{u}_i^T \quad //SVD \\
P_{N(J)} &= I - J^{\dagger}J \\
\lambda_\sigma &= \begin{cases} \frac{d}{2} & \text{if } \sigma_{\min} \leq \frac{d}{2} \\ \sqrt{\sigma_{\min}(d - \sigma_{\min})} & \text{if } \frac{d}{2} \leq \sigma_{\min} \leq d \\ 0 & \text{if } \sigma_{\min} \geq d \end{cases} \\
d &= \frac{\|\vec{e}\|}{b_{\max}}
\end{aligned}
\tag{6.18}
$$

The simulations using DLS were ran using Skeleton C, for which each joint can rotate only 90° to each side, therefore making it much more difficult to face orientations that are *behind* the robot. Using ERIK, that was not a problem given that we have the extension $\Xi_{\text{SymmetricEndpoint}}$, which allows the end-effector to be used upside-down. While this feature is still used within DLS at the error function level, it is not properly considered by the actual algorithm. We therefore also apply a correction to the orientation target in order to keep its up-side oriented in a way that it is reachable by the test skeleton given its joint limits. This correction was the only one that we ever added to enhance the results for a particular skeleton or technique, and in fact, is used only to enhance the results of the technique to which we are comparing ERIK, for more realistic results. Initially we considered the results of DLS too bad, and therefore the comparison (while optimistic for ERIK) was considered inappropriate.

The correction is made by flipping the target orientation's quaternion upside-down (i.e., performing a rotation of π about the unit $\vec{Z}$ vector) in specific regions of the target space, so that we guarantee that the target's up-side is always directed to facilitate the result of DLS using Skeleton C, i.e., when the target is facing forward then its Y-axis will always be facing down; when the target is facing backward, then its Y-axis will always be facing up. Therefore the target is never an orientation that is mechanically unachievable a priori.

The classic Denavit-Hartenberg parameters used to model Skeleton C are presented in Table 6.6.

Finally, because the DLS technique outcome is very dependent on the maximum number of iterations execution, we also ran several trials with the technique using 100, 200, 400 maximum iterations. Each will be referred to as e.g. DLS100 or DLS400. Whenever we refer solely to DLS, we will be referring to the best version of it (DLS400).

We started by running a set of simulations using DLS100_nopost, which is DLS100 without the homogeneous solution, i.e., without the second part of the equation which attempts to solve for the target posture using the primary

Jacobian's null space. The goal with these simulations was to assess how well the DLS implementation was able to solve solely for orientation targets using Skeleton C, which is highly prone to singularities. Although the technique we follow is stated to be free of *algorithmic* singularities, the parametrization of the skeleton may also introduce *kinematic* singularities (e.g. gimbal lock). In fact, upon running the simulation using DLS100_nopost, we found that there were many target orientations for which the algorithm became stuck yielding a very high orientation error, which we attribute to such type of singularities. In order not to impair the results of the DLS simulations when compared to ERIK (which does not suffer from such singularities), we further used this simulation to filter the DLS results in order to remove all the samples for which the DLS100_nopost version yielded an **orientation error** above 3x the specified threshold, thus excluding from the comparison exceptionally bad results that were not due to the posture constraint, but to inappropriate handling of kinematic singularities. As such, in the comparison of ERIK and the DLS variants (Section 6.3.10) the results from the DLS simulations presented are the results of applying such filter.

Figure 6.28 compares the resulting orientation error histograms and normal distribution plots for ERIK and the filtered DLS100_nopost on Skeleton C. Note that the data from ERIK actually resulted of the full simulation of ERIK with both orientation and posture targets. Therefore we see that ERIK has performed better in solving the orientation constraint even through it also had the posture goal. The DLS performed slightly below expectations, but we must consider that the simulation attempted to orient the end-effector to a very large big range of orientations in full 3D, i.e., pan, pitch and roll of the end-effector, towards the full vertical and horizontal 360°range. The DLS results here have been corrected to match the number of samples of the ERIK one (which was also solved for each posture), therefore presenting a comparable scale; otherwise considering only the range of orientation targets, the DLS result would present far less samples and thus make these difficult to compare.

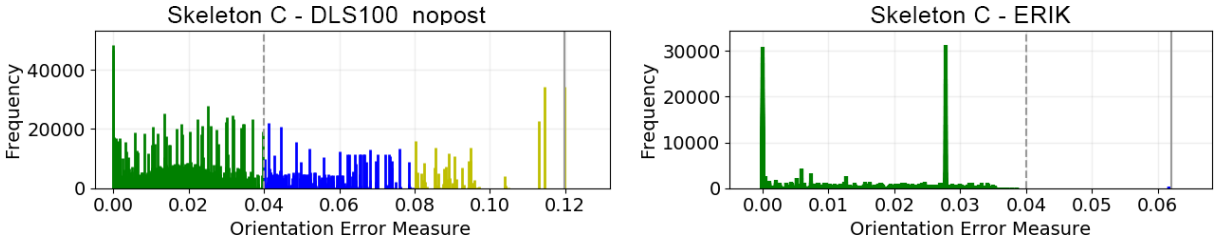
Results

Using ERIK, a total of 239 245 243 samples were simulated from 39 739 postures across all 7 skeletons. The DLS was simulated for a total of 86 491 503 samples from 3789 postures using Skeleton C, in this case using three different maximum iteration counts. However as explained in the previous section, the DLS results were further filtered resulting in a total of 61 684 497 selected samples.

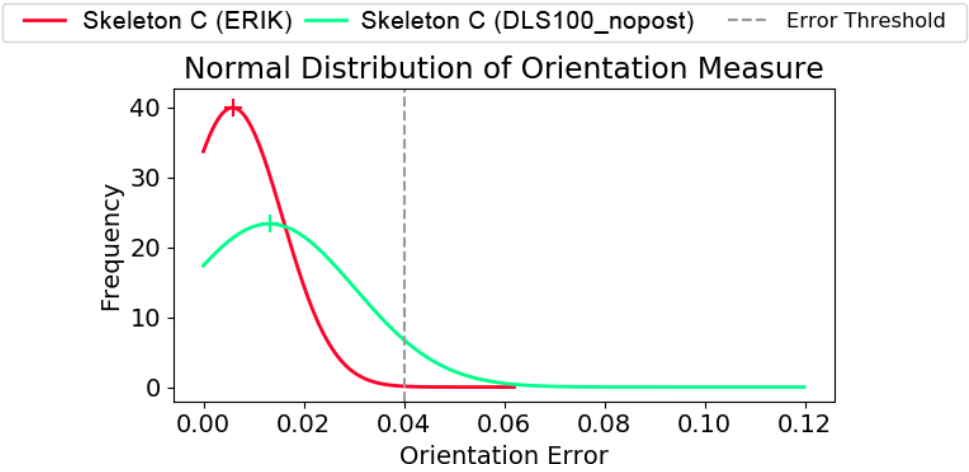
The simulations were ran on a high-performance computer cluster (HPCC) containing a mix of nodes with AMD Opteron 6180 SE and 6344, and AMD EPYC 7401 CPUs, organized into nodes of either 48 or 96 CPUs. In order to normalize and interpret the performance results from these simulations in comparison with a typical laptop CPU, we have searched for single-core benchmarks of these CPUs on community-sourced benchmark websites. We took as an example the Intel i7-7700HQ, which is a popular CPU, featured in many mid and high-end personal laptops, and that is also at least 2 years old (launched Q1'2017) to represent an average laptop CPU. Despite the multiprocessing capabilities of any of them, we were interested in the single-core performance, as each simulated sample ran as a single-core process, and are also expected to run as such in a real-world application (even if it is used within a multi-threaded/multi-core application, the IK engine per se should run sequentially in a single thread).

Table 6.7 shows the highest benchmark of each of these CPUs, using a Linux 64-bit system, as found on the community-sourced Geekbench website[19]. We consider these values to stand as an acceptable comparison of how

[19] `https://browser.geekbench.com/` (accessed January 12, 2019)



(a) Orientation error histogram for skeleton C running ERIK compared to DLS100_nopost.



(b) Normal distribution plots of the orientation error of ERIK compared to DLS100_nopost.

Figure 6.28: Comparison of orientation errors of ERIK on Skeleton C compared to DLS100_nopost after filtering out samples with excessive orientation error in the latter.

the performance statistics collected through the HPCC compare to those of an average computer. By considering this score instead of theoretical values such as MIPS or GFLOPS, we are also considering more of a general performance capability without considering particular architecture-wise optimizations. For ERIK, all the simulations except the largest one were arbitrarily assigned to an Opteron node, which leads us to consider an average of both those CPUs for those simulations (these CPUs were distributed 50/50 among the total). The largest simulation, for Skeleton G, was specifically assigned to an EPYC 7401 node. For DLS, the simulations were arbitrarily assigned to any of the available nodes, being mostly attributed to an Opteron one. However in various cases the simulations were ran on an EPYC node. As such, for the DLS simulations we consider the weighted average benchmark score for all nodes, given that from a total of 672 CPUs, there were 192 EPYCs, and 240 of each of the Opteron types.

Table 6.7: A comparison of the single-core performance of the CPUs used in the HPCC for the simulations, and how they related with the performance of a typical laptop CPU (ratio).

CPU	Max Benchmark Score	Ratio
Intel i7-7700HQ	5341	1.0000
AMD Opteron 6180 SE	1615	0.3024
AMD Opteron 6344	2233	0.4181
AMD EPYC 7401	3853	0.7214
AMD 6180 SE & 6344 average	1924	0.3602
ALL AMD - weighted average (4:5:5)	2475	0.4634

The statistics regarding the whole procedure are summarized in Table 6.8. This table contains the number of postures and total samples ran for each skeleton (recall that each posture was simulated on 7609 target orientations).

It additionally contains various run-time statistics regarding the execution time to process a single sample (posture-orientation pair), and on the number of iterations that were ran. The execution time presented was corrected based on the ratios from Table 6.7 and therefore represent *measured time · ratio* in order to present all the statistics corrected as if they had all been ran on an average computer (taking an Intel i7-7700HQ as example).

Table 6.8: Statistics regarding the evaluation experiments with a total of $\sim$ 239M samples. Note that for the DLS cases, we present the total number of postures simulated, but the number of samples corresponds to the result of applying the filter explained in the previous section.

Skeleton		Number of		Iteration Count				Time per Sample (ms)			
	DoFs	Postures	Samples	Min	Max	Mean	S.D.	Min	Max	Mean	S.D.
A	3	33	251 097	1	7	4.14	2.17	6	126	45	28
B	4	377	2 868 593	1	10	3.42	2.19	8	189	45	30
C	5	3 789	28 830 501	1	12	2.31	2.08	6	1165	37	36
C-DLS100	5	3 789	20 561 499	1	100	63	43.6	1.60	2112	167	121
C-DLS200	5	3 789	20 561 499	1	200	120	92.8	1.57	3822	298	240
C-DLS400	5	3 789	20 561 499	1	400	234	191.6	1.55	7520	678	570
D	5	3 789	28 830 501	1	13	2.02	1.67	10	458	36	30
E	5	3 789	28 830 501	1	12	1.93	1.78	6	225	28	27
F	6	8 305	63 192 745	1	13	1.61	1.46	13	368	35	35
G	8	19 657	149 570 113	1	11	1.35	1.02	22	759	74	63

Analysis of Results: ERIK

After running the simulations on the different skeletons, we collected all the data and plotted the histogram for the error function and measures, as presented in Figure 6.30. Each line of the histogram figure represents an embodiment, from skeleton A to G, as indicated in the titles of the individual graphs. The first column of graphs contains the results for the value of the (combined) error function Λ_ϕ for each final solution. The second and third columns of graphs contain the final error for the individual measures $\epsilon_{\text{Orientation}}$ and $\epsilon_{\text{Posture}}$. At the top of the figures matrix we have placed the Legend, which applies to all the graphs.

Each graph shows the distribution of the error for all the solutions. The vertical axis represents the total count (frequency) of solutions that yielded a final error, given by the horizontal axis. Note also the dashed vertical lines, which represent the intended maximum error ($\Lambda_{\text{Threshold}\varepsilon}$), and also the solid vertical line, which aids in the visualization of the data, by representing the maximum error produced within the graph's samples. Note also that the range of the horizontal axis (error range) is the same in all rows except for the shaded ones in the first row, and that in the 5-link skeleton rows, each column presents the same Y value across all the three rows in order to help comparing between these cases.

Plotting the normal distribution of the error function results for each skeleton, provides further support on the interpretation of the results beyond the individual histograms, as illustrated in Figure 6.29. This figure shows the normal distribution for the combined error and for each of the error measures, for each of the skeletons except for A, which, due to its large error, disrupts the presentation of the others (and does not provide a significant interpretational value). The general interpretation taken from the normal plots is that all skeletons performed well regarding the Orientation Measure, and that the performance on the Posture Measure increased with the number of DoFs in the skeleton. Detailed interpretations will follow below.

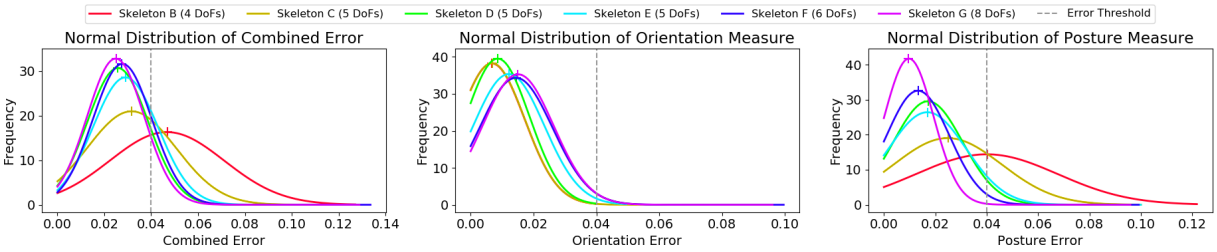


Figure 6.29: Normal distribution plots of the final combined error and error-measures for each skeleton except skeleton A.

Results for Skeleton A

These show that the target orientation goal failed immensely. This was highly expected given that the mechanical limitations of its joints, with only one *pitch* joint, limited to $[-\frac{\pi}{2}, \frac{\pi}{2}]$, would not allow it to aim at any orientations below the horizon. We also see that the posture errors do not seem so bad - that is because being a single-segment embodiment, the single and only posture it can perform is a straight line. Given that the whole corpus of experiment data would also generate only straight postures to be tested, it ended up not performing *so bad* there. Despite that, this case was meant to test if the algorithm reflected the expected results on such a constrained embodiment, with nearly no possibility of performing expressive postures while aiming at a given direction. The results confirm our hypothesis.

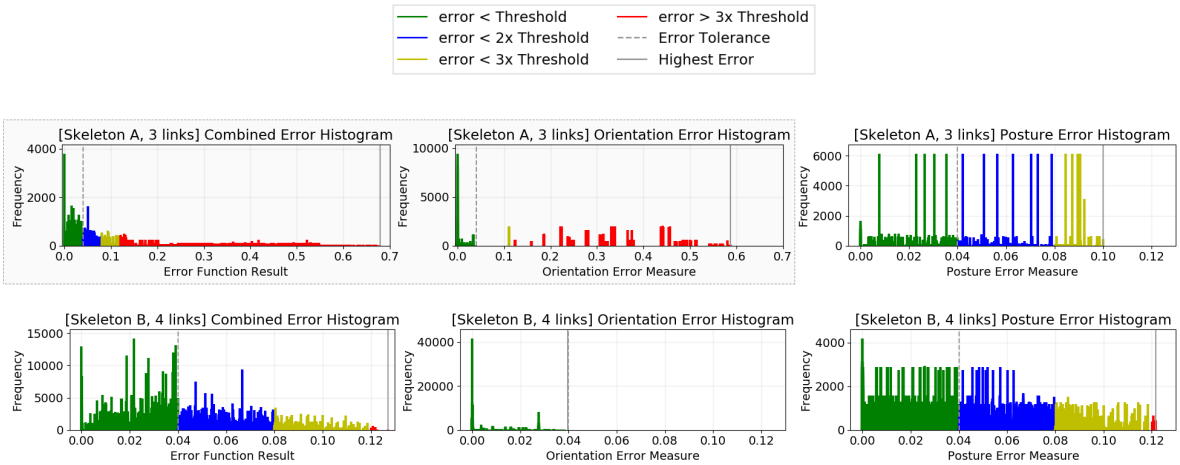
Results for Skeleton B

These show a substantial decrease in error compared to Skeleton A. By adding one more DoF, and allowing each DoF to have a higher range of motion, the skeleton was able to aim even at orientations below the horizon, as can be seen in its Orientation Error histogram, which always produced an error below the threshold. However, in order to achieve all the target orientations, the posture goal was largely missed, as seen in its Posture Error histogram. The normal plot shows that the error for Skeleton B (in red) was largely distributed beyond the specified threshold on the Posture Measure, and consequently on the Combined Error. However the graph for the Orientation Measure shows a good performance (its curve overlaps with the one of Skeleton C, in yellow). Still this skeleton does not represent a useful use-case for ERIK - instead it provides further support over the validity of the algorithm and its evaluation, as its *bad* results go in line with our expectation.

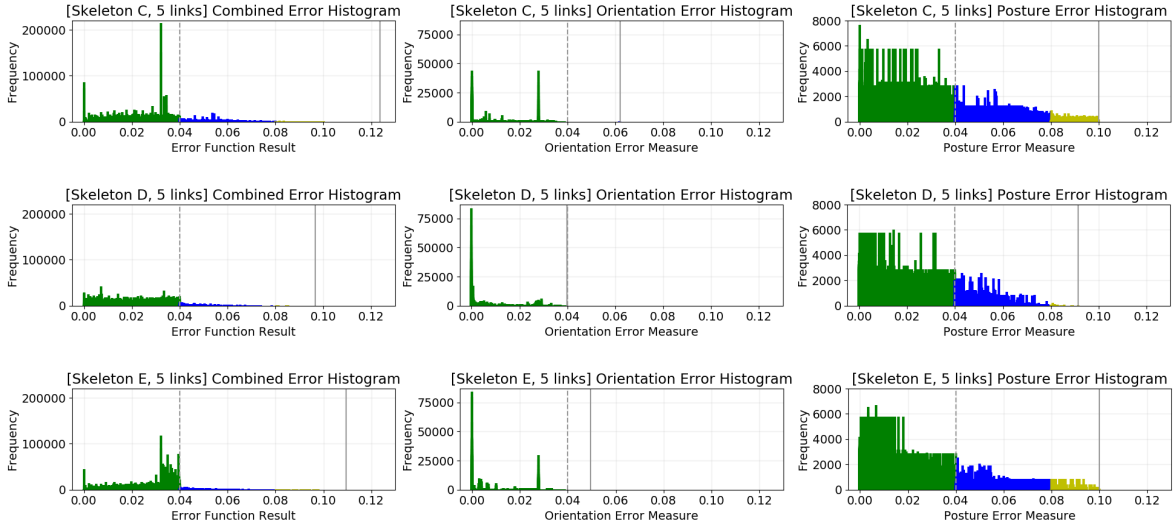
Results for Skeletons C, D, E

By adding another DoF, these results show lower error values compared to those of Skeleton B. We can note that in particular, the maximum Posture Error has decreased, meaning that the extra DoF provided the character with the ability to perform more expressive postures towards any direction. In C and E, some Orientation Error outliers have however produced an error above the intended threshold. However, it seems that there were very few of these situations, which makes them nearly imperceptible in the graph, if it wasn't for the *Highest Error* line.

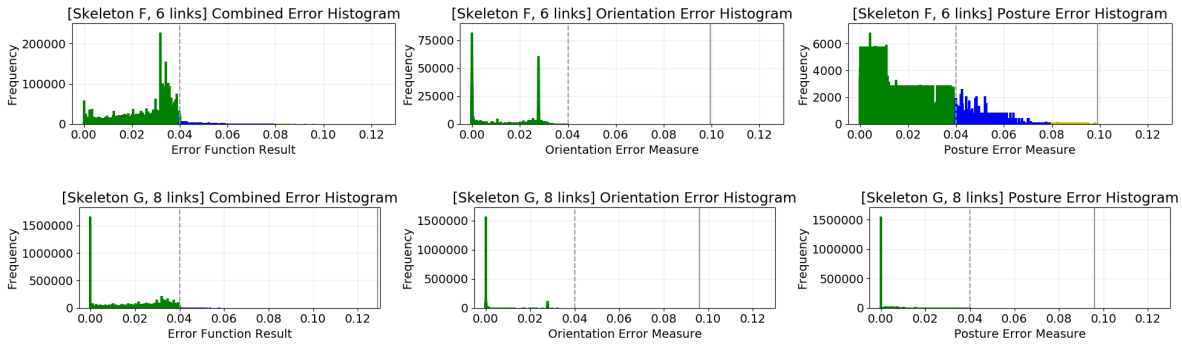
These skeletons start to yield results as we expect: to successfully orient to any given direction, while holding an arbitrary expressive posture that is allowed to slightly distort in order to ensure the prioritized orientation constraint. We take these conclusions from the Orientation Error histogram, which contains only some outliers beyond our



(a) Results for skeletons A and B, the 3- and 4-link skeletons, which perform below expectations, but contribute to verify that the algorithm results the expected results for those cases.



(b) Results for skeletons C, D and E, the three different 5-link skeletons, which start to yield satisfactory results.



(c) Results for skeletons F and G, the 6- and 8-link skeletons, which return the most satisfactory results.

Figure 6.30: Results of ERIK's evaluation process. Each line corresponds to one of the seven skeletons used. The columns correspond to each one's Error Function result, Orientation Error and Posture Error. Please note that the legend above the graphs applies to all of them.

given error threshold, and by the fact that the majority of the Posture Error is within the threshold, and that the ones that were distorted beyond it are contained within at most 3x that threshold, with frequency decreasing as the error increases.

Is is however interesting to perform a comparison between these three 5-link skeletons. Although they all have the same number of DoFs, they are configured in different ways. As seen in Table 6.5, skeleton C has an YXXZY configuration, while skeletons D and E use an YXZXY configuration. Furthermore, the angular limits of skeleton D are $[-\pi, \pi]$ while skeletons C and E have limits $[-\frac{\pi}{2}, \frac{\pi}{2}]$. The normal plots make this comparison more explicit. It becomes clear that from these three (C in yellow, D in green, E in cyan), all performed approximately well in the Orientation Measure, with Skeleton D performing best in the Posture Measure and the Combined Error, where Skeleton C performed worst. This draws the conclusion and illustrates that 1) a different joint configuration such as between C and E affects the performance, with, in this case, the layout of E providing better expressive capabilities than the one of C, while also showing, as expected, that by providing a wider range of motion, as in D versus E, that D, the one with the wider motion, can also perform better.

Results for Skeleton F

By introducing just one additional DoF as compared to C, D and E, the algorithm increases its performance. It is interesting here to compare in particular skeleton F to skeleton D, being that F has a lower angular range than D, but an additional DoF. While it may seem unclear from the histograms which of the two performed best, the normal plots does elucidate that Skeleton F performs better as seen in the normal distribution of the Combined Error, and of the Posture Measure. Interestingly skeleton D performed better in the Orientation Measure, however both performed within the threshold.

Results for Skeleton G

Finally, Skeleton G, with 8 links shows the best results as can be clearly seen in both the histograms and the normal plots. Again, through the normal plots it is seen to be not the best performer on the Orientation Measure, however, its ability to perform well on that measure, and perform exceptionally on the Posture Measure make it the best from this case set.

Results Comparison

The results presented here confirm our initial hypothesis that, as long as an embodiment has enough DoFs, it is able to use ERIK orient its endpoint towards any given target orientation, while successfully portraying a given expressive posture with minimal disruption. We group the results in three groups. Skeletons A and B can be seen as proofs of concepts, that serve to show that the algorithm fails when and how we expect it to fail (in highly constrained skeletons, with very few DoFs). Skeletons C, D, E and F are representative of cases where the algorithm starts to show positive results - with 5 or 6 DoFs it is mostly able to comply with all the constraints we have imposed, such as the joint limits and the error threshold, when solving for an integrated posture-orientation goal. Finally, skeleton F, with 8 DoFs already represents a case where the problem is solved in the most acceptable way, with both error measures performing below the threshold for nearly all the tested samples.

Analysis of Results: ERIK vs DLS

The same procedure was followed for analysing the results of the DLS technique. Figure 6.31 shows the normal distribution plots of the errors compared to the ones of ERIK with the same skeleton. Here we find that despite the improvements, there was nearly no difference in the general distribution of the errors across the different variations of DLS. In fact the three curves nearly overlap and become indistinguishable. We additionally detail the mean value and standard deviation for each of the cases in Table 6.9, which shows that there was a very slight improvement in the errors as the maximum number of iterations was increased.

In particular, and as we had already foreseen, ERIK performed better in achieving the correct target orientation. What we were most interested in finding out was how the posture error of the DLS would perform. Here we find similarly shaped curves for both ERIK and DLS, although ERIK's curve is centred around a lower mean error, thus revealing that it did in fact also perform better than DLS on solving the the posture target.

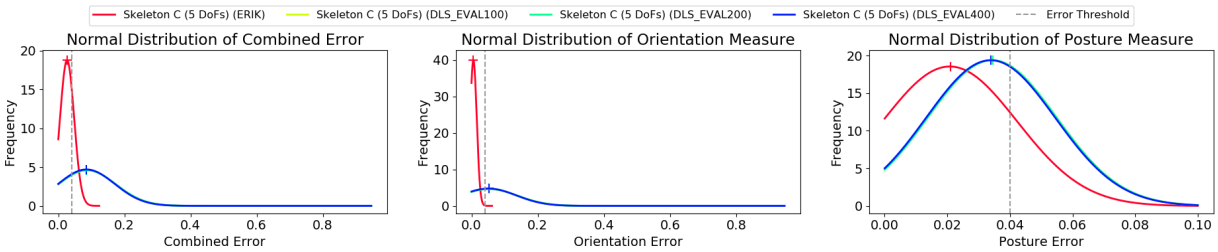


Figure 6.31: Comparison of the normal distribution plots of the errors for each DLS version and for ERIK Skeleton-C.

	Combined Error		Orientation Error		Posture Error	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
ERIK	0.026614	0.021258	0.005819	0.009977	0.020795	0.021520
DLS100	0.087067	0.086930	0.052744	0.085997	0.034323	0.020593
DLS200	0.086973	0.086897	0.052652	0.085980	0.034321	0.020591
DLS400	0.086831	0.086581	0.052523	0.085669	0.034308	0.020579

Table 6.9: Mean value and Standard Deviation for the ERIK and DLS variants comparison.

6.3.11 Discussion

The ERIK technique is a promising new step in the field of character animation, especially for robots and other interactive and immersive characters that are driven by AI. When driven by such AIs, and/or subject to stimuli such as user perception, it is important that the character animation engines for real-time, interactive characters, are able to process the flow of information that arrives through its sensors, and use it to influence and drive the character's behavior and animation. Our work takes an important step in that direction as the results support our initial claim that ERIK is able to provide expressive inverse kinematics solutions in real-time which simultaneously solve for an expressive posture goal, and for a target orientational goal.

Aiming at characters that are driven in real-time, and need to be expressive while also using their body to interact, such as gaze-tracking a person or object, ERIK succeeds in tackling both goals simultaneously for the majority of the situations. It was expected that by having more DoFs in the embodiment, both goals could be solved with lower error measures. In average, for a 5-link skeleton, the algorithm took 34ms to calculate a solution, and

74ms for a more complex 8-link skeleton, yielding a solution rate of 30 and 14 Hz respectively. While it would be desirable to have higher performance rates, our own implementation has show it to be adequate for real-time applications, as long as the IK solver is not synchronously running with the output module. By using the Nutty Motion Filter on the output module to smoothly interpolate the IK solutions in real-time, we are able to achieve smooth, sustained motion that can be used in such applications.

We have tested various skeleton configurations and ran extensive simulations in order to validate our claims. It is arguable how such an evaluation should be performed, however, in order to provide a general view, we opted out of evaluating the use of a robot using ERIK in a particular application with a smaller set of expressive postures, as that would also confine the validity of any conclusions to that single embodiment and set of postures. We therefore outlined the requirements that should be met by the algorithm to allow it to be used in any application, with any posture and with an arbitrary skeleton layout. Instead of using a small set of animator-designed postures, we took a sample of all the possible postures that each skeleton would allow to design. Instead of measuring how well a result met an animator's expectations (which is a subjective evaluation), we measured how close the resulting postures were to the original posture in terms of shape, using a heuristic method (the Posture Measure). By ensuring that the resulting posture is similar to the original, which in a real-world application, would be given by an animator, we expect and claim that the animator would also find the resulting posture satisfactory.

We additionally compared the results of the ERIK simulations to the same simulations using the DLS technique with postural control as the secondary task. Results showed that ERIK performed better in both the individual orientation task and the posture task. We argue that the DLS simulations using the tested Skeleton C are prone to kinematic singularities, which may have not been properly addressed. Therefore the result comparison was made with a filtered version of the DLS results, in order to excluded samples that seemed excessively bad due to that issue. The filtered results still show a worse performance compared to ERIK. Furthermore, even if all the kinematic singularities were properly dealt with in every case, the DLS simulations performed significantly slower than ERIK. Therefore, for the problem that we specify, ERIK represents a substantial improvement against using the DLS technique with the secondary task for posture-control, given that:

- Using ERIK one can use any embodiment even if highly redundant, which would be prone to kinematic singularities using Jacobian methods;
- ERIK performs on average about 4.5x faster than DLS on a similar task.

Chapter 7

Case Studies

7.1 Architectural Studies

The theories, practices and technologies for robot animation presented in the previous chapter have been used through the years in various different HRI scenarios, using different robots. This chapter presents some of the most relevant applications that were developed based on the contributions of this thesis.

7.1.1 EMOTE

The EU FP7 EMOTE project[1] developed an autonomous empathic robotic tutor to teach topics about sustainable development to children in schools. This represents the first use of the SERA ecosystem, along with the first mature version of the Thalamus framework (which was initially created prior to EMOTE and SERA). A NAO torso robot [2] plays an educational video game called Enercities[3] on a large touch-table along with one or two children (Figure 7.1).

Figure 7.2 illustrates the components used in the EMOTE scenario. All components were developed as Thalamus modules, with Skene acting as the main central point of the system. As the earliest scenario using the SERA ecosystem, it still didn't use Nutty Tracks as the animation system. Instead, a specific NAO Robot module was written to connect Thalamus with the NAO's api (NAOqi framework[4]). However, this scenario was developed during the whole course of the EMOTE project, and as such, it was also a sandbox for experimentation on, for example, the role of Skene, and what kind of perception information could be generalized in such a system. Skene was used to manage all the gazing, both between the robot and the children, and also towards specific on-screen targets. Because it includes a model of the environment, it is able to translate screen coordinates provided by the Enercities Game (in X,Y pixel coordinates) to angles that are then used to generate gazing or pointing commands. Because NAO includes its own TTS, there was no need to include it as a separate component.

Two lavalier microphones were used for more accurate perception of when and which student was speaking, so that the robot could properly gaze towards that student, or wait for silence before starting any speech behavior. The

[1] EMOTE project: `http://www.emote-project.eu` (accessed January 12, 2019)
[2] NAO Torso Robot: `http://doc.aldebaran.com/2-1/family/nao_t14/index_t14.html` (accessed January 12, 2019)
[3] EnerCities: `http://www.enercities.eu/` (accessed January 12, 2019)
[4] NAOqi: `http://doc.aldebaran.com/1-14/dev/naoqi/index.html` (accessed January 12, 2019)

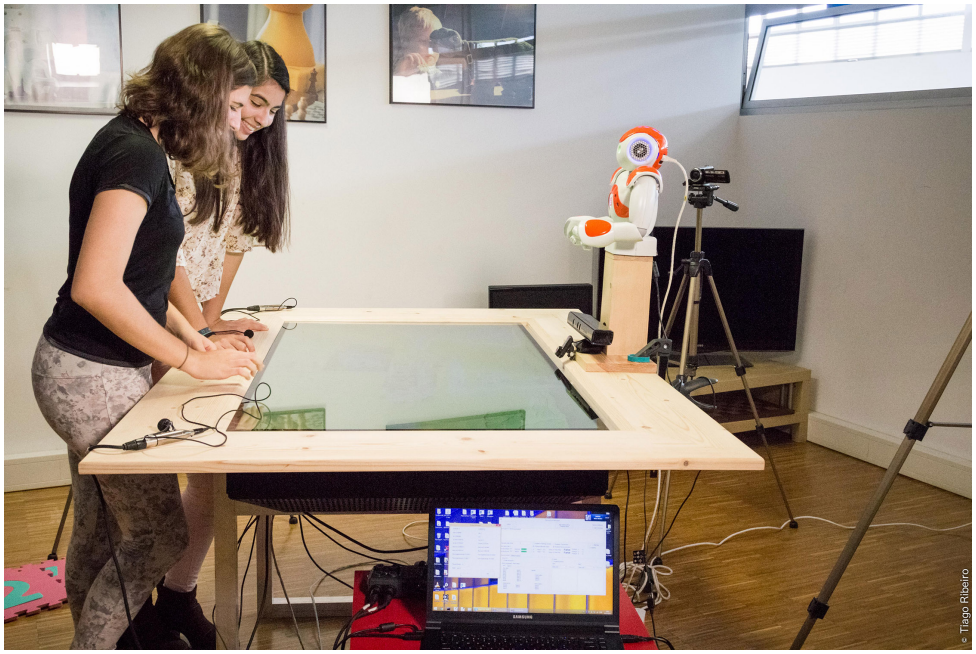


Figure 7.1: The physical setting of the EMOTE Enercities scenario.

specification of perception messages however, are abstracted in such a way that, in the absence of the microphones, such role can be performed by the Kinect, even if with less accuracy.

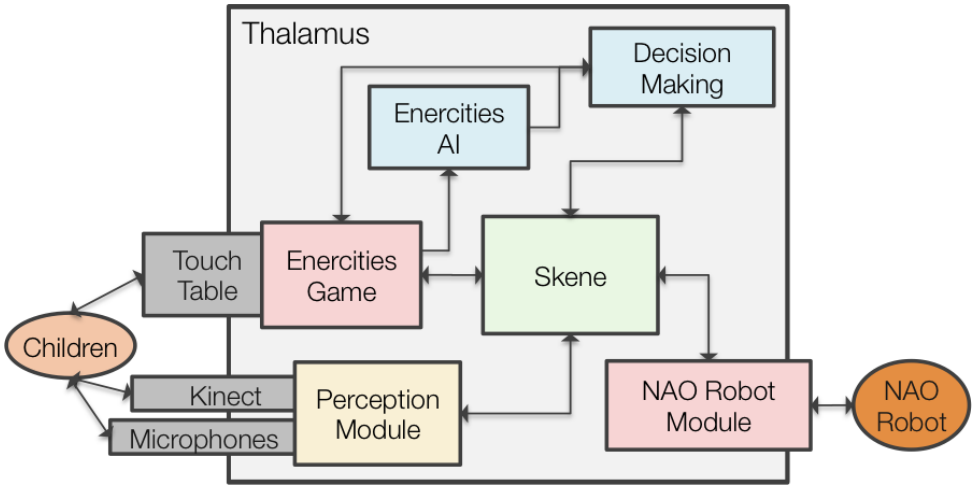


Figure 7.2: System used in the EMOTE scenario. Coloring of the Thalamus components are meant to match the ones of the SAIBA and SERA models (Figures 3.1 and 5.3)

.

7.1.2 E-Fit Keepon

E-Fit is an adaptive HRI application that keeps early adolescents engaged in physical activity over long periods of time [116] (Figure 7.3). The Keepon robot [175] interacts with participants once a day for approximately 5-10 minutes. Each day, the robot asks a series of questions and collects data from an off-the-shelf fitness sensor worn by participants. The robot's back-story unfolds over time. It is a robot-alien, named EfiT, that landed on Earth and needs the adolescent's help to return home. If the user accomplishes daily physical activity goals he will be helping

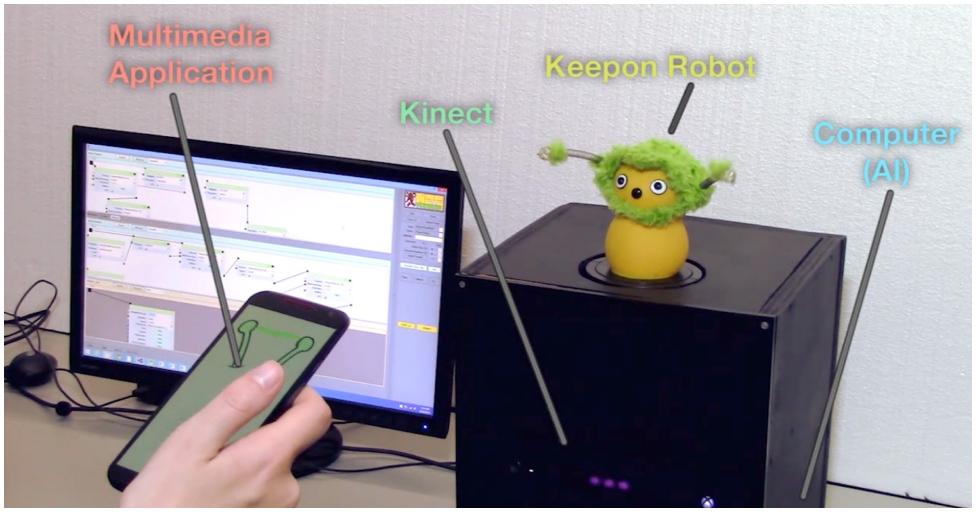


Figure 7.3: The E-Fit scenario.

the robot get back to its home planet.

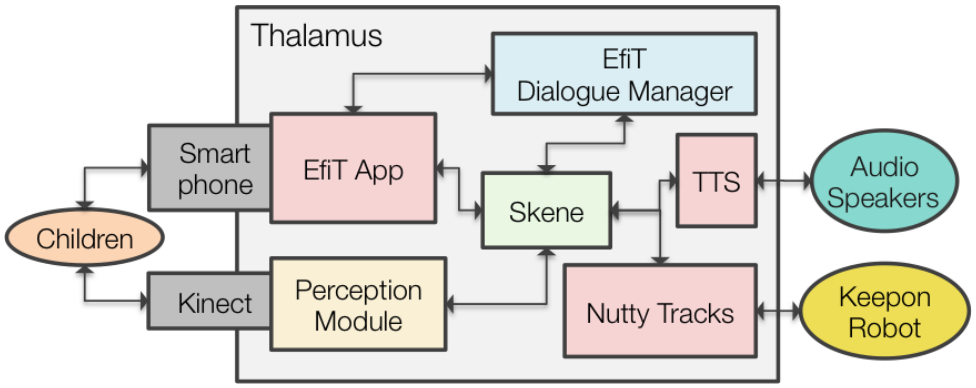


Figure 7.4: System used in the E-Fit scenario.

This project was a collaboration with Yale University's Social Robotics Lab[5]. Figure 7.4 shows the structure of the E-Fit scenario. Following the SERA model, it looks similar to the EMOTE project's architecture in Figure 7.2. The major difference is that the decision making is performed solely by a Dialogue Manager, while in EMOTE there was a separate AI just for calculating game moves. An EfiT App also runs on a smart-phone, instead of a touch-table and provides task-related interaction to the children. The Perception is simplified to using the Kinect only for face-tracking, because contrary to EMOTE (where microphones are placed on the students), in this scenario children interact freely with the system. Finally, the NAO robot is replaced with the Keepon robot, which was now controlled using Nutty Tracks.

7.1.3 Sueca

In the Sueca scenario (Figure 7.5), the EMYS robot plays a traditional Portuguese card-game called Sueca[6,7] [176]. This HRI scenario is aimed at the elder population, where the Sueca card game is very popular. As such, both the game and behavior of the robot were designed following an initial user-center design study lead by psychologists,

[5] `http://scazlab.yale.edu` (accessed January 12, 2019)
[6] Sueca: `https://en.wikipedia.org/wiki/Sueca_(card_game)` (accessed January 12, 2019)
[7] Sueca-EMYS video: `https://vimeo.com/153148841` (accessed January 12, 2019)

Figure 7.5: The physical setting of the SUECA scenario.

involving members of the elderly population [177, 140]. Figure 7.6 shows the components of the scenario. Similarly to the previous scenarios, the system relies on Thalamus, Skene and Nutty Tracks to integrate and manage the behavior of the EMYS robot. As in the EMOTE scenario, the decision making was split between two components: because Sueca is a game, there is a Sueca AI dedicated to calculating game moves. Along with that, the main decision making is performed by FAtiMA [178], which includes social behaviour.

This scenario accommodates four simultaneous players (one of them is EMYS, the other three are humans), each sitting at one side of a touch table with fiducial marker recognition capabilities. It is played using real card games, using a custom deck in which each card contains a specific fiducial marker on each side. After a player deals the cards to each player, each of EMYS's cards is placed facing down on the table in order for the fiducial markers to be recognized. That way the robot is informed of the hand that was dealt to it. Those physical cards are placed aside, and throughout the game, EMYS plays with a virtual version of them. All the other players however, still play with the real physical cards, which upon being cast on the table, are immediately recognized by the game and inform EMYS on what has been played.

The Sueca game is played in teams, being that each opposing pair of players play together against the adjacent ones. As such, an external Perception component to track all the human players has not been included yet, given the wide angle it would have to track. EMYS therefore relies on contextual information to know who's turn it is, and uses that information to, for example, gaze or speak at the current player. It also uses all the information and events provided through the Sueca Game application to know that the player has touched the screen (and generate gaze commands), or performed a game move.

A further evolution of the Sueca scenario was later developed which features two EMYS robots playing in the same game, along with two human players (Figure 7.7) [179]. One of the robots is still called Emys, while the other is called Glin. The overall architecture of the system is pretty similar as can be seen in Figure 7.8, however it was mostly duplicated so that each robot was independently controlled while still sharing the same activity (the Sueca

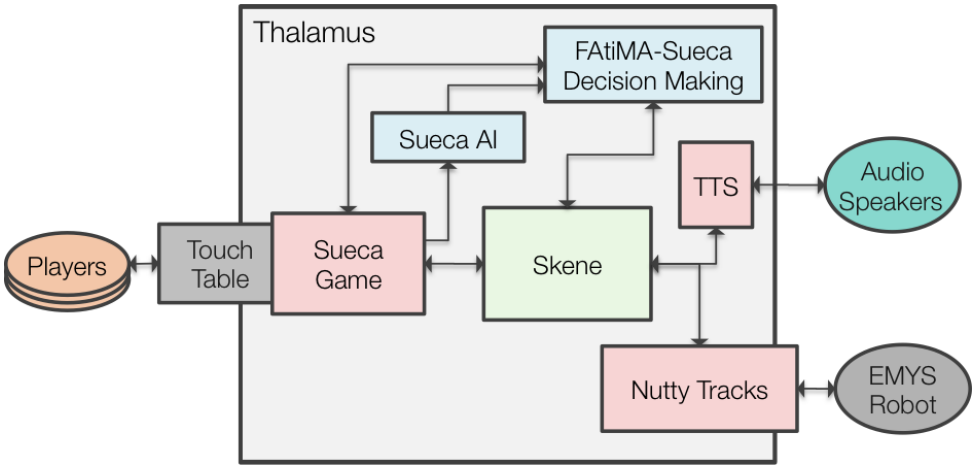


Figure 7.6: System used in the Sueca scenario.

Figure 7.7: The physical setting of the Mixed Teams SUECA scenario.

game). For that, two independent Thalamus characters were launched, and the Sueca Game module connected to both, thus acting as a bridge between them. This way for each robot was controlled by its own Decision Making/AI, Skene and by its own Nutty Tracks, while also using their own TTS voice.

7.2 User Studies with ERIK and Adelino

Upon developing ERIK and evaluating it algorithmically, we realized that further evaluation with users should be taken to assess how well a real system that uses ERIK would be able to stand up to our expectations. Because in most cases there exists no solution that simultaneously satisfies the target posture and target endpoint orientation, the computed solution will tend to satisfy the orientation constraint, while slightly allowing to alienate the intended expressive posture.

In the next two section we present two user-studies that were developed to address particular questions regarding the use of ERIK in real-world applications. In both scenarios we used the Adelino robot, which has a 5-dof manipulator-like embodiment, and an abstract face. Its kinematic structure is the same as Skeleton C in the ERIK

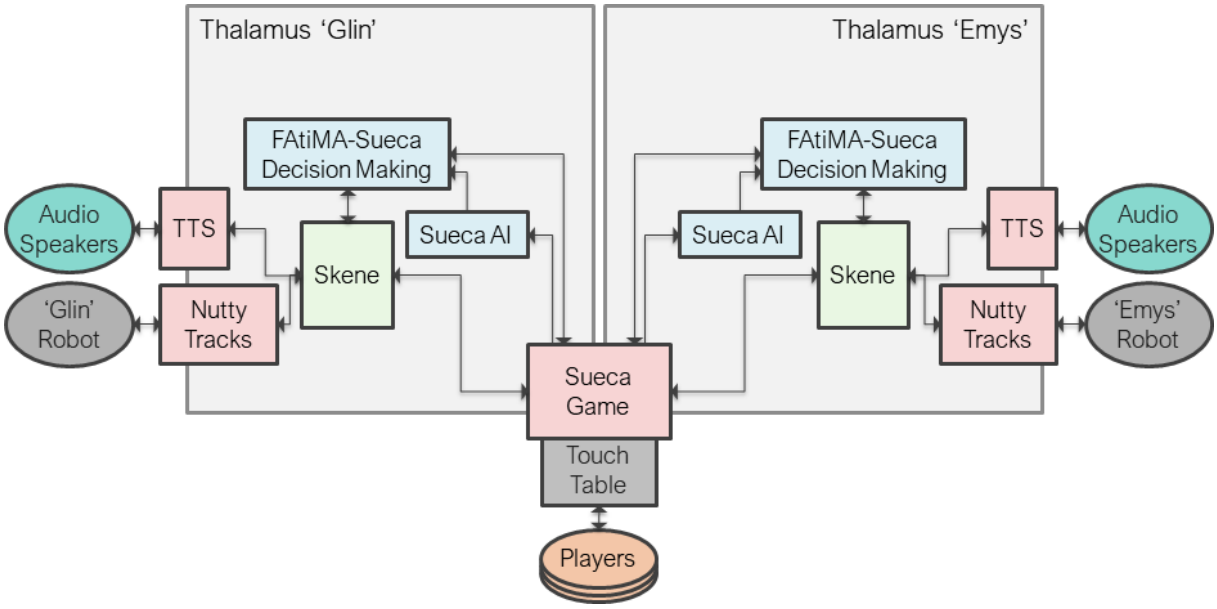


Figure 7.8: System used in the Mixed Teams Sueca scenario where two independent EMYS robots interact with two human players through a same game application.

Evaluation (Section 6.3.10).

7.2.1 Adelino, The Craft Robot

The Adelino robot was designed and built in order to fully challenge and test the capabilities of ERIK, with several goals in mind:

- Demonstrate ERIK in a custom built robot;
- Demonstrate ERIK with an autonomous craft robot;
- Promote the design and creation of craft robots for Do-It-Yourself (DIY) audiences;
- Provide animation software for Craft/DIY robots;
- As a learning and exploration exercise, to understand how to design and build robots that balance expressivity and affordability, in order to promote them to animation artists in the future;

We started by designing the concept of the robot using 3d animation software, as was previously done by Hoffman & Ju [96]. This can bee seen in Figure 7.9. The robot was designed as a line shape, as it is common for traditional and 3D animators to start their learning process by animating *lines of action*. These lines of action are "the first line indicated in a pose, that shows the basic overall posture, prior to adding the rest of the details" [154].

The lines are used to design the *attitude poses* of a character - ones "that convey what a character is feeling while he's moving". We can think of the line of action of a humanoid character to be its spine, grounded by its dominant leg. Based on this concept, we created a robotic line of action, with a minimum amount of degrees of freedom (DoFs). We added however, a face-like tip on the line in order to allow it to express gazing behaviour.

After our design concept phase, during which we tested some 3D animations with the character, we settled with five DoFs. In order to overcome typical low-end servos' limitations of rotating only 90 degrees to each side, the head was designed to be nearly symmetric along its vertical axis. Instead of having a fixed up or down side,

Figure 7.9: The concept design of the Adelino robot. It was initially modelled and animated using 3D animation software, to explore the size and placement of each segment and articulation, in order to maximize its expressive capabilities. Note that in this image, at some points, there are articulations that can rotate more than 90 degrees to each side. This feature was not maintained in the actual robot due to typical servos' limitation.

depending on the target posture and gaze direction, the robot could twist and turn upon itself, using the head "upside down", while still having the same appearance.

The actual robot is pictured in Figure 7.10. Its structure was hand crafted using balsa and pine stripwood, screws, washers, nails, some aluminium wire, one bearing, a hammer, a saw, and a drill/screwdriver. It is controlled using five hobby-grade servos connected to an Arduino[8], and an extra 5V, 2.0A power supply to feed the servos. The servos were chosen in order to be the cheapest ones that could handle the expected load. The commonly available and low-end motors are, however, restricted to 180 degrees of motion, thus allowing each joint to rotate only 90 degrees in each direction, given a rest pose that shapes the robot to the form of a vertical line. It also contains two small LEDs on the tip, allowing it to act as a face, so that it can portray the impression of gazing towards a given direction by pointing the tip towards that direction. In our vision, this *organo-tech* look, with exposed electronics (servos and wires), actually adds a more life-endued feeling to the robot in addition to its organic feel, as it breaks away from seeming a mere sculpture or piece of furniture, to become a creature that was gifted with some force and vitality (*anima*), especially as the wires' appearance resemble veins or a nervous system (which in fact, they ***are***).

Our motivation on building such a robot was not to make it a prototype, nor solely to lower the cost of its production. We wanted to build and show a robot that could appeal to non-technical audiences, such as animation artists, and to follow a construction process that did not require complex machinery or 3D printing.

In the future, we want artists to be able to design and build their own robots following on Adelino, and to be able to change the design and improvise *at will*, without having to go back and forth between CAD software and 3D

[8] `https://www.arduino.cc` (accessed January 12, 2019)

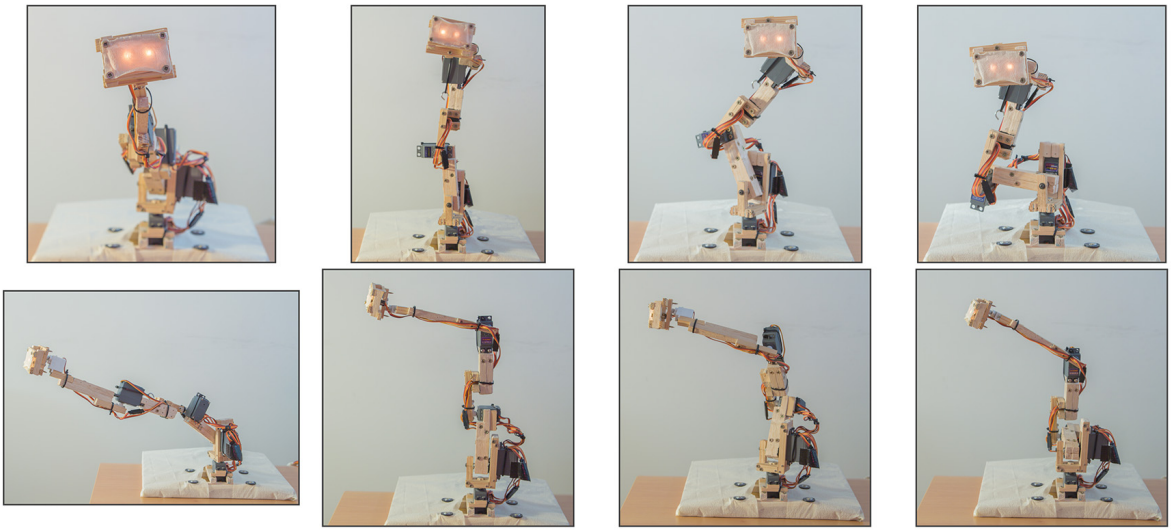


Figure 7.10: The Adelino robot, in four different expressive postures. The top row shows front views of each posture. Under each is the corresponding side view.

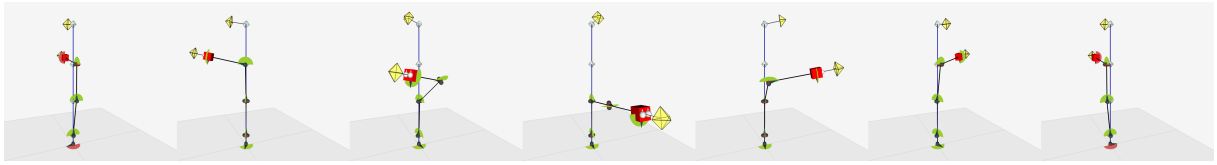
printers, nor any other technical machinery. Using craft materials, they can cut pieces with different lengths, shapes, paint it, or glue anything to it, with more creative freedom.

At the same time, because our goal is to challenge and test our algorithm, we wanted the final evaluation to heavily rely on our software, and not on the use of expensive motors or precision machinery. As such, one of the major motivation for our craft approach was to demonstrate ERIK using the lowest-quality robot we *could* build. That way, we argue that there is only space for improvement on the resulting animation quality, given that the motors, structure, and build process can all be enhanced. Although the robot exhibits a somewhat *unintentional* shaky behaviour, we considered that such shakiness could become part of its own unique *character*, and aesthetically, to actually contribute to a sense of lifelikeness.

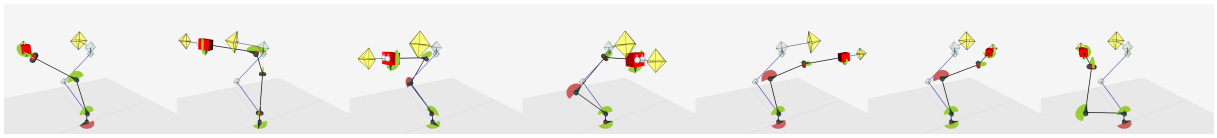
Given that the ERIK algorithm is proven to yield reliable results in terms of inverse kinematics, we decided to proceed to use Adelino on user-studies given that despite its shakiness, it still seemed to manage to convey the expressive postures as intended (which we further confirmed in the following two sections). Furthermore, we designed animations and postures in a way to ensure that the intentional expressions of the robot were clear and explicit, even slightly exaggerated, so that they were clearly distinguishable from any involuntary shaky motion.

Using Adelino to illustrate the capabilities of ERIK, and taking as example Figure 7.11, given an expressive posture, the character is able to gaze towards different directions while attempting to maintain the given expressive posture. This figure demonstrates it by showing the virtual view of the robot's skeleton, holding an expressive posture while shifting the orientation target, as if the robot was gaze-tracking the user with an expressive stance.

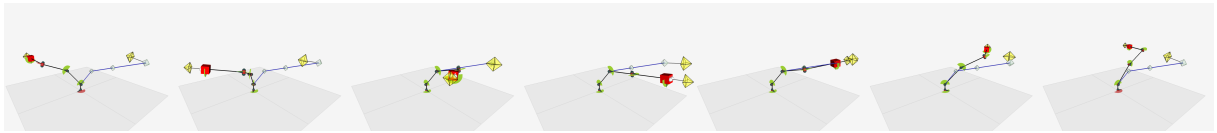
In another demonstration, Figures 7.12 and 7.13 show the robot holding an orientation while shifting between postures, as if the robot was gazing towards a stationary user or location, and solely shifting its expressive stance. Note that during an interaction, the transition between postures may even give an impression that it was pre-designed. However, with pre-designed animations it would not be possible to maintain gaze-contact with the user.



(a) A *neutral* stance from -180°to 180°in 60°steps.

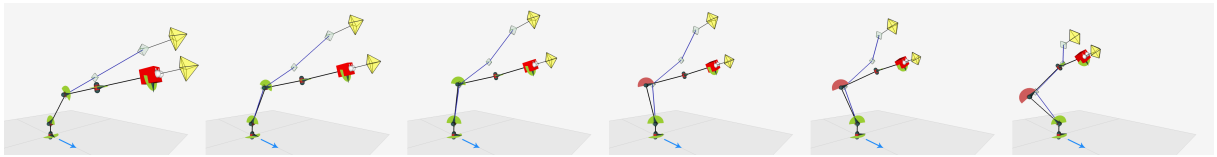


(b) An *unengaged* stance from -180°to 180°in 60°steps.

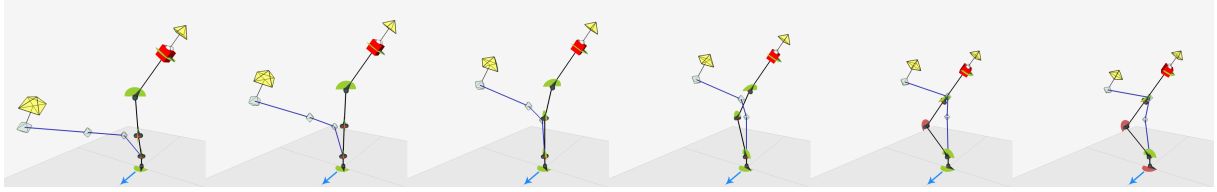


(c) An *engaged* stance from -180°to 180°in 60°steps.

Figure 7.11: Demonstration of the ERIK algorithm on Adelino's skeleton. Three different expressive postures are presented. For each posture the ERIK orientational target was swept 360°. The images were captured directly from ERIK's visualizer in Nutty Tracks. The blue lines represent the target expressive posture which remains fixed throughout each frame. The yellow arrows represent the target orientation. The circular sector at each joint represents its rotational limits. Red means that the joint is at its rotational limit.

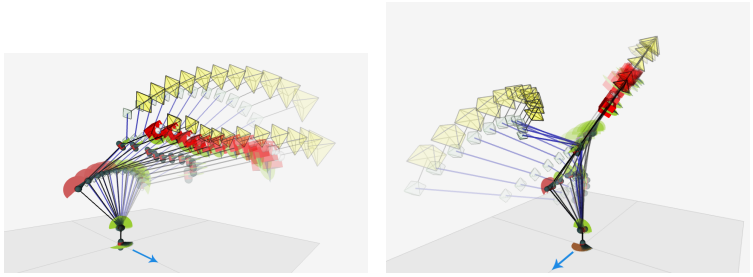


(a) A shift from an *engaged* posture to an *unengaged* posture in an *easy* angle, relatively towards the front of the character.



(b) A shift from an *engaged* posture to an *unengaged* posture in an *extreme* angle, upwards and towards the side of the character.

Figure 7.12: Demonstration of orientation hold during posture shifts. Two shifts of posture are shown, one for a common *easy* orientation, and another for a more complicated, *extreme* orientation. The blue arrow represents the character's frontal *direction*. The remaining elements are described in the caption of Figure 7.11.



(a) Stack of frames showing the shift from *engaged* to *unengaged* at an *easy* orientation.

(b) Stack of frames showing the shift from *engaged* to *unengaged* at an *extreme* orientation.

Figure 7.13: The accumulation of frames for each of the cases described in Figure 7.11. This *onion-skin* view helps to illustrate the motion of shifting between both postures.

7.3 Ahoy - The Pantomimic Expressive Manipulator

Our first user-study of ERIK was created to address the following research question:

> Can an expressive posture be encoded into a face-tracking manipulator-like robot, using ERIK, in order to convey a purposeful intention to the user?

With this question in mind, we formulated the design of a scenario in which participants would benefit of interpreting understanding the robot's non-verbal queues in order to progress or succeed in a given task.

For that we developed the Ahoy scenario, in which a human plays a pantomime game with a robot. On each round, the robot performs an animation that represents a word that must be guessed by the user within a time limit.

A Wizard (Controller) was given the role of selecting the expressive posture, which the IK algorithm tried to maintain while tracking the user, during the guessing phases. The user-study was performed using two different conditions:

C1 - Intention-expressive : During the guessing of the word by the user, the robot changes its posture depending on a hot-cold measure, in order to provide hints to the player about their guess;

C2 - Non-intention-expressive : The robot maintains the same posture ('neutral') throughout all the guesses.

No other verbal or nonverbal communication was used for the game.

Upon this, in order to address the research question, we have formulated the following hypotheses:

H1 The robot is perceived to play the game similarly well, in both conditions. *Measure: Performance Mean (1.*)*

H2 The robot's animation conveys the illusion of life, in both conditions. *Measure: Animation Mean (2.*)*

H3 In the Intention-expressive condition (C1), players perceive the robot's posture changes as its intention of providing hints to the player. *Measure: Intention Recognition (3.1.a)*.

H4 In the Intention-expressive condition (C1), players correlate the intended hint of the robot with their performance in the game. *Measure: Intention Legibility Mean (3.2.*))*.

7.3.1 Sample

For this study we had a total of 42 university students (22 males and 20 females) with ages ranging from 18 to 34 ($M = 22.71$; $SD = 2.95$). 31% of the participants had interacted with a robot before and 25% frequently played the pantomime game. Half of the participants were randomly assigned C1, and the other half to C2. One participant from each condition were later excluded due to not complying with the questionnaire instructions.

7.3.2 Procedure

Upon arrival to the experimental setting participants were given an explanation about the study and as they enter a room where the robot was, the robot would see him/her and start interacting. A word category would be projected on a wall, and the robot would pantomime it through an animation. As in a regular pantomime game, the participant should verbally keep on guessing the correct word until either a correct answer was performed, or until time was out (40 seconds per word). The robot would then let them know if they either got it right, or if they were unable to answer within the time limit, and would then move on to the next word, until it finished. In total there were seven rounds. They were told they would understand when the game was finished, and could then leave the room, as the experimenter would be waiting right outside. Upon the interaction, the participant was led to a private room to answer the questionnaires. At the end of the study, a lottery was ran to draw thank you gifts between the participants (12 movie tickets were drawn within the 42 participants). In C1, upon each guess, the robot changes its posture depending on a hot-cold measure, in order to hint the player about their guess; C2, in which the robot maintains the same posture ('neutral') throughout all the guesses.

In a physically separate room, two wizards (W1 and W2) teamed to replace an artificial intelligence capable of quickly assessing the hot-cold quality of the participants' guesses. This design was chosen due to the an open-ended game vocabulary, and because participants were not equipped with a wearable microphone. W1 listened to the player's guesses through a hidden microphone and based on a predefined list of words that are semantically similar to the correct guess, W1 would perform a measure of hot, warm or cold. The list for each answer contained approximately 30 words, ordered alphabetically, organized into hot, and warm words and formatted in order to provide a fast visual search by the wizard. They were gathered beforehand, both based on an online resource[9], and on the experimenters' intuition. Any word not on the list was considered to be cold. Upon assessment, W1 would verbally notify W2 of the result (hot, warm or cold), and W2 would use a simple WoZ interface to quickly trigger a new posture for the robot. Both wizards were previously trained by the experimenters, both informally, and in an initial pilot version of the study. Stable communication between the wizard-room and experimentation-room was guaranteed by an ethernet cable and audio extension cord (about 30 meters each).

The robot plays the pantomime game with a human, on a one-to-one interaction. A word category would be projected on a wall, and the robot would pantomime it through an animation. Participants had 40 seconds to guess the correct word by shouting it towards the robot, and could make an indefinite number of guesses. The robot would provide a visual feedback if they got it right (with a positive nod), or a negative nod in case the time was up. It would then proceed to the next round. In total there were seven rounds. In the end, they were led to a private room to answer a questionnaire.

[9] `http://swoogle.umbc.edu/SimService/top_similarity.html` (accessed January 12, 2019)

In a physically separate room, two wizards (W1 and W2) teamed to replace an artificial intelligence capable of quickly assessing the hot-cold quality of the participants' guesses. W1 listened to the player's guesses through a hidden microphone and based on a list, would assess and verbally communicate to W2 if the guess was *hot*, *warm* or *cold*. W2 was in charge of a control panel from where he could trigger (in C1) a new posture for the robot, depending on the participant's guess. The triggered postures, which have been illustrated in Figures 7.11 and 7.12, were designed to convey the impression of being hot (*engaged*) or cold (*unengaged*).

7.3.3 Measures

In order to investigate our research question and test our hypothesis, we considered a set of specific subjective measures created for our goal, another set of subjective questions taken from literature, and also objective measures collected during the interaction with the users. There were three types of specific subjective measures: ***Performance***, ***Animation*** and ***Intention***.

We considered questions concerning three types of specific subjective measures: ***Performance***, ***Animation*** and ***Intention***. ***Performance*** was included to assess how well the robot was perceived to play the game. ***Animation*** was measured looking at the following aspects: *Quality*, *Lifelikeness*, *Staging*, *Thought* and *Motivation*. ***Intention*** was measured using the perception of *Recognition* and *Legibility* of the motion.

Table 7.1 presents the questions used in each specific measure.

The subjective measures taken from literature were ***Perceived Message Understanding*** and ***Co-Presence***, from the *Networked Minds* questionnaire [180], the ***Inclusion of Other in Self*** (IOS) measure [181], ***Perceived Adaptability*** from the *Almere model* [182], and finally, the dimensions of ***Perceived Intelligence***, ***Animacy*** and ***Likeability*** from the *Godspeed* questionnaire [183].

All the questionnaire scales except IOS were answered in a 6-point Likert scale and when necessary, items were shuffled to mask for their dimensions. The IOS measure was answered in a 7-point scale.

The objective measures were collected to assess the effect of our technique in the participants' performance through the game. For each participant we collected the *Total Duration* of the activity, the number of *Total Guesses*, *Total Hot Guesses*, *Total Cold Guesses*, *Total Correct Guesses*, *Total One-Shot Correct Guesses*, and *Total Give-ups*. A One-Shot Correct Guess was counted when the participant got the correct answer in a round as its first guess. A Give-up was counted when the participant did not perform any guess in a round. Per each participant's round we also collected the *Duration*, number of *Guesses*, number of *Hot Guesses*, number of *Cold Guesses*, and number of *Correct* guesses (which would either be zero or one).

7.3.4 Results

To understand if our algorithm allowed Adelino to convey an intention while gaze-tracking users, statistical analysis was performed on the subjective data collected through the questionnaires, and the objective data collected during the interactions. The Shapiro-Wilk test was used to test of distributions are normal or non-normal. Where normal distribution existed, we performed a t-student for independent samples. When the normality assumption was not met we used a Mann Whitney U test.

Table 7.2 shows the results for all the subjective measures and sub-measures. We have highlighted the measures

1. **Performance**

   1.a. The robot was good at pantomiming the words.
   1.b. I was able to think of words that were perhaps being pantomimed by the robot.

2. **Animation**

   2.1. *Quality*

      2.1.a. During the pantomime, the motion of the robot seems natural.
      2.1.b. The robot's movement while I was attempting to guess were smooth and natural.

   2.2. *Lifelikeness*

      2.2.a. The robot seemed to be alive.
      2.2.b. The robot reminded me of the characters I know from movies.

   2.3. *Staging*

      2.3.a. The robot performed the pantomimes in a way that it was easy for me to see what he was doing.
      2.3.b. The robot was moving in tune with me while I was trying to guess the correct answer.

   2.4. *Thought*

      2.4.a. The robot seems to understand the concept of the words it was pantomiming.
      2.4.b. The robot thought of every word before it performed the pantomime.

   2.5. *Motivation*

      2.5.a. The robot was enthusiastic with my attempts to get the right answer.
      2.5.b. The robot wanted me to get the right answer.

3. **Intention**

   3.1. *Recognition*

      3.1.a. The robot gave me tips while I was trying to guess the right answer.

   3.2. *Legibility*

      3.2.a. I was able to understand, through the robot's tips, if I was far or close to the right answer.
      3.2.b. The robot's tips seemed coherent with my guesses.
      3.2.c. The robot's tips helped me to get the right answer.

Table 7.1: The questions used in each specific measure on the Ahoy study.

* reports significant differences		Shapiro-Wilk		Statistics				T-Test		Mann-Whitney	
Measure	*C*	*Sig*	*Normal*	*Min*	*Max*	*Mean*	*SD*	*t*	*ρ*	*U*	*ρ*
Performance	1	0.140	yes	2	6	3.875	0.872	0.983	.332		
Mean	2	0.334	yes	1	5	3.600	0.897				
Animation	1	0.576	yes	1	6	3.635	0.995	0.606	.548		
Mean	2	0.242	yes	1	6	3.455	0.879				
Animation	1	0.525	yes	1	6	3.400	1.273	0.888	.380		
Quality	2	0.075	yes	1	5	3.075	1.030				
Animation	1	0.128	yes	1	6	4.000	0.973	-0.335	.740		
Lifelikeness	2	0.132	yes	1	6	4.125	1.356				
Animation	1	0.280	yes	1	6	3.575	1.030	2.165	.037		
Staging*	2	0.445	yes	1	5	2.900	0.940				
Animation	1	0.072	yes	2	6	3.650	1.137	0.998	.325		
Staging.a	2	0.055	yes	1	5	3.300	1.081				
Animation	1	0.094	yes	1	6	3.500	1.469			123.5	.034
*Staging.b**	2	0.027	no	1	5	2.500	1.277				
Animation	1	0.093	yes	1	6	3.350	1.479	-0782	.439		
Thought	2	0.286	yes	1	6	3.675	1.127				
Animation	1	0.097	yes	1	6	3.850	1.299	0.877	.386		
Motivation	2	0.075	yes	1	6	3.500	1.225				
Intention**	1	0.393	yes	1	6	2.838	0.779	3.195	.003		
Mean	2	0.062	yes	1	5	2.088	0.704				
Intention	1	0.038	no	2	6	3.450	1.317			105.5	.008
Recognition**	2	0.021	no	1	5	2.350	1.040				
Intention	1	0.299	yes	1	6	2.633	0.864	2.569	.014		
Legibility*	2	0.122	yes	1	4	2.000	0.684				
Godspeed	1	0.021	no	1	6	4.058	0.660			117.0	.024
Animacy	2	0.179	yes	1	6	3.642	0.565				
Godspeed	1	0.087	yes	1	6	4.680	0.691	1.554	.129		
Likeability	2	0.169	yes	1	6	4.340	0.693				
Godspeed	1	0.612	yes	1	6	4.090	0.706	0.227	.821		
Perceived Intelligence	2	0.889	yes	1	6	4.040	0.686				
Networked Minds	1	0.010	no	3	6	5.48	0.557			195.5	.902
Co-Presence	2	0.006	no	3	6	5.47	0.533				
Networked Minds **Perceived Message Understanding**	1	0.221	yes	1	6	3.630	0.832	0.687	.496		
	2	0.243	yes	1	6	3.440	0.852				
Almere	1	0.502	yes	1	5	3.050	0.975	0.829	.412		
Perceived Adaptability	2	0.221	yes	1	6	2.820	0.798				
Inclusion	1	0.034	no	2	6	3.60	1.273			160.0	.268
Inclusion of Other	2	0.216	yes	1	6	3.10	1.447				

Table 7.2: Statistical data and significance test results for all measures and sub-measures, plus a granularization of the Animation Staging measure.

that reported statistically significant results ($p - value < 0.05$) by marking the measure with an asterisk (*) in the first column, and by coloring its ρ value in yellow. A double asterisk (**) reports a $p - value < 0.01$. The green-and-red coloured column indicates solely if the distribution is normal (green) or not (red).

Regarding the specific subjective measures, we expected to find significant differences in the Intention measure. However, we also expected that all the other measures would not be affected by our algorithm, supporting solely that the participants of C1 should perceive the robot to have the intention of providing hints, while those of C2 had not.

We immediately start by verifying that the *Intention Mean* measure (mean of both *Recognition* and *Legibility*), revealed a significant difference between the two conditions, showing that in C1 the hint-providing intention of the robot was perceived to be higher than in C2. However, given that the mean value for C1 was not very high we also analysed the *Intention Recognition* and *Intention Legibility* measures in separate. Both presented significant differences between conditions, with *Recognition* providing a much more accentuated result.

Neither *Performance Mean* nor *Animation Mean* presented significant results between conditions. That means that the use of the expressive postures in this scenario did not make the robot seem either better at playing the game, nor more animate, and both of these measures had positive results. When we broke down the *Animation* sub-measures, we did however find a significant difference between conditions for the *Staging* measure.

Regarding the non-specific subjective measures taken from literature, only the *Animacy* dimension of the *Godspeed* questionnaire reported significant differences between the two conditions. As to the objective measures, differences between conditions were reported only for the **Round**-measures *Duration*, *Number of Guesses* and *Hot Guesses* in Round 3.

Figures 7.15-7.16 summarize the objective data collected throughout the interactions. These graphs concern the average of all the complete sessions.

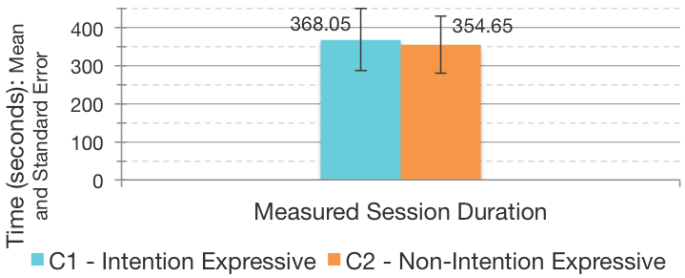


Figure 7.14: Objective data from the Ahoy study: mean duration of each session, per condition.

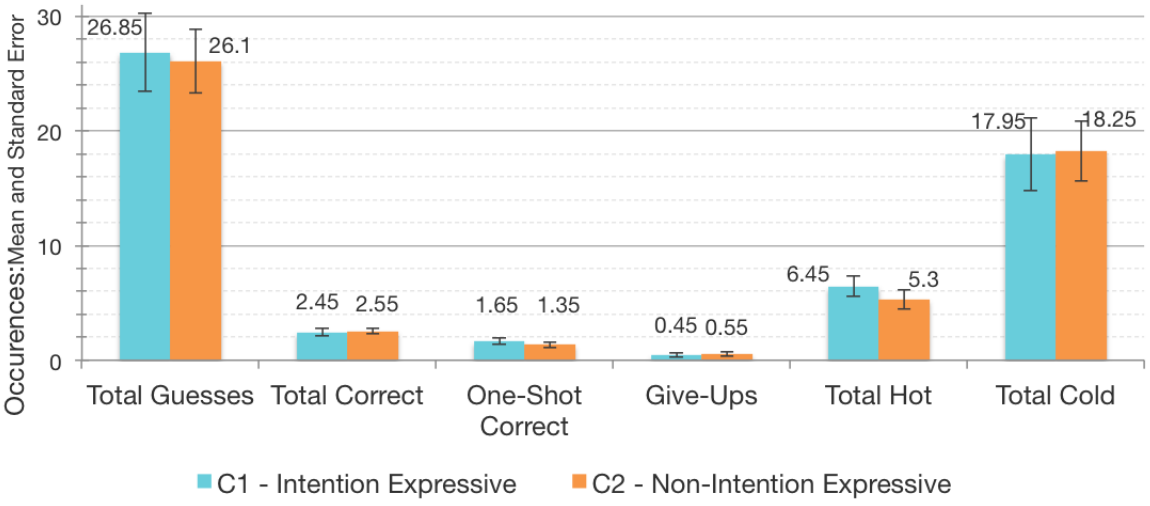
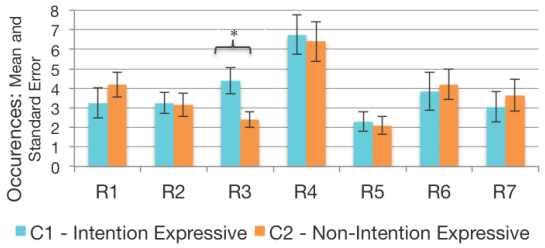
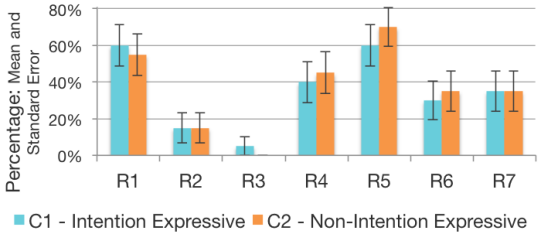


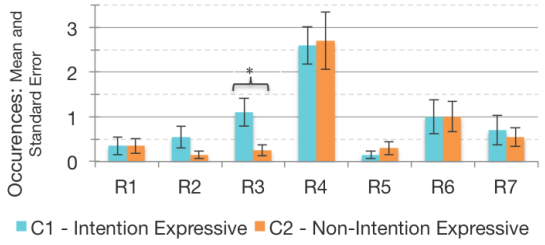
Figure 7.15: Objective data from the Ahoy study: Mean value of each objective measure (except *Total Duration*), per condition.



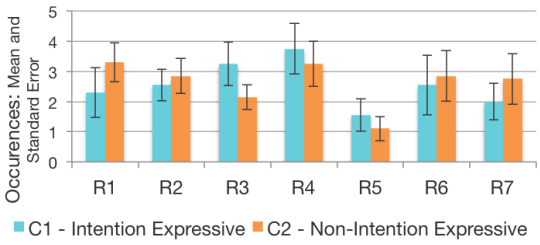
(a) Average number of *Guesses*, per round.



(b) Percentage of participants able to guess the *Correct* word, per round.



(c) Average number of *Hot* guesses, per round.



(d) Average number of *Cold* guesses, per round.

Figure 7.16: Objective data from the Ahoy study: analysed per round.

7.3.5 Discussion

The data collected through the Ahoy study provides evidence to supported all of our hypothesis. **H1** is supported given that there were no significant differences between conditions for the *Performance* measure, and **H2** for the *Animation* measure. In particular, related with H1, although the mean scores were not very high, we still found that the use of our algorithm did not bias the perception how well the robot played the game. H3 and H4 are supported as both the Intention-Recognition and Intention-Legibility measures rendered significant differences.

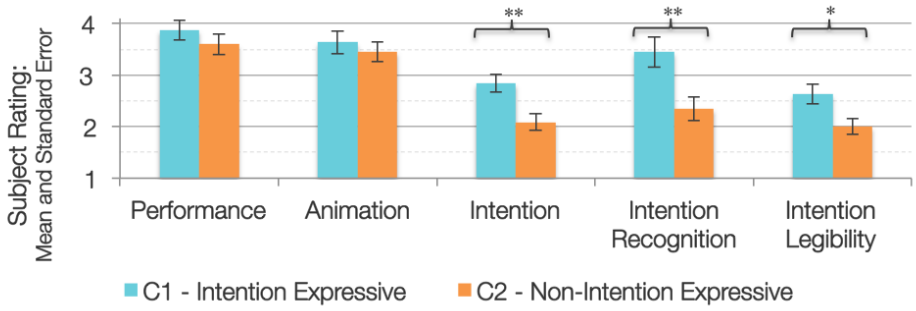


Figure 7.17: The results collected from the specific measures. An asterisk (*) reports a $p - value < 0.05$. A double-asterisk (**) reports a $p - value < 0.01$.

Subjective Measures

The results collected for the subjective specific measures are presented in Figure 7.17. Individually, we feel that most of these measures and sub-measures did not report strong values, i.e., the mean values are not very far from the scale's median value, and on some cases are even below.

Regarding the measure of *Performance*

The positive results from the ***Performance*** measure suggest that the pantomimed animations were well designed for the purpose (as it was perceived to play the game well). This measure yielded a mean of 3.74, which statistically might seem just slightly over the scale's mean. However, considering that this measure aimed at the robot's ability to play the game, we should interpret the results monopolarly, and not bipolarly, meaning that a total lack of ability would be reflected as the scale's minimum value (1). Therefore a value of 3.74 as a reflection of the robot's ability to play, albeit not being *great*, can still be interpreted as being good, thus still supporting **H1**'s statement of "play the game well".

Regarding the measure of *Animation*

The positive results from the ***Animation*** measure suggest that the robot was well designed, and that the IK algorithm (regardless of the demonstrated posture) yielded smooth, natural results, and that the dynamic gaze-tracking behaviour was well perceived. This measure yielded a mean of 3.55, just above the scale's mean, which can be considered as just enough to argue for the illusion of life. By analysing the sub-measures' means we find that *Lifelikeness* and *Motivation* both yielded above-average means (4.06 and 3.675 respectively), while the poor results of *Staging* in particular, brought the whole measure down. *Staging* had two questions (see Table 7.1) related to how the robot was positioning itself for the participant (thus addressing the *Staging* animation principle). We therefore separately analysed the results for each question and found a significant difference only on the second one. The first question addressed how well the animations had been designed so that the participant could understand them, and yielded good results with no significant differences across conditions. The second had been defined to address the quality of the gaze-tracking. However, after we found participants to rate this questions significantly lower in C2 than in C1, we realized that the participants in C1 may have interpreted the *movement in tune* as the posture changes that the robot performed in order to provide them hints. Moreover, because most participants would actually

stand still in front of the robot (as seen in recorded video), the gaze-tracking behaviour may have not been very apparent, and thus also lead to lower ratings in C2 (where the robot would not even perform posture shifts). Further investigation is would be required to properly conclude on this sub-measure. If we disconsidered this sub-measure, the whole Animation measure would yield a mean of 3.62.

The low results of the *Quality* sub-measure may have also been due to the rudimentary design and build of the robot, which causes a lot of jitter. However, the robot was purposely designed and built that way in order to challenge the sturdiness of the animation algorithms. Our initial intention was to see how good the software was with nearly the worst-quality embodiment, meaning that with simple enhancements (e.g. better quality servos), we would expect to see an increase in the *Animation Quality* measure.

Regarding the measure of *Intention*

While the measures and sub-measures related to Intention all yielded significant differences between condition, the overall values were not high. We can see that the sub-measure of Intention-Recognition had in general, higher results than the sub-measure of Intention-Legibility, meaning that it was more obvious for the participants in C1 that the robot was trying to provide them hints, than it was for them to relate those hints to their own guesses. Although these results did support our hypothesis, further development and investigation is required to understand what can be done to allow the robot to convey a stronger intentionality through its expressivity.

Some participants were informally approached after completing the experiment and filling in all the questionnaires. While no formal interview was made, several participants mentioned that there was some latency between their guesses, and them noticing the change in posture of the robot. Given the nature of the pantomime game, they would expect the robot to respond immediately as a human did, and as such, did not always connect the robot's movement with what they had just guessed. Some unstructured observation was performed during the preliminary pilot testing sessions where we accessed the robot's latency in response to a guess, to be around 1 to 2 seconds, depending on the participant's guess. While we did design a double-wizard setting in order to minimize this latency, it was still faster for the Wizards to recognize that a word was hot (i.e. find it in the list), than it was to access that it was cold (i.e. finding that it was not on the list).

Although we theorize that an autonomous system could have mitigated the latency, we went for a Wizard-of-Oz solution due to the open-ended lexicon of this game, and also in order not to have to constrain the participants to wear a microphone. Further studies will target an autonomous system, in a different scenario where we will take these lessons into account.

Subjective Measures from Literature

The results collected for the subjective measures taken from literature are presented in Figure 7.18. Within the non-specific measures, some had been considered in order to bring additional support to the specific ***Intention*** measure, while other were considered in order to support the specific ***Animation*** measure.

Supporting the specific measure of *Animation*

Within the *Godspeed* questionnaire we found only the dimension of ***Animacy*** to report significant differences between conditions. We initially expected no differences to be reported for this dimension, just as there were none

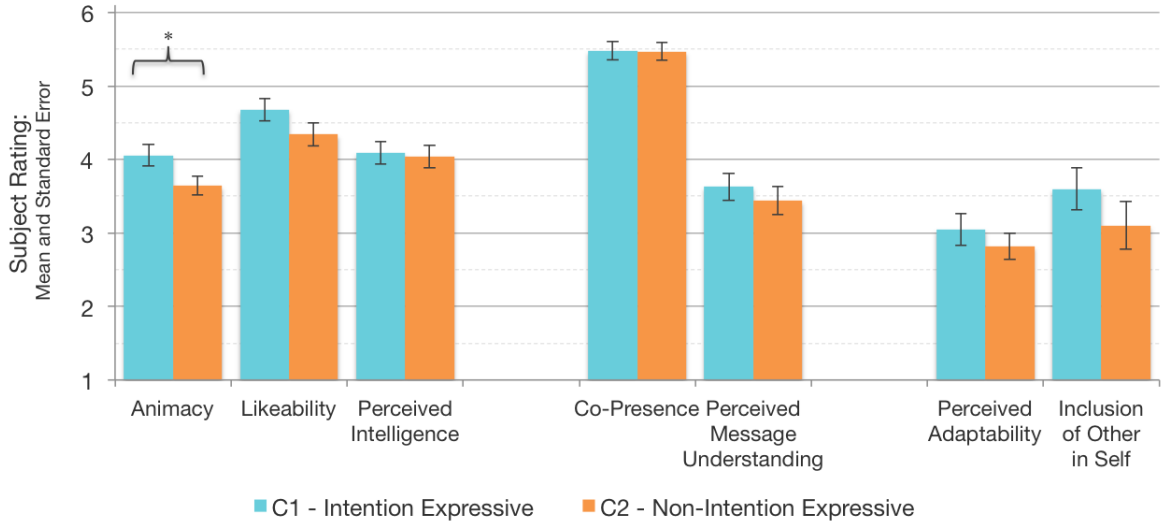


Figure 7.18: The results collected from measures taken from literature. An asterisk (*) reports a $p - value < 0.05$. A double-asterisk (**) reports a $p - value < 0.01$.

for our specific Animation measure. However, we understand that these differences can reasonably be attributed to the fact that in C1 the robot would change postures frequently, thus moving more and this also conveying a higher sense of Animacy ($M_{C1} = 4.058$, $SD_{C1} = 0.660$;$M_{C2} = 3.642$, $SD_{C2} = 0.565$). It is important however, to note that in order to support our hypothesis, we attribute more importance to the specific measures than to the Godspeed measure. That is because *Godspeed* is meant for general assessment of the perception of a robot by users, for which the Animacy dimension seems coherent in our scenario. The robot's behaviour in C1 was in fact more active and animate. The questions for our specific measures were purposefully designed to capture the general perception of the robot as an animated character, based on animation principles, regardless of it being more active in one condition than in the other.

The ***Likeability*** dimension of *Godspeed* could be argued to support the Animation measure. It did not report significant differences between the conditions, and reported good results ($M = 4.510$, $SD = 0.705$).

Within the *Networked Minds* questionnaire we found only ***Co-Presence*** to yield very high results, ($M = 5.471$, $SD = 0.538$) with the scale maximum being 6. This was a positive finding that could actually be brought in to support the Animation measure. Because there was no verbal communication with the robot, we could attribute the whole of this measure's imputation to the design of the robot's behaviour and its execution through animation.

Supporting the specific measure of *Intention*

All the other measures of ***Perceived Intelligence***, ***Perceived Message Understanding***, ***Perceived Adaptability*** and ***IOS*** were considered to maybe support the ***Intention*** measure. We expected to find differences between conditions, as in C1, the robot's ability to non-verbally respond to the player's guesses would result in the perception that the robot was more intelligent, more able to understand and adapt to the player in that condition, and also to create a closer sense of proximity. Although the means for all of them were lower in C2 than in C1, the differences were not statistically significant to increase the support of our hypothesis. This can be because the questionnaires were not specifically designed to support the measure of the robot's intention during a task such as the one of our study.

Objective Measures

The objective measures collected presented (Figure 7.16)no relevant differences among conditions. Although statistically significant differences were reported on some measures in one of the round, we considered it to be an isolated occurrence most likely due to a small sample size. Moreover, Round 3 was noted to be the most complicated round because most people were unaware of what the pantomimed word was at all (it was a Metronome, a device used during musical training). Not only did we see only one participant to be able to guess the word, but during informal post-experimental conversation with some participants, most of them asked us what it was. Therefore we should not draw conclusions regarding any factor reported solely on Round 3. The fact that there were no other differences reported means that while the participants in C1 did perceive that the robot was trying to help them, in the end that help did not translate into a better outcome for the participants' performance in the game. It was unknown for us whether or not to expect differences in this matter. While it would have seemed positive to have differences reported regarding the participants' performance, those differences would possibly bias the perception of the robot's intention by the users. In that case, one could argue that, had the participants in C1 reported a better score, it might have been not because they really understood the robot's intention during the interaction, but instead, felt that the robot was helping, solely because they had achieved a higher success rate.

The fact that our data showed no relevant significant differences actually means that we can fully attribute the results from the questionnaires to the robot's expressivity, thus further supporting our hypothesis.

7.4 AvantSatie - The Piano Game Companion

After analysing the results and conclusions from the Ahoy scenario, we became interested in studying the use of ERIK in a more realistic and fully autonomous setting, by studying not only the robot's ability to convey a recognizable expression, but more importantly, its ability to direct the user's choice of action in a problem-solving task. In the previous evaluation the players were standing still which fails to validate the algorithm's ability to express a given posture towards different directions. Furthermore, the queues were used to try to direct the players towards the correct answer in a pantomimic game, which took an open input (through speech), and required the use of Wizards to listen and decide on the robot's reactive expression. In that scenario, the robot seemed to be more like part of the riddle, than as part of a solution to it. We therefore drew a new goal of understanding if the technique can actually be used in a collaborative scenario in which users solve a problem that is independent of the robot, but in which the robot's expressive behaviour may take a role in assisting the user, by providing expressive queues that facilitate the user's action selection.

The new question we therefore posed to address is:

> Can ERIK be used to provide expressive behaviour to an *autonomous* face-tracking articulated robot, in order to convey a *utilitarian intention* to the user and *direct its action selection* within a task, while also conveying the illusion of life?

For that purpose we established the following requirements for the new evaluation scenario:

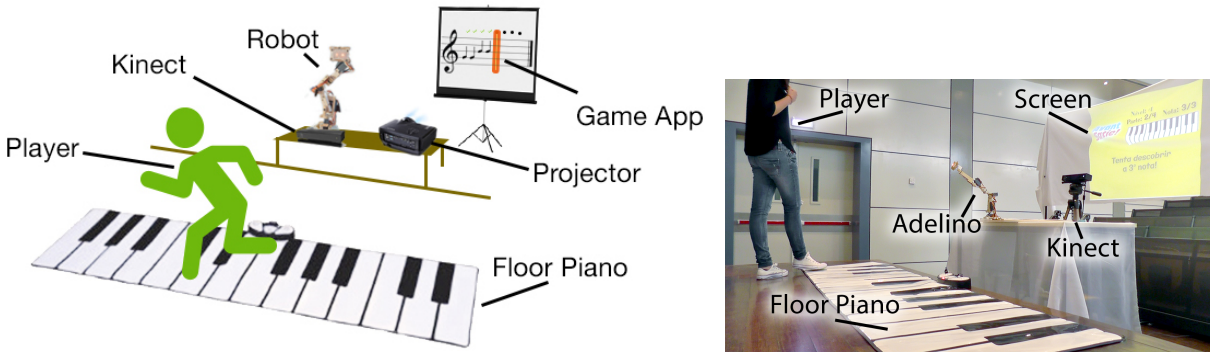
- The robot must be **fully autonomous**, in order to evaluate ERIK based on its actual response-time;

- Users must be forced to **move around** in order to fully evaluate the expressive gaze-tracking behaviour;
- The task must be a non-subjective problem-solving one, and must be **solvable even without expressive queues** (hints) through a *trial-end-error* method, however such hints should allow to solve it with significantly less *errors*.

Because we thought Adelino's design was still appropriate for our evaluation, we decided to use the same robot as in the previous one, given that our major concern was on the actual evaluation's activity design and conclusions, and not on the use of this particular robot. In fact, if the claims do stand, then our opinion is that Adelino may also represent a breakthrough in robot design for HRI and inspire future generations of robots.

7.4.1 Avant Satie - Game description

AvantSatie is a pervasive game where players must discover the musical score of a piece using a floor piano where they can step on to play notes. To help them in the game, they interact with an autonomous Adelino robot that will help them discover which notes compose the musical scores of two different pieces (each piece corresponds to a level). By interacting with the piano, observing the robot and following the instructions, participants either adopt a trial-and-error process, or track the robot's hints, to discover each successive note. The game's set-up is illustrated in Figure 7.19a along with a shot of an actual experimental session (Fig. 7.19).



(a) The concept set-up of the AvantSatie game.

(b) The set-up of the AvantSatie study.

Figure 7.19: The setting of the AvantSatie scenario and study.

The fully autonomous robot enriches the setting of the game by providing the story as well as being socially present by performing compound-gazing. The robot's gazing behaviour combines both a face-tracking feature, and deictic gazing towards specific piano keys. At the same time, the robot is expressive and can shape its posture while gazing, in order to convey hints to the player. Through the understanding of such hints, players can play the game while minimizing the amount of mistakes performed through the trial-and-error gameplay. Yet, if they don't pay attention to the robot, their task in the game becomes much more difficult. The game was designed and iteratively tested with users in pilot studies, in order to ensure that the instructions and gameplay were clear, instead of relying on initial instructions given by the experimenters, which could introduce biases.

As the game is about discovery, the scores and composition are initially unknown to the player. Therefore, they must attempt to play keys on the piano until they find each correct note. Because the robot's behaviour is fully non-verbal, a screen is projected behind it, providing basic instructions and progress (e.g. current level and part).

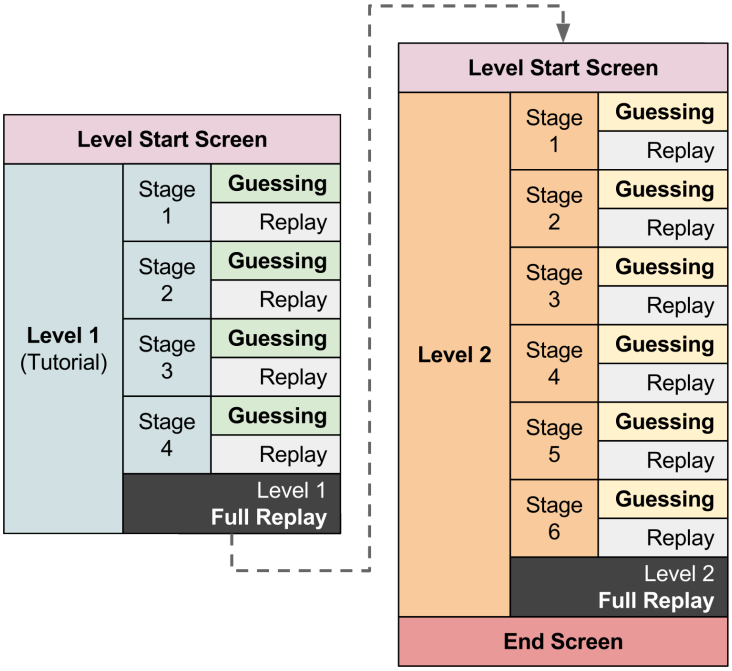


Figure 7.20: A diagram of the game-flow of Avant Satie. There are two levels, composed of a sequence of stages. Each stage is composed of a a challenging activity (Guessing) followed by a rewarding activity (Replay), with each level ending with a greater reward (Full Replay).

The structure of the game-play is outlined in Figure 7.20. There are two levels in the game, which correspond to two musical compositions, each one with distinct levels of difficulty. Each level (music) is split into stages, which we called Parts, as that term fits better in the context of the game (e.g. the music in Level 1 is composed of 4 Parts/Stages). Each Part contains a sequence of one to six individual notes to be discovered one by one in the correct order.

Figure 7.21 shows some shots of the AvantSatie screen throughout the game. Additionally, here we present shots of the English version, while the study was ran using the Portuguese version (as can be seen by comparing with Figure 7.19.b).

The start screen requires the player to interpret a piano figure and to interact with the floor piano (fig. 7.21.a), upon which the robot performs an affirmative animation, i.e., nodding as if trying to say "yes!" (first positive feedback). This ensures that the player understands the basic interaction pattern of the activity (i.e., screen displays instructions, playing the piano triggers a reaction on the robot). It then follows with a little storyline and instructions on how to play (fig. 7.21.b). Because Adelino is designed and animated as a fully non-verbal character, we relied on the screen to present in-game instructions, which also helps to immerse the player into the activity (versus having provided instructions prior to the activity). Whenever an instruction screen is being presented (e.g. fig. 7.21.b) the robot turns to face the screen, as a mechanism to direct the player's attention to it (otherwise due to enthusiasm, the player might just be analysing the robot and overall set-up). This also adds to a feeling of presence - the robot is aware both of the player and of its surroundings (i.e. the screen - a point of shared attention). When the first instruction set is over, the robot turns back towards the user and plays the affirmative animation again. This animation is later used throughout the game, so it was important to initially reinforce it as a positive feedback.

On the first Stage of all, the player is presented with no information except for the instruction "Discover the

1^{st} note!" (fig. 7.21.c). While the player is performing his/her attempts, the robot only performs face-tracking behaviour. Upon playing some note, the robot assesses it as the player's guess. If it is correct, the robot performs the affirmative animation, after which the screen progress updates to e.g. "Discover the 2^{nd} note!", and the robot goes back to face-tracking. This way we manage to keep the player's visual attention focused on the robot instead of the screen, as the robot always provides feedback before the screen does. These steps repeat until all the notes of the current Stage are found. After the player has discovered all the notes of the current Stage, the robot replays all the Stage's notes, while pointing at each corresponding piano key, and then instructs the user to repeat it, with the screen exhibiting an illustration of the piano, highlighting each note, so that the player can unequivocally follow (fig. 7.21.d). This Replay phase serves as a reward to the player for having struggled to discover the composition. Each level was therefore built as a sequence of Challenging-Rewarding phases in order to maintain the user's engagement. At the end of each level the player gets to replay the full Level's composition as a bigger reward (fig. 7.21.e-f). The name of the piece and composer is revealed, and the player replays each part as before, but successively.

7.4.2 Study Conditions

The main purpose of this experiment was to find out if ERIK could be used to introduce extra task-directed information that would be conveyed fully through its posture, while it is gaze-tracking the player. For that, we compared three versions of AvantSatie.

In version **C-ERIK** and **C-EBPS**, Adelino would respond to each of the player's guesses by modifying its posture based on a "hot-cold" heuristic. Upon each wrong guess the robot would therefore shift its expressive posture to either *Hot* or *Cold*, while keeping the gaze-tracking behaviour towards the player.

We did not initially mention that the robot would perform this type of behaviour. Instead, it was expected that during the first minutes of the game (i.e. the Tutorial level), the players would notice that the robot was performing some behaviour that seemed congruent with their guesses, and would learn how to read the robot's posture in order to score better in the game.

The difference between **C-ERIK** and **C-EBPS** is purely technological - in **C-ERIK** we used ERIK to perform the gaze-tracking with expressive posture. For **C-EBPS** we created EBPS, a non-IK example-based posture synthesis technique for which we previously authored a large number of postures for each *Hot*, *Neutral* and *Cold* expressions, each posture representing a pair (*expression*, *direction*), which sets the robot facing through a range of directions that are expected for this game. In run-time, given the face-tracking information, we take the two postures that represent the directions that are closest to the target one and interpolate them. In our pilot studies we verified that this technique provided very smooth and acceptable results, by creating postures for each horizontal direction from -70° to 70° with 10-degrees interval between them, which leads to 15 postures per expression, for each vertical direction. We initially considered to use 3 or 4 vertical directions, but upon testing realized that 2 would be enough, and that the interpolation between the two extreme vertical positions yielded acceptable results. The two vertical directions for which the postures were created were at 0° (looking straight ahead, i.e., to the horizon) and 80° (upwards).

We note that, while the end-result looked similar to the one we wanted to achieve with ERIK, it required a considerable amount of work to create all those postures; furthermore if we wanted to modify one of the expressions, we would have to re-create all the postures for that expression. Using ERIK only requires to author one single

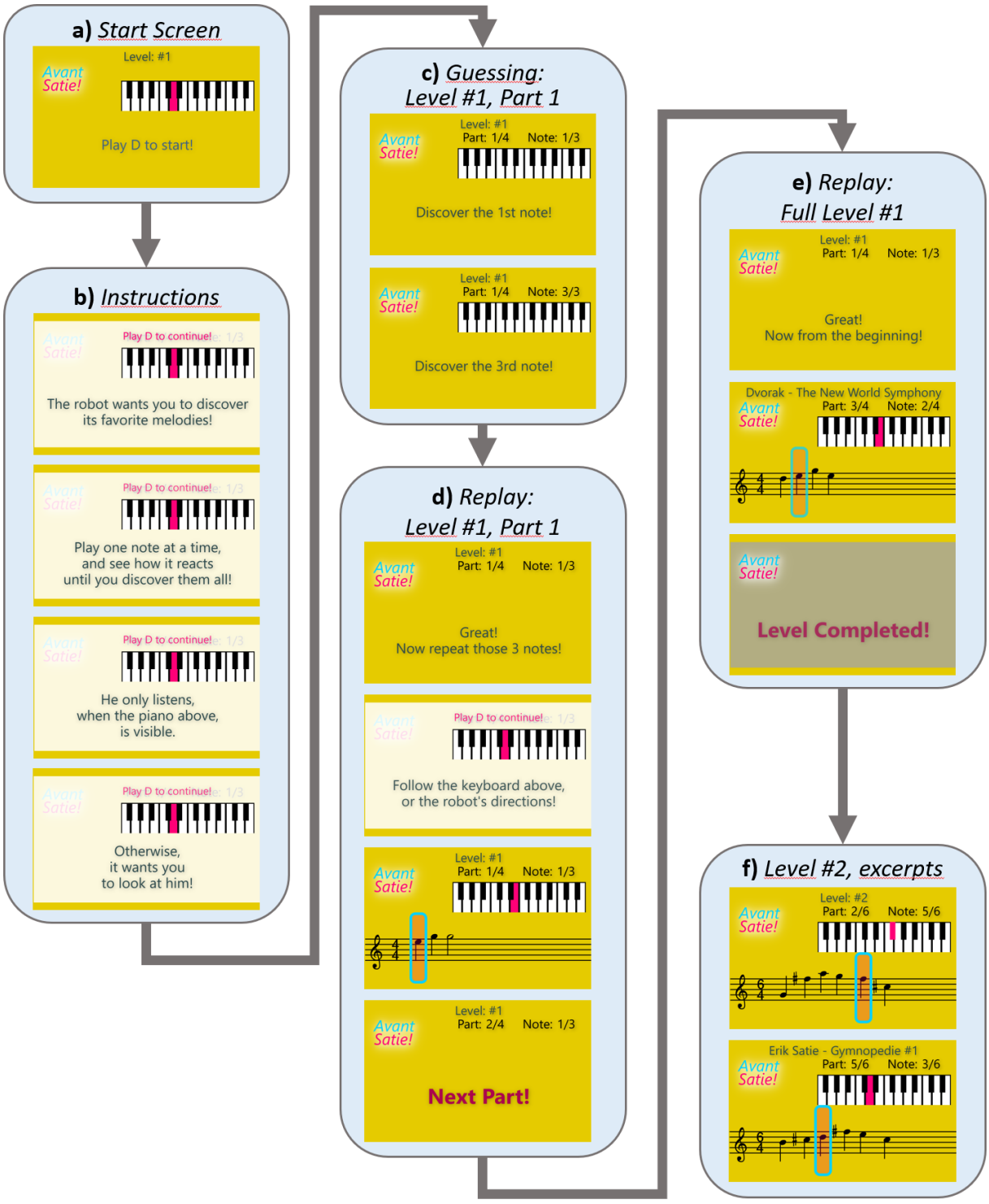


Figure 7.21: Various screenshots of the projected screen of the AvantSatie game: **a)** The first screen that is presented to the player. **b)** After the first screen, the player is instructed on how to play. **c)** While guessing the notes in Level #1. **d)** Replaying a *Part* of level #1, including the instruction which is presented only on the first Replay phase, and the *Part* transition screen. **e)** Replaying and completing the full level #1. **f)** Some excerpts of a *Part* replay and the level replay for level #2.

Figure 7.22: Pictures of the three different postures used by Adelino in the AvantSatie scenario. These postures are the actual output of ERIK as used in the game, while gazing forward, and not a set of static pre-designed postures. The top row shows a frontal view, while below each is its corresponding side view. The postures represented are: **left** - Neutral; **center** - Hot; **right** - Cold. The postures used in C3 are not depicted, but are very similar to these, as they were designed so that both conditions would *seem* alike.

posture per intended expression.

Finally, **C-Control** differs from the other two conditions by not performing any posture-expressive behaviour at all. The whole game is exactly the same, and the robot also performs gaze-tracking behaviour, but upon each guess, it never changes its posture from the initial *Neutral* one. Therefore the players rely solely on trial-and-error to discover all the notes. Technologically though, **C-Control** still uses ERIK to perform gaze-tracking, while always holding the same, *Neutral* posture.

7.4.3 Study Design

The study followed a between subjects design with random assignment within the three different conditions. In all conditions the game design follows the same structure, with the same levels and tutorial information. As detailed in the previous subsection, the differences apply **only** to the Guessing phases. Upon assessment of a player's guess, the result can be **Correct** if the player found the correct note at this point of the game; **WrongHot** if the player guessed wrong, but is moving towards the Correct one; or **WrongCold** if the player guessed wrong, but is moving away from the Correct one.

Whenever the player's guess is not **Correct**:

- **C-ERIK**: the robot uses ERIK to portray a *Hot* or *Cold* expression to the player;
- **C-Control**: the robot uses ERIK to hold the *Neutral* pose (does not portray a *Hot* or *Cold* expression;
- **C-EBPS**: the robot uses the EBPS method to portray a *Hot* or *Cold* expression to the player;

In any case the robot always maintains the face-tracking behaviour throughout the Guessing phases. Therefore tracking is blended with postural expression using either the ERIK algorithm (**C-ERIK**, **C-Control**), or the EBPS method (**C-EBPS**).

In order to address our research question, we established the following hypotheses:

- **H1**: Players in **C-Control** will play worse than in **C-ERIK** and in **C-EBPS**.
- **H2**: Players in **C-ERIK** will play at least as well as in **C-EBPS**.
- **H3**: The robot is perceived to convey the illusion of life in all conditions.
- **H4**: The robot is perceived to convey the illusion of life more in **C-ERIK** and **C-EBPS** than in **C-Control**.
- **H5**: The players are able to perceive the robot's intention and motivation as being towards helping the player in both **C-ERIK** and **C-EBPS** but not in **C-Control**.
- **H6**: The game is understandable and the robot is perceived to play well in all conditions.

The first hypothesis H1 is expected to be confirmed given that in **C-ERIK** and **C-EBPS**, players should be able to self-assess their own guesses, by observing and interpreting the robot's expressive behaviour. The same argument also applies to H5, albeit they are based on different measures (H1 is objective, H5 is subjective). For **C-Control**, we expected that players would either play fully through trial-and-error, look for but fail to interpret any information from the robot's expression (which is correct), or that they would wrongly interpret the robot's face-tracking behaviour as being related with their guesses. In any case, the players in **C-Control** are expected

to perform randomly, given that there was no available mechanism for self-assessment of their performance throughout the game. The hypothesis H2 is also expected to be confirmed given that although **C-EBPS** provides more consistent pre-designed poses, **C-ERIK** should still be able to convey the same contextual information using the ERIKalgorithm.

We also expect to confirm H3 and H4 given that we have tailored this experiment to detect only **differences regarding the perception of the robot's hint-providing intention along with its motivation** of wanting the player to succeed, in **C-ERIK** and **C-EBPS**. However, by having endowed the robot with the expression of thought and motivation [31] in **C-ERIK** and **C-EBPS**, we expect that to contribute to higher scores in those conditions as compared to **C-Control**.

Finally, it is important to verify that neither the inclusion nor the lack of the hint-providing behaviour hinders the player's ability to understand and play the game (H6), which could potentially lead participants to rate the whole scenario (including the robot) with lower scores.

7.4.4 Sample

For this study we recruited a total of 59 university students (30 females and 29 males) with ages ranging from 18 to 35 (M = 22.78; SD = 3.96). From these, two were excluded due to not complying with the instructions, thus yielding a total of 57 valid participants, which resulted in a balanced distribution of 19 participants per condition. 19% of the participants had already interacted with a robot before once, and 37% more than once. Regarding their knowledge of music, 42% reported a low to no level of expertise playing some musical instrument, while 39% reported an intermediate expertise, and 19% an advanced expertise. As for experience reading sheet music, 63% reported a low score, 25% an intermediate score, and 12% declared to be experts. It became clear that our sample contained a high level of musically-instructed participants, which we attribute to the dissemination of the study in which participants were requested to play a game using a floor piano, through a flyer shared via social-media networks.

7.4.5 Procedure

Upon arrival, the participants filled out the consent form in a separate room before being led to the game room. There they were given the same initial instructions, without revealing that the robot would indicate the result of their guesses through a change in posture. Instead they were solely informed that there would be two simple compositions to discover, that they should perform each guess and observe the robot, until they were able to discover all the notes, and otherwise just follow the instructions on screen. While this gave them a tip that the robot might perhaps help them, we ensured that each player would perform their own interpretation about the way the robot moved, by understanding the relationship between the robot's expression and their guesses. This also ensured that all participants had been hinted to look at and try to interpret the robot, in all conditions. The researcher would direct them to enter the room and start interacting without following them, as the robot was already active and would start face-tracking them once they stepped into the Kinect's field of view. This ensured that the participants noticed it immediately as an autonomous entity and would become immersed into the game. The screen provided the starting instruction, which was to play a D note on the floor piano, along with an icon showing explicitly which key that was. Therefore it was the participant who took the step to initiate the game, while also ensuring that they understood the

piano to be a controller for it. When finished, the participants were taken back to the initial room, administered a set of questionnaires, and received a thank-you gift (a movie ticket) at the end.

7.4.6 Measures

To represent our sample, demographic information was requested in the questionnaires (gender, age, previous interaction with robots and level of expertise in both musical instrument playing and score reading capabilities). In addition, all participants, responded to the following questionnaires:

- *Networked Minds* [180] scale (**NM**), from which we took the *Perceived Message Understanding* (**PMU**) and the *Co-Presence* (**CP**) dimensions to measure the degree to which the participant believes s/he is not alone, i.e., that the robot is present as a social entity (CP dimension), and that its communicative behaviour was clear and understandable (PMU dimension);

- *Inclusion of Other in Self* [181] scale was used to measure the closeness that the participants felt between them and the robot (measure **IOS**);

- **RoSAS** [184] scale was used to measure the participant's perception of the robot's social attributes regarding Competence, Warmth and Discomfort (dimensions **RC**, **RW** and **RD**);

- *Perceived Adaptability* (**PA**) from the *Almere model* [182] which measures the perceived ability of the system to adapt to the needs of the user;

- *Robot's Performance & Usability* (**RPU**) scale was used to measure how well the participants felt the robot to be able to play the game, and how well the overall gameplay was designed;

- *Robot's Intention & Motivation* (**RIM**) scale was used to measure how much participants felt the robot was there to provide tips and how much it wanted to succeed in helping them;

- *Animation Illusion of Life* (**AIL**) scale was used to measure the illusion of life of the robot.

The questionnaires for the RPU, AIL and RIM scales were specifically designed to address our research question, and carefully written in order not to bias the participants' assessment of the robot's animation qualities throughout the variations of the robot's behaviour that are contained in the study. The RPU scale in particular is composed of two dimensions:

- *Task Performance* (**TP**) measures how well the participants perceived the robot to know the game and perform the task well;

- *Task Usability* (**TU**) measures how easy and intuitive the participants felt it was to understand the task and the game-play interaction with the robot and the screen.

The RIM scale is also composed of two dimensions:

- *Robot's Intention* (**RI**) measures how much the participants felt that the robot was providing hints to them throughout the task;

- *Robot's Motivation* (**RM**) measures how much the participants felt that the robot's intrinsic motivation (i.e. *purpose*) was to help them (by providing hints).

Table 7.3 lists the questions used for each of these measures.

All the questionnaires were answered in a 6-point Likert scale except for the RoSAS, which was answered in a 9-point Likert scale, and the IOS measure which was answered in a 7-point pictorial scale. The RD scale is negatively-scored, i.e., lower scores in the questionnaire mean better scores. Therefore in our analysis and results presentation, we have corrected the data (by inverting the scale) in order to present and compare its data as it were also a positively-scored scale. All items were shuffled to mask for their dimensions.

	RPU (TP + TU) - Robot's Performance & Task Usability
TP1.	The robot knew where each note of each music was.
TP2.	The robot always understood what note I had played.
TP3.	The robot knew each music very well.
TP4.	The robot knew every music by heart.
TU1.	I had to look at the screen to know what happened at each moment.
TU2.	I wouldn't understand the game without looking at the screen.
TU3.	The game screen had all the info I needed to understand the game.
TU4.	I had to follow the screen to know what to do.
	RIM (RI + RM) - Robot's Intention & Motivation
RI1.	I wouldn't have understood the game without the robot.
RI2.	I managed to find the correct notes thanks to the robot.
RI3.	I wouldn't have discovered the musics without the robot's help.
RI4.	The tips that the robot gave me helped me to find the correct notes.
RI5.	The robot's tips were consistent with my attempts to find each correct note.
RI6.	The robot gave me tips while I was trying to find each correct note.
RI7.	I was able to understand if I was close or far from the correct note thanks to the robot's tips.
RM1.	The robot wanted me to find the correct notes.
RM2.	The robot wanted me to discover all of the musics.
RM3.	The robot was enthusiastic with my attempts to find the correct notes.
RM4.	The robot thought about helping me.
	AIL - Animation Illusion of Life
AIL1.	The robot's movement was smooth and natural.
AIL2.	The robot seemed to be alive.
AIL3.	The robot reminded me of characters I know from movies.
AIL4.	The robot's motion followed my rhythm.

Table 7.3: Questionnaires used for the RPU RIM and AIL measures in AvantSatie.

The following objective data was also collected during each session, and measured only during the parts of the game in which the players were performing guesses:

- **Time** spent guessing;
- **WrongHot** number of incorrect guesses which were however assessed as Hot;

- **WrongCold** number of incorrect guesses which were however assessed as Cold;
- **WrongTotal** aggregates WrongHot and WrongCold.

7.4.7 Results

Throughout the presentation and analysis of the results, we will be considering each measure's average across each of the different three conditions, plus an extra **C-Enhanced** group which averages the measures of both **C-ERIK** and **C-EBPS** in order to treat them both as a single group.

We started by verifying the internal consistency of the scales by measuring the Cronbach's *alpha* (α) and the MacDonald's *omega* (ω) for each measure, as shown in Table 7.4, and considered an optimistic selection of the result, i.e., we considered the scale reliable as long as either *alpha* or *omega* are above the threshold value of 0.7.

AIL	0.708	0.721	0.577*	0.630*	0.577*	0.602*	0.770	0.779	0.559*	0.606*
RIM	0.899	0.935	0.887	0.935	0.890	0.944	0.863	0.925	0.857	0.911
RPU	0.731	0.850	0.755	0.902	0.744	0.907	0.728	0.853	0.728	0.861
NM	0.870	0.924	0.847	0.930	0.891	0.956	0.835	0.914	0.866	0.923
PA	0.732	0.785	0.804	0.884	0.569*	0.656*	0.773	0.795	0.698*	0.773
RW	0.727	0.845	0.626*	-	0.731	0.858	0.553*	0.768	0.750	0.873
RC	0.900	0.938	0.869	0.949	0.860	0.957	0.922	0.973	0.864	0.928
RD	0.648*	0.826	0.568*	0.840	0.405*	0.495*	0.748	0.939	0.441*	0.604*
ri	0.924	0.959	0.929	0.971	0.887	0.960	0.889	0.943	0.908	0.961
rm	0.698*	0.759	0.317*	0.548*	0.570*	0.716	0.752	0.857	0.414*	0.599*
tp	0.749	0.798	0.807	0.879	0.846	0.754	0.540*	0.667*	0.828	0.866
tu	0.694*	0.761	0.775	0.912	0.534*	0.759	0.731	0.825	0.678*	0.172*
cp	0.745	0.911	0.728	0.939	0.881	0.969	0.636*	0.828	0.791	0.923
pmu	0.872	0.930	0.788	0.898	0.935	0.979	0.836	0.903	0.867	0.935

Table 7.4: Reliability analysis of the various scales and dimensions, using Cronbach's alpha (α) and McDonald's omega (ω). Green represents a *reliable* scale, while yellow with an asterisk* represents otherwise.

The Shapiro-Wilk test of normality was then used to verify for which measures the data was normally distributed (ρ>0.05), as shown in Table 7.5.

Although we had three conditions and present the data in four groups, in our analysis we were interested in comparing the means of only two groups at a time, thus we used the Student's t-Test when the data is normally distributed in both groups being tested, and the Mann-Whitney U test otherwise.

Table 7.6 shows the results of comparison of the means of the various subjective measures scales' and sub-dimensions between the different groups. These results become even better illustrated in Figure 7.23a, which shows only the scales, and in Figure 7.23b which show only the dimensions that compose some of the scales. This level of detail is intended to provide further insight onto the analysis and conclusion about what has caused any observed differences (or not). Through the reading of this presentation of results, those figures may be used as a handy assistant.

Finally, Figure 7.23c shows how each measure in each group compares to the scale's median value, i.e., is the average score significantly *positive*, *negative*, or *neutral*. In order to assess that, we took each scale's median value (e.g., 'AIL' is answered in a 6-point likert scale, from 1 to 6, thus the median is 3.5), and compared the distribution of each measure within each group, to the measure's scale median. Whenever the data within the group followed a normal distribution we used the One-Sample t-Test, and otherwise we used the One-Sample Wilcoxon Signed Rank

	C-ERIK	C-Control	C-EBPS	C-Enhanced
AIL	$\rho = 0.219$	$\rho = 0.080$	$\rho = 0.550$	$\rho = 0.056$
RIM	$\rho = 0.006*$	$\rho = 0.943$	$\rho = 0.037*$	$\rho = 0.001*$
RPU	$\rho = 0.050$	$\rho = 0.767$	$\rho = 0.030*$	$\rho = 0.004*$
NM	$\rho = 0.027*$	$\rho = 0.271$	$\rho = 0.013*$	$\rho = 0.002*$
PA	$\rho = 0.030*$	$\rho = 0.557$	$\rho = 0.557$	$\rho = 0.029*$
IOS	$\rho = 0.279$	$\rho = 0.202$	$\rho = 0.023*$	$\rho = 0.007*$
RW	$\rho = 0.576$	$\rho = 0.989$	$\rho = 0.477$	$\rho = 0.848$
RC	$\rho = 0.176$	$\rho = 0.478$	$\rho = 0.005*$	$\rho = 0.000*$
RD	$\rho = 0.013*$	$\rho = 0.001*$	$\rho = 0.059$	$\rho = 0.001*$
ri	$\rho = 0.027*$	$\rho = 0.120$	$\rho = 0.163$	$\rho = 0.003*$
rm	$\rho = 0.550$	$\rho = 0.640$	$\rho = 0.251$	$\rho = 0.066$
tp	$\rho = 0.001*$	$\rho = 0.399$	$\rho = 0.000*$	$\rho = 0.000*$
tu	$\rho = 0.499$	$\rho = 0.326$	$\rho = 0.771$	$\rho = 0.472$
cp	$\rho = 0.000*$	$\rho = 0.004*$	$\rho = 0.000*$	$\rho = 0.000*$
pmu	$\rho = 0.121$	$\rho = 0.739$	$\rho = 0.006*$	$\rho = 0.004*$

Table 7.5: Results of the Shapiro-Wilk's test of normality on each of the subjective measures. Green marks data which are normally distributed; Yellow with an asterisk* marks otherwise.

	ERIK-Control	EBPS-Control	Enhanced-Control	ERIK-EBPS
AIL	$(t = 2.23, \rho = 0.033*)$	$(t = 2.13, \rho = 0.042*)$	$(t = 2.36, \rho = 0.026*)$	$(t = 0.25, \rho = 0.808)$
RIM	$(U = 310, \rho = 0.000*)$	$(U = 301, \rho = 0.000*)$	$(U = 610, \rho = 0.000*)$	$(U = 166, \rho = 0.683)$
RPU	$(t = 1.43, \rho = 0.161)$	$(U = 273, \rho = 0.007*)$	$(U = 507, \rho = 0.014*)$	$(U = 135, \rho = 0.187)$
NM	$(U = 248, \rho = 0.048*)$	$(U = 286, \rho = 0.002*)$	$(U = 534, \rho = 0.003*)$	$(U = 136, \rho = 0.193)$
PA	$(U = 250, \rho = 0.042*)$	$(t = 2.14, \rho = 0.039*)$	$(U = 503, \rho = 0.016*)$	$(U = 208, \rho = 0.436)$
IOS	$(t = 1.80, \rho = 0.081)$	$(U = 282, \rho = 0.003*)$	$(U = 516, \rho = 0.008*)$	$(U = 122, \rho = 0.084)$
RW	$(t = 0.30, \rho = 0.766)$	$(t = -0.07, \rho = 0.942)$	$(t = 0.13, \rho = 0.899)$	$(t = 0.33, \rho = 0.743)$
RC	$(t = 2.11, \rho = 0.043*)$	$(U = 208, \rho = 0.429)$	$(U = 448, \rho = 0.139)$	$(U = 212, \rho = 0.363)$
RD	$(U = 204, \rho = 0.508)$	$(U = 169, \rho = 0.747)$	$(U = 372, \rho = 0.851)$	$(U = 214, \rho = 0.331)$
ri	$(U = 286, \rho = 0.002*)$	$(t = 3.92, \rho = 0.000*)$	$(U = 582, \rho = 0.000*)$	$(U = 188, \rho = 0.849)$
rm	$(t = 3.40, \rho = 0.002*)$	$(t = 3.04, \rho = 0.005*)$	$(t = 3.44, \rho = 0.002*)$	$(t = 0.23, \rho = 0.823)$
tp	$(U = 250, \rho = 0.041*)$	$(U = 266, \rho = 0.012*)$	$(U = 517, \rho = 0.008*)$	$(U = 164, \rho = 0.646)$
tu	$(t = 0.62, \rho = 0.537)$	$(t = 2.37, \rho = 0.024*)$	$(t = 1.61, \rho = 0.115)$	$(t = -1.39, \rho = 0.175)$
cp	$(U = 246, \rho = 0.046*)$	$(U = 260, \rho = 0.017*)$	$(U = 506, \rho = 0.009*)$	$(U = 180, \rho = 1.000)$
pmu	$(t = 1.82, \rho = 0.077)$	$(U = 286, \rho = 0.002*)$	$(U = 529, \rho = 0.005*)$	$(U = 127, \rho = 0.121)$

Table 7.6: Results of the comparison of means tests on each scale of the subjective measures. Green with an asterisk* marks comparisons which are significantly different.

test. Therefore the interpretation of the figure is as following: if the average of a measure's score within a given group is significantly above the scale's median, then the score is considered to be *positive*. If it is significantly below the median, then it is considered to be *negative*. Otherwise, no conclusion can be drawn on the score's polarization and therefore it is considered to be *neutral*.

While analysing the results in the next subsections we will be especially interested in identifying common types of differences regarding the comparison between groups. In order to make the reading and interpretation easier to follow, we have gathered the following cases:

Strong Expressivity Difference (*Strong E-D*) : Significant difference between the **C-Control** and all **C-ERIK**, **C-EBPS** and **C-Enhanced** groups, with no difference between the **C-ERIK** and **C-EBPS** conditions. These findings will be attributed to the inclusion of the postural/intention-expressive behaviour in the activity, regardless of its technical implementation.

Soft Expressivity Difference (*Soft E-D*) : Significant difference between the **C-Control** and the **C-Enhanced** groups, and also between **C-Control** and just one of either **C-ERIK** or the **C-EBPS** condition. These findings will also be attributed to the inclusion of the postural/intention-expressive behaviour in the activity, but with an indication that one of the technological implementations may have performed better in some aspect.

Technological Difference (*T-D*) : Significant difference between the **C-Control** and either the **C-ERIK** or the **C-EBPS**, but not between the **C-Control** and the aggregated **C-Enhanced** group. These findings will be attributed to some difference in the technological implementation only.

7.4.8 Regarding the Subjective Measures

Within the various subjective measures used in our study, we draw the following statements (follow using Figure 7.23):

AIL, RIM, NM and PA all reported significantly lower scores in **C-Control** with a ***Strong E-D***.

RPU reported a significantly lower score in **C-Control** with a ***Soft E-D***.

IOS, RC reported a significantly lower score in **C-Control** compared to **C-EBPS** with a ***T-D***.

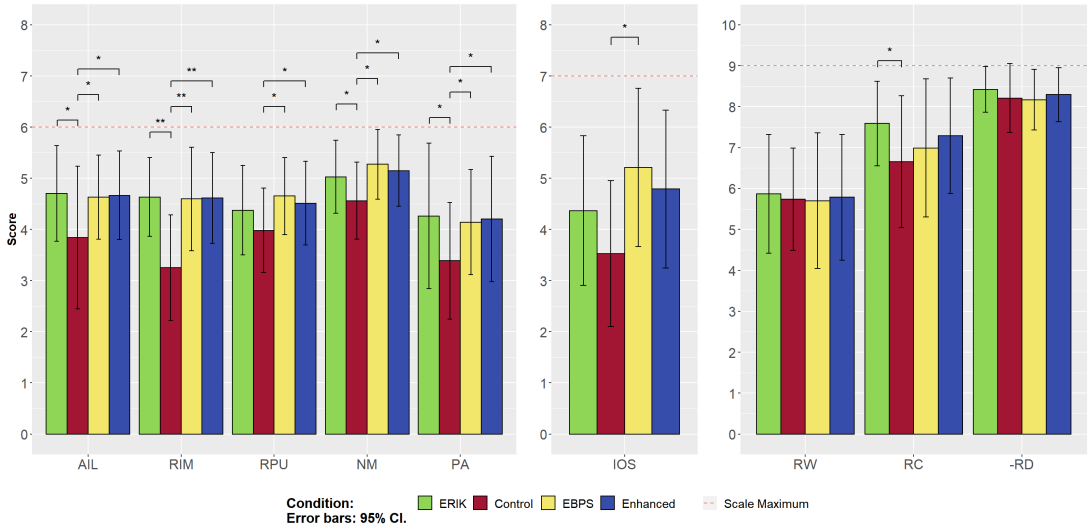
RPU, NM, RC and RD all reported positive scores in all groups.

AIL and RIM reported positive scores in all enhanced groups, and neutral in the **C-Control** group.

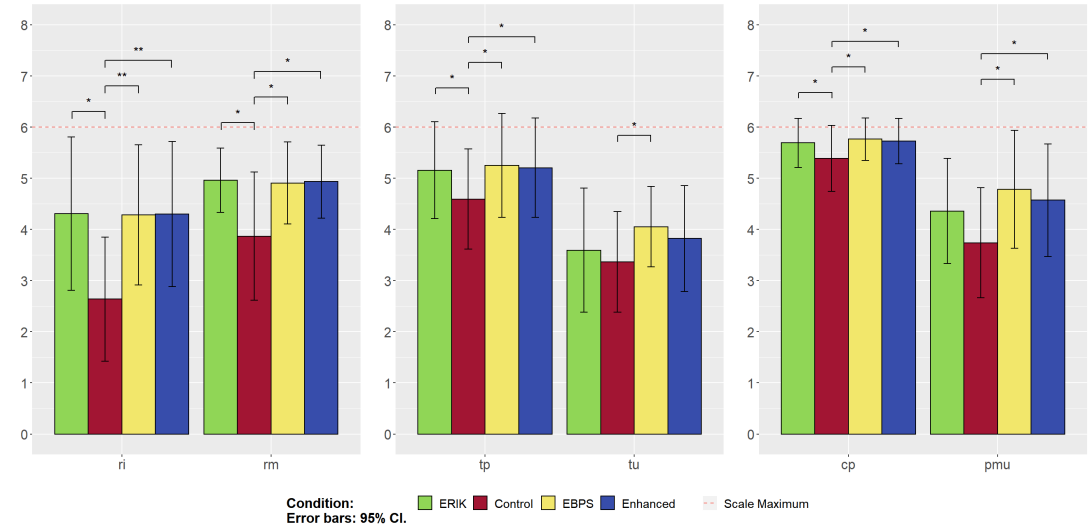
PA, IOS and RW are inconclusive regarding the polarity of the scores, given that they show mixed results.

Within the dimensions that compose these scales (Figure 7.23b), we can verify as expected based on the **RIM** and **NM** results, that the **ri**, **rm**, **cp** and **pmu** dimensions all report the same ***Strong E-D***. However the RPU scale had reported only a ***Soft E-D***. Analysing further, we can see that the **tp** dimension (robot's Task Performance) did exhibit the expected ***Strong E-D***, while the **tu** dimension (Task Usability) reported only a significant ***T-D*** between the **C-Control** and the **C-EBPS** conditions.

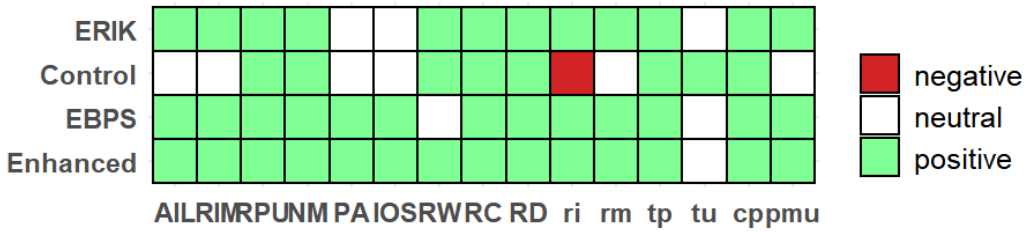
Regarding the polarities of the scales and its sub-dimensions, we find that the the Illusion of Life scale (**AIL**) and the **RIM** scale both reported a positive score in all groups except *C-Control*, where it scored neutral. These two scales are at the core of our research question. We further have a very interesting finding within the **RIM** scale, that regarding the Robot's Intention (**ri**), the **C-Control** scored significantly negative while all the other score positively (and not neutral). This is a very strong difference (which had already been pointed out in the comparison of means). Similarly, albeit with a smaller difference, the Robot's motivation (**rm**) and the Networked Minds' Perceived Message Understanding (**pmu**) were also perceived to be neutral in **C-Control**, while in all the others it was positive. Finally it is important to note that while the **RPU** scale was positive across all groups, we found that this was mostly due to the perceived robot's Task Performance (**tp**), as the Task Usability (**tu**) dimension scored significantly neutral on all groups, except on **C-EBPS** where it scored positively.



(a) Comparison of the scores of the subjective measures' scales.



(b) Comparison of the scores of the dimensions used to compose some of the subjective measures' scales. **ri** and **im** compose the **RIM** scale. **tp** and **tu** compose the **RPU** scale. **cp** and **pmu** compose the **NM** scale.



(c) Comparison of means to the scale's median value on each of the subjective measures for each group. Green illustrates an average positive score, white an average neutral score, and red an average negative score.

Figure 7.23: Comparison of the subjective measures' scales and sub-dimensions.

7.4.9 Regarding the Objective Measures

Figure 7.24 shows how the four objective measures performed across the different groups, with Table 7.7 containing details on the statistical tests performed. The measure of **WrongCold** did not reveal any differences between conditions. However both the measures of **WrongHot** and **WrongTotal** show a ***Strong E-D*** with the **C-Control** condition containing a significantly higher amount of wrong answers than the other groups. The Time measure shows a ***Soft E-D*** as the **C-Control** group performed significantly faster than the **C-EBPS** group (t=2.406, ρ=0.022) and the aggregated **C-Enhanced** group (U=242.0, ρ=0.044).

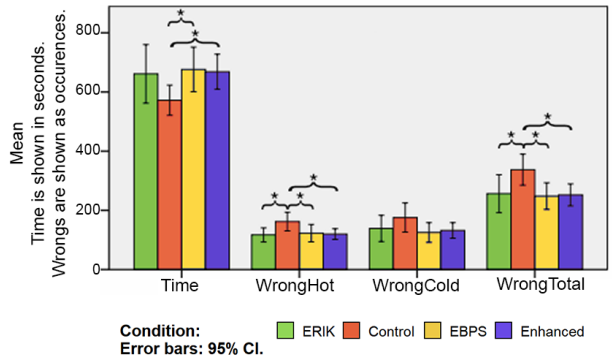


Figure 7.24: Comparison of the results of the objective measures.

	ERIK-Control	EBPS-Control	Enhanced-Control	ERIK-EBPS
WrongTotal	(U=90.0, ρ=0.008∗)	(U=93.5, ρ=0.011∗)	(U=183.5, ρ=0.003∗)	(U=170.5, ρ=0.770)
WrongCold	(U=128.5, ρ=0.129)	(U=118.0, ρ=0.068)	(U=246.5, ρ=0.053)	(U=176.5, ρ=0.907)
WrongHot	(U=84.5, ρ=0.005∗)	(U=94.0, ρ=0.012∗)	(U=178.5, ρ=0.002∗)	(U=173.5, ρ=0.840)
Time	(t=1.676, ρ=0.105)	(t=2.406, ρ=0.022∗)	(U=242.0, ρ=0.044∗)	(t=0.257, ρ=0.798)

Table 7.7: Results of the comparison of means tests on each of the objective measures. Green with an asterisk* marks comparisons which are significantly different.

7.4.10 Discussion

The results collected and analysed show us in general that our research question is supported. Looking into each of our initial hypotheses:

- **H1** - Players in **C-Control** will play worse than in **C-ERIK** and in **C-EBPS**: **True**. The objective results show that participants in both **C-ERIK** and **C-EBPS** committed less mistakes than in the **C-Control**, given that they were able to decode and exploit the hints given by the robot through its full-body posture, while simultaneously performing face-tracking.

- **H2** - Players in **C-ERIK** will play at least as well as in **C-EBPS**: **True**. The objective results show that there was no significant difference between the total amount of wrong answers given by the players in **C-ERIK** and in the **C-EBPS**. This means that the expressive postures provided by the ERIK algorithm were as useful, relevant and legible as ones that were manually designed with a much higher work effort for the EBPS technique.

- **H3** - The robot is perceived to convey the illusion of life in all conditions: **Partially True**. The subjective results show that all conditions except for **C-Control** reported a positive AIL average score. The control group reported a neutral score, which becomes inconclusive as to whether or not we may consider it to have conveyed the illusion of life or not. However we can understand that because in the enhanced groups, the robot was performing an additional intentional-directed expressive behaviour, that contributed to convey they illusion of thought (which is core to the illusion of life).

- **H4** - The robot is perceived to convey the illusion of life more in **C-ERIK** and **C-EBPS** than in **C-Control**: **True**. The subjective results show that the both **C-ERIK** and **C-EBPS** scored significantly higher in the AIL measure than the **C-Control** group. This was already expected, for the same reason as discussed in the previous hypothesis H3.

- **H5** - The players are able to perceive the robot's intention and motivation as being towards helping them in both **C-ERIK** and **C-EBPS** but not in **C-Control**: **True**. The subjective results show that the RIM was significantly higher and positive both **C-ERIK** and **C-EBPS** than in **C-Control**. When analysing RIM's sub-dimensions, i.e., the Robot's Intention (**ri**) and Robot's Motivation (*rm*) separately, we find that the Robot's Intention, not only follows the same tendency, but also scored significantly positive in the enhanced conditions, while scoring negatively in the C-Control condition. This was actually the only dimension that scored negatively. Regarding the Robot's Motivation, we find a similar pattern, except that in **C-Control** it scored neutral (slightly better). Our guess is that in general the players had a positive feeling about the robot (based on the results from RoSAS) and therefore, maybe considered that the robot did intrinsically want to help them (although they reported negatively on its hint-providing intention). It is interesting to note that the same tendency is found on the enhanced conditions, i.e., the **rm** scores are also higher in those conditions than the **ri** ones, thus supporting that in general, and regardless of the actual perceived intention of the robot, it was felt as being there to (supposedly) help them. Furthermore, the Networked Minds (**NM**) and Almere's Perceived Adaptability (**PA**) scales both follow the same patter, all scoring significantly higher scores in the enhanced conditions compared to **C-Control**, thus reinforcing that regardless of the technique used, the intention-directed behaviour of the robot, as designed and integrated into the gameplay, had a positive effect on various measures regarding the perception of the robot's intention, motivation and closeness towards the player.

- **H6** - The game is understandable and the robot is perceived to play well in all condition: **Partially True**. The **RPU** measure shows a positive score in all groups. However when breaking down the scale, we find that the robot's Task Performance (**tp**) was perceive to be positive in all groups, while the Task's Usability (**tu**) was scored as neutral in all except the **C-EBPS** group, and that this difference is actually significant compared to **C-Control**. We initially wanted to ensure that the participants would not become too affected by the lack of the posture-expressive behaviour (in **C-Control**) that they would not understand the task at all. While the **tu** measure reports a **T-D** on **C-EBPS**, it was not reported in the whole **C-Enhanced**, which means that we fail to refute that the inclusion or absence of the robot's intention-directed expressive behaviour does not cause a significant effect on the participants' understanding of the game and the task. Therefore the iterative game-design method (with 3 pilot tests) allowed to tweak the usability of the game to an acceptable level,

even in the absence of the robot's full expressive behaviour, while also noting that the overall game-design and/or interaction design could have still been made better.

We further develop on additional findings that were not initially expected or foreseen, or directly relevant to our research question. We noted however that participants in **C-Enhanced** took more time to complete the task than in **C-Control**. Inspection of the audio-video data captured from the study revealed that participants in the former conditions, having understood that the robot was giving tips, would try notes at a lower pace in order to inspect the robot's response. In **C-Control**, after a while they would quickly play random notes, which, although counting up to a significantly higher number of mistakes, drove them quicker through the task. This was an interesting finding for which we initially had no expectations, but further supports the design quality of the activity and of the distinction between **C-Enhanced** and **C-Control**.

The RoSAS scale shows only a significant difference in the Robot Competence (**RC**) measure between **C-ERIK** and **C-Control**. However the difference did not hold for the whole of the **C-Enhanced** group. We suspect that the ERIK algorithm may have yielded a higher feeling of competence, because the use of that algorithm is prone to result in more dynamic/responsive motion, which players may have attributed to a higher sense of acknowledgement of the other, and capability of attention, on the robot's part. In overall however, no concrete difference may be concluded between **C-ERIK** and **C-EBPS**, given that on comparing the various scales, there were either none, or mixed differences (e.g. in contrast to the previous remarks, for the IOS scale, the **C-EBPS** scores significantly higher than **C-Control**, but here **C-ERIK** does not). Although the RoSAS scale seems not to have added any relevant information, that fact may be used to also hypothesize that across all groups, the robot was perceived as being nearly the same *character* - which was desirable for our study.

To conclude, and recalling our research question:

> Can ERIK be used to provide expressive behaviour to an *autonomous* face-tracking articulated robot, in order to convey a *utilitarian intention* to the user and *direct its action selection* within a task, while also conveying the illusion of life?

We have found evidence that this question is positively supported, given that:

- When ERIK was used, the participants noted its intention-directed postural behaviour, and were able to intuitively understand it without having been given any information about its existence in order to perform better on a task that required it.

- The effect of using ERIK was similar to that of a manually tailored (and laborious) alternative technique EBPS, in that no significant differences were found for any of the measures between those two groups, while significant ones were found especially on key measures when compared to the Control condition. This means that ERIK can be used in substitution of such currently existing manually-tailored and arduously worked techniques (such as ones based on *learning-by-examples*).

- The difference between the various conditions did not hinder the player's understanding and playability of the game, impacting only on their performance, which reveals that the selected task was properly designed to answer our question.

- Despite the shakiness of the robot due to the fact of it being a low-fidelity craft robot, results show that the algorithm succeeded in making it convey both the intended expressivity, and the illusion of life, meaning that it is likely to work on both similar or more solid robots.

In addition, we highlighted the importance of the *illusion of thought* to the overall *illusion of life in robots*, as already had been initially proposed by Takayama et al. [31]. In our case we further demonstrated that such illusion can also be expressed through fully interactive expressive postures that are computed in real-time, and are therefore most appropriate for use-cases involving autonomous social robots.

Chapter 8

Conclusion

No matter how broad a thesis may be, and how diversely it may approach a topic, it still seems like it's never enough to behold the full breath of outcomes and contributions that we have aspired to. Nevertheless, by the time we complete this chapter, we finally see ourselves in a position of fully wrapping up this stage of our research. Controversially, the conclusive chapter of a thesis does not really stand up to its name, as it generally performs one glimpse into the past, and another glimpse into the future.

Through these years we have explored and achieved ground-breaking developments on the field of robot animation, in particular on how it can be used in HRI applications featuring autonomous social robots.

We have established our grounding on the field of character animation, where artists have dwelled within the last 100 years. There we we also found land upon which to build the foundations on robot animation principles, and from where we further built a bridge that connects the animation field with robotics, spanning over and across the broader field of CGI animation.

A major part of our contribution is of technical nature. That is because CGI animation and robotics typically follow different procedures, workflows and paradigms, and therefore we were required to establish new techniques and tools that could be supported on the principles and requirements of both fields. These tools and techniques therefore pose today as examples from which to build other tools in the future, to address particular problems either in the academia or in the industry, and to keep developing further on the design and execution of artist-directed animation in autonomous social robots.

Within our contributions we make a final remark on the ones that we consider our signature, most groundbreaking work, and upon which we hope the future to be built after:

The Principles of Robot Animation

Just as most character animators start by learning Disney's Twelve Principles of Animation, we believe that people working in the field of social robots in which expressivity is a key part of their work, should also have a similar list to provide guidance and remarks on the use of animation principles with robotic embodiments. The Principles of Robot Animation we first drawn in 2012, and further refined up to 2019, based on the increased experience that we gathered through those years. While it may have seemed pretentious to launch a new field through a set of principles (in 2012), we actually saw that initial version as the layout of a foundation to discuss with our peers in order to

allow them to be further refined throughout the years. We do not ignore the fact that Thomas & Johnston did not write the Disney principles in the early stage of their work, but after 60 years of experience. Therefore as of today, we still do not consider the Principles of Robot Animation to be concluded - instead they are likely to be revised over the years, during which new principles may even arise, or old ones may be removed, reframed or merged.

ERIK

ERIK is an expressive inverse kinematics technique that allows a manipulator-like robot to perform expressive poses towards given directions, which enhances such a robot's ability to interact socially with humans. It can run in real-time and adapt to different embodiments and expressive poses without requiring any additional demonstration or training. The challenge of developing ERIK was one that was most notably filled with uncertainty in the beginning, and overgrown in reward at the end. In the field of inverse kinematics we clearly see a marked difference between robotics and CGI. Within 3d animation software and video games there are nearly infinite possibilities for animating 3d characters in real-time - and they can be animated from a timeline, from motion capture, from an AI that controls a walk-cycle and full-body IK, etc. Because robotics deals with the kinematics at a much more precise and consistent level - the physical level - the techniques used are much more difficult to tweak without breaking the mathematical principles under which they lie. Our approach of tackling a grey area - it must work on robots, but it doesn't have to be *that perfect* - allows us to think out of the box, and address the problem from a different perspective. Additionally, it introduced us to new world of possibilities regarding what we prefer to call expressive kinematics. That is because such techniques nowadays do not remain limited to pure, close-form inverse kinematics solutions, but may become quite more hybrid especially when we introduce expressive goals into the problem. The current implementation of ERIK is quite complex and is likely not to be the must efficient computationally. It also lacks some of the expected features that initially drove us to FABRIK, such as the use of multiple end-effectors, and consequently an application featuring a full-body ERIK-controlled robotic character. However future work will allow us to refine the algorithm to make it easier to replicate and understand, while also extending it with those features. Although more research is also required to understand what are the actual limits of ERIK in terms of precison-control, we successfully concluded that it poses as a technique that may be used across various applications and embodiments, except ones in which precision is critical, such as when the robot engages in tactile interaction with humans, when performing pick'n'place tasks, or other precision-oriented tasks such as surgery or using tools. However in a case such as surgery in particular, we would not even advocate for the use of ERIK given that in such an extremely critical task expressivity needs not need to play a role.

Nutty Tracks and Pipeline

Nutty Tracks is the programmable animation engine we have developed to support the exploration and development of our proposed robot animation principles and practices. The Pipeline at its core defines its capabilities and can be further mimicked into future animation engines. Throughout much of our work, Nutty Tracks has become our signature technology. Its concept, dating back to 2013, still supports our needs and current HRI scenarios without having suffered any major revamping. It was created to allow us to experiment new robot animation techniques that we could not easily explore otherwise. That purpose is also where it got its name from. We followed on a common practice among the major animation studios in the early days of animation. All of the studios had their

own experimental animation department, in which they would develop new techniques either in the animation process or its production, which would be used to create series of short animated cartoon movies. Such series were named using two terms: one word that is an adjective to *joyfully crazy*; another that represents some kind of musical piece. Therefore they created names we all know such as *Looney Tunes* and *Merrie Melodies* (Warner Bros.), *Silly Symphonies* (Disney) or *Happy Harmonies* (Metro-Goldwyn-Mayer). We followed by introducing Nutty Tracks, in particular due to a personal fascination with animated squirrels (*Nutty*), and the fact that *Track* worked both as a music track/sound track, while also relating to the paradigm of Nutty's layer-based visual appearance. There are plenty more ideas that we hope to keep exploring and developing through Nutty Tracks, namely in order to reduce even more the code required to support new embodiments, to create a more flexible motion filter designer, and to keep enhancing its inverse kinematics abilities. By now we consider it to have achieved its goals by having been the main *sandbox* for our different techniques and practices in HRI, and the most successfully of our endeavours. In the future we expect to continue developing Nutty Tracks or even to create new polished versions of it from scratch. In any case its Nutty Pipeline, internal mechanics and GUI concept will remain as what we consider to be the foundation for any further programmable robot animation engines.

Nutty Motion Filter

The Nutty Motion Filter is a signal processor that allows to ensure C^1- to C^3-continuity on the motion signal that is output from an animation engine into a virtual or robotic character controller. Since the beginning, our approach of taking methods from CGI and applying them to robots, has required a way to ensure that the resulting motion is continuous and smooth. Through the years we have used ad-hoc solutions until we finally developed the Nutty Motion Filter, which allows us to control several characteristics of the resulting signal, in a way that keeps the signal smooth and controlled, while also providing parameters that allow to transfer expressivity into the resulting motion. There is still a lot of work to be done here, which will be addressed in the future, namely, that no evaluation of the expressive capabilities of the NMF was actually performed. The filter can be used as a signal processor in a wide range of applications. It can be used e.g. in joint motion, on the interpolation of LED lights and of display features, or on spacial motion. An extended evaluation across various applications will allow us to further establish templates and use-principles for the NMF. In particular, we expect to use the filter to just animate the motion of virtual abstract shapes in a 2D view, and by changing the parameters, access if in fact, the motion through the same trajectory points, filtered using different parameters, conveys different expressivity or not.

Adelino

Adelino is a low-fidelity craft manipulator-like robot with a minimalist expressive face that was designed to resemble a snake, or a pure *line of action*, and was built and used especially to challenge the limits of what we could achieve in robot animation using Nutty Tracks and ERIK. Despite its rudimentary build quality, Adelino has highly exceeded our initial expectations regarding what it could be used for in HRI applications. Even with its low-fidelity motors and shakiness, its design is shown to be appropriate for non-verbal expressive behaviour. While building new Adelino-type robots using the same materials still remains an option, we advocate that its design and mechanics can pose as a paradigm for future robot designs, even with more solid materials and motors, in various dimensions, with various number of links, and even slightly different head formats. The use of a manipulator-like robot as a full-body

expressive character in which its end-effector is used as an expressive head, and also the idea that this head may be symmetric in order to flip upside-down depending on the kinematic constraints, has shown to be a success and is very welcome by the community. Further development at the IK level may be taken to provide even better results on Adelino-type robots, similarly to how the $\Xi_{\text{SymmetricEndpoint}}$ extension was developed for ERIK to enhance its support of Adelino.

SERA

SERA is the toolkit and agent architecture that has supported all the HRI scenarios we have worked on and presented throughout this thesis. Since we started developing software for authoring and execution of HRI scenarios in 2012, many tools have been built, such as Thalamus, Skene, and later Nutty Tracks. At some point we realized that we were following nearly the same structure and procedure on different application and even using different robots. The toolkit developed at the INESC-ID's GAIPS lab was therefore turned into what became know as the SERA architecture and toolkit. The toolkit aggregated our tools so that other people could access and use them, while the architecture defined the guidelines on how to set-up the components of the scenario. The SERA toolkit still remains as the main HRI software used at GAIPS even with nearly no development made during the last two years. It has become part of the lab's legacy by providing new students and researchers with a set of tools that can readily be used to prototype and develop full HRI scenarios consistently and robustly, and that allows both technical and non-technical parts (such as psychologists) to work together on those scenarios. As such the overall SERA toolkit (which includes Nutty Tracks) does remain as the major technological legacy that is yielded from the work that has lead up to this thesis. Just like in the case of Nutty Tracks, few developments were made in the past years, although many ideas for features are left to implement. Therefore we expect to keep SERA in an open development state and even to make it fully available to the general public. Before we do so however, the requirement we have set is to develop more documentation both on the code and on the use of the tools.

Bibliography

[1] G. Bendazzi. *Cartoons: One Hundred Years of Cinema Animation*. Indiana University Press, 1994. ISBN 9780253209375.

[2] R. Fleischer. *Out of the Inkwell: Max Fleischer and the Animation Revolution*. University Press of Kentucky, 2005. ISBN 9780813137117.

[3] J. Canemaker. *Felix: the twisted tale of the world's most famous cat*. Da Capo Press, 1996. ISBN 9780306807312.

[4] F. Thomas and O. Johnston. *The Illusion of Life: Disney Animation*. Hyperion, 1995. ISBN 0786860707.

[5] J. Canemaker. *Tex Avery: The MGM years, 1942-1955*. Turner Publishing, 1996.

[6] J. Bates. The role of emotion in believable agents. *Communications of the ACM*, 37(7):122–125, 1994. ISSN 00010782. doi: 10.1145/176789.176803.

[7] C. Breazeal. Towards Sociable Robots. *Robotics and Autonomous Systems*, 42(3-4):167–175, 2008.

[8] C. Bartneck and J. Forlizzi. A design-centred framework for social human-robot interaction. *RO-MAN 2004. 13th IEEE International Workshop on Robot and Human Interactive Communication*, pages 591–594, 2004. doi: 10.1109/ROMAN.2004.1374827.

[9] A. V. Breemen. Animation engine for believable interactive user-interface robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems - IROS '04*, volume 3, pages 2873–2878, 2004. ISBN 0-7803-8463-6. doi: 10.1109/IROS.2004.1389845.

[10] D. A. Norman. *Emotional Design: Why We Love (or Hate) Everyday Things*. Basic Books, 1 edition, dec 2003. ISBN 0465051359.

[11] U. Hess. The communication of emotion. *World*, (1872):397–409, 2001.

[12] P. Ekman and W. Friesen. *Facial Action Coding System: A Technique for the Measurement of Facial Movement.* Consulting Psychologists Press, Palo Alto, 1978.

[13] A. Ortony, G. L. Clore, and A. Collins. *The Cognitive Structure of Emotions*. Cambridge University Press, jul 1988. ISBN 0521353645.

[14] C. Bartneck. *eMuu - An Embodied Emotional Character for the Ambient Intelligent Home. PhD Thesis*. PhD thesis, 2002.

[15] A. Mehrabian. Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in Temperament. *Current Psychology*, 14(4):261–292, dec 1996. ISSN 0737-8262. doi: 10.1007/BF02686918.

[16] P. Ekman and Friesen. The Repertoire Of Nonverbal Behavior - Categories, Origins, Usage and Coding, 1969.

[17] J. Allwood. Bodily Communication Dimensions of Expression and Content. *Language and Speech*, pages 1–15, 2002.

[18] M. Argyle. *Bodily Communication*. University paperbacks. Methuen, 1988. ISBN 9780416381504.

[19] H. Wallbott. Bodily expression of emotion. *European journal of social psychology*, 896(November 1997), 1998.

[20] A. Kendon. *Gesture: Visible Action as Utterance*. Gesture: Visible Action as Utterance. Cambridge University Press, 2004. ISBN 9780521542937.

[21] J. Lasseter. Principles of traditional animation applied to 3D computer animation. *ACM International Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '87*, 21(4):35–44, 1987. ISSN 00978930. doi: 10.1145/37402.37407.

[22] T. Ribeiro and A. Paiva. The Illusion of Robotic Life Principles and Practices of Animation for Robots. In *ACM/IEEE International Conference on Human-Robot Interaction - HRI '12*, number 1937, pages 383–390, 2012. ISBN 9781450310635.

[23] J. Beck. *The Animated Movie Guide*. Cappella Bks. Chicago Review Press, 2005. ISBN 9781556525919.

[24] C. Jones. *Chuck Amuck: The Life and Times of an Animated Cartoonist*. Farrar Straus Giroux, 1999. ISBN 9780374526207.

[25] C. Finch. *Jim Henson: The Works*. Random House, 1993. ISBN 0679412034.

[26] M. J. Lyons, R. Campbell, A. Plante, M. Coleman, M. Kamachi, and S. Akamatsu. The Noh mask effect: vertical viewpoint dependence of facial expression perception. *Proceedings. Biological sciences / The Royal Society*, 267(1459):2239–45, nov 2000. ISSN 0962-8452. doi: 10.1098/rspb.2000.1274.

[27] K. Komparu. *The Noh theater: principles and perspectives*. Floating World Editions, 2005. ISBN 9781891640179.

[28] S. Roberts. *Character Animation in 3D*. Elsevier, 2004. ISBN 0240516656.

[29] C. Breazeal, A. Brooks, J. Gray, M. Hancher, J. Mcbean, D. Stiehl, and J. Strickon. Interactive Theatre. *Communications of the ACM*, 46(7):76–84, 2003.

[30] C. Breazeal. Emotion and sociable humanoid robots. *International Journal of Human-Computer Studies*, 59 (1-2):119–155, jul 2003. ISSN 10715819. doi: 10.1016/S1071-5819(03)00018-1.

[31] L. Takayama, D. Dooley, and W. Ju. Expressing thought. In *ACM/IEEE International Conference on Human-Robot Interaction - HRI '11*, page 69, 2011. ISBN 9781450305617. doi: 10.1145/1957656.1957674.

[32] M. J. Gielniak and A. L. Thomaz. Enhancing interaction through exaggerated motion synthesis. *ACM/IEEE International Conference on Human-Robot Interaction - HRI '12*, page 375, 2012. doi: 10.1145/2157689.2157813.

[33] W. S. N. Reilly. Believable Social and Emotional Agents, 1996.

[34] J. Bates, A. B. Loyall, and W. S. Reilly. An Architecture for Action , Emotion , and Social Behavior. *Science*, i(May), 1992.

[35] K. Perlin and A. Goldberg. Improv: A System for Scripting Interactive Actors in Virtual Worlds. *ACM International Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '96*, (212): 205–216, 1996. ISSN 00978930. doi: 10.1145/237170.237258.

[36] N. Badler. Real-time virtual humans. *IEEE Workshop on Non-Rigid and Articulated Motion*, (June):28–36, 1997. doi: 10.1109/PCCGA.1997.626166.

[37] S. Kopp, B. Krenn, and S. Marsella. Towards a common framework for multimodal generation: The behavior markup language. In *International Conference on Intelligent Virtual Agents - IVA '06*, pages 205–217, 2006.

[38] S. A. Moubayed, M. Baklouti, M. Chetouani, T. Dutoit, A. Mahdhaoui, J.-C. Martin, S. Ondas, C. Pelachaud, J. Urbain, and M. Yilmaz. Multimodal Feedback from Robots and Agents in a Storytelling Experiment. *Digital Media*, 2008.

[39] R. Niewiadomski, E. Bevacqua, M. Mancini, and C. Pelachaud. Greta : an interactive expressive ECA system. *Environments*, pages 1399–1400, 2009.

[40] R. Niewiadomski, M. Obaid, E. Bevacqua, J. Looser, Q. A. Le, and C. Pelachaud. Cross-media agent platform. 1(212):11–20, 2011.

[41] T. Ribeiro, E. Di Tullio, L. J. Corrigan, A. Jones, F. Papadopoulos, R. Aylett, G. Castellano, and A. Paiva. Developing Interactive Embodied Characters using the Thalamus Framework: A Collaborative Approach. In *14th International Conference on Intelligent Virtual Agents*, 2014.

[42] T. Ribeiro, A. Pereira, E. Di Tullio, and A. Paiva. The SERA ecosystem: Socially Expressive Robotics Architecture for Autonomous Human-Robot Interaction. In *AAAI Spring Symposium*, 2016.

[43] M. Schröder. *The SEMAINE API: A component integration framework for a naturally interacting and emotionally competent Embodied Conversational Agent*. PhD thesis, 2012.

[44] M. Kriegel, R. Aylett, M. Vala, and A. Paiva. Robots Meet IVAs : A Mind-Body Interface for Migrating Artificial Intelligent Agents. In *International Conference on Intelligent Virtual Agents - IVA '11*, pages 282–295, 2011.

[45] M. Quigley and B. Gerkey. ROS: an open-source Robot Operating System. *ICRA workshop on open source software.*, 3(3.2), 2009.

[46] Open Source Robotics Foundation. ROS - Robot Operating System. `http://www.ros.org`. Accessed: 2019-10-21.

[47] J. Cassell and T. Bickmore. BEAT : the Behavior Expression Animation Toolkit. *Knowledge Creation Diffusion Utilization*, (August):12–17, 2001.

[48] Y. Arafa and A. Mamdani. Scripting Embodied Agents Behaviour with CML : Character Markup Language. *ACM International Conference on Intelligent User Interfaces - IUI '03*, 2003.

[49] H. Prendinger. MPML: a markup language for controlling the behavior of life-like characters. *Journal of Visual Languages & Computing*, 15(2):183–203, apr 2004. ISSN 1045926X. doi: 10.1016/j.jvlc.2004.01.001.

[50] B. D. Carolis, C. Pelachaud, and I. Poggi. APML , a Mark-up Language for Believable Behavior Generation. *Informatica*, pages 1–22, 2004.

[51] N. Badler, J. Allbeck, L. Zhao, and M. Byun. Representing and Parameterizing Agent Behaviors. *Analysis*, 2002.

[52] J. Hodgson. *Mastering Movement: The Life and Work of Rudolf Laban*. Theatre Arts. Routledge, 2001. ISBN 9780878300808.

[53] M. Schröder. First suggestions for an emotion annotation and representation language. *Proceedings of LREC*, pages 3–6, 2006.

[54] S. Kopp and I. Wachsmuth. Synthesizing multimodal utterances for conversational agents. *Computer Animation and Virtual Worlds*, 15(1):39–52, mar 2004. ISSN 1546-4261. doi: 10.1002/cav.6.

[55] MindMakers. Bml 1.0 standard. `http://www.mindmakers.org/projects/bml-1-0/wiki`. Accessed: 2019-10-21.

[56] A. Heloir and M. Kipp. EMBR: A realtime animation engine for interactive embodied agents. *3rd International Conference on Affective Computing and Intelligent Interaction and Workshops*, 1:1–2, sep 2009. doi: 10.1109/ACII.2009.5349524.

[57] B. Tomlinson. From linear to interactive animation: how autonomous characters change the process and product of animating. *Computers in Entertainment (CIE)*, 3(1):1–20, 2005.

[58] M. Thiebaux, M. Rey, A. N. Marshall, S. Marsella, and M. Kallmann. SmartBody : Behavior Realization for Embodied Conversational Agents. *Information Sciences*, (Aamas):12–16, 2008.

[59] A. W. Feng, Y. Xu, and A. Shapiro. An example-based motion synthesis technique for locomotion and object manipulation. *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games - I3D '12*, page 95, 2012. doi: 10.1145/2159616.2159632.

[60] S. Levine, J. M. Wang, A. Haraux, Z. Popović, and V. Koltun. Continuous character control with low-dimensional embeddings. *ACM Transactions on Graphics*, 31(4):1–10, jul 2012. ISSN 07300301. doi: 10.1145/2185520.2335379.

[61] I. S. Pandzic and R. Forchheimer. *MPEG-4 Facial Animation: The Standard, Implementation and Applications*. John Wiley & Sons, Inc., New York, 2003.

[62] M. Moussa and Z. Kasap. MPEG-4 FAP animation applied to humanoid robot head. *Proceeding of Summer . . .*, pages 1–11, 2010.

[63] S. Marsella, S. M. Carnicke, J. Gratch, A. Okhmatovskaia, and A. Rizzo. An Exploration of Delsarte's Structural Acting System. In *International Conference on Intelligent Virtual Agents - IVA '06*, volume 4133, pages 80–92, 2006. ISBN 978-3-540-37593-7. doi: 10.1007/11821830.

[64] D. Chi, M. Costa, L. Zhao, and N. Badler. The EMOTE Model for Effort and Shape. *ACM International Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '00*, pages 173–182, 2000. ISSN 0097-8930. doi: 10.1145/344779.352172.

[65] L. Wilke, T. Calvert, R. Ryman, and I. Fox. From dance notation to human animation: The LabanDancer project. In *Computer Animation and Virtual Worlds*, volume 16, pages 201–211, 2005. doi: 10.1002/cav.90.

[66] OpenRAVE. Ikfast. `http://openrave.org/docs/latest_stable/openravepy/ikfast`. Accessed: 2019-10-21.

[67] A. Aristidou, J. Lasenby, Y. Chrysanthou, and A. Shamir. Inverse Kinematics Techniques in Computer Graphics: A Survey. *Computer Graphics Forum*, 37(6):35–58, 2018. ISSN 14678659. doi: 10.1111/cgf.13310.

[68] R. Boulic. Hierarchical kinematic behaviors for complex articulated figures. *Interactive computer animation*, (1):1–27, 1996.

[69] R. Kulpa and F. Multon. Fast inverse kinematics and kinetics solver for human-like figures. *Proceedings of 2005 5th IEEE-RAS International Conference on Humanoid Robots*, 2005:38–43, 2005. doi: 10.1109/ICHR.2005.1573542.

[70] L. Unzueta, M. Peinado, R. Boulic, and Á. Suescun. Full-body performance animation with Sequential Inverse Kinematics. *Graphical Models*, 70(5):87–104, 2008. ISSN 15240703. doi: 10.1016/j.gmod.2008.03.002.

[71] P. Harish, M. Mahmudi, B. L. Callennec, and R. Boulic. Parallel Inverse Kinematics for Multithreaded Architectures. *ACM Transactions on Graphics*, 35(2):1–13, 2016. ISSN 07300301. doi: 10.1145/2887740.

[72] S. R. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *University of California, San Diego, Technical Reports.*, 2009. ISSN 0306-4522. doi: 10.1016/j.neuroscience.2005.01.020.

[73] P. Baerlocher. Inverse kinematics techniques for the interactive posture control of articulated figures. *Epfl*, 2383:1–156, 2001. doi: 10.5075/epfl-thesis-2383.

[74] D. E. Orin and W. W. Schrader. Efficient Computation of the Jacobian for Robot Manipulators. *The International Journal of Robotics Research*, 3(4):66–75, 1984. ISSN 17413176. doi: 10.1177/027836498400300404.

[75] S. R. Buss and J.-S. Kim. Selectively Damped Least Squares for Inverse Kinematics. *Journal of Graphics, GPU, and Game Tools*, 10(3):37–49, 2005. ISSN 2151-237X. doi: 10.1080/2151237X.2005.10129202.

[76] A. A. Maciejewski and C. A. Klein. Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments. *The International Journal of Robotics Research*, 4(3):109–117, 1985. ISSN 17413176. doi: 10.1177/027836498500400308.

[77] A. A. Maciejewski and C. A. Klein. Numerical filtering for the operation of robotic manipulators through kinematically singular configurations. *Journal of Robotic Systems*, 5(6):527–552, 1988. ISSN 10974563. doi: 10.1002/rob.4620050603.

[78] M. Neff and E. Fiume. Methods for exploring expressive stance. *Graphical Models*, 68(2):133–157, 2006. ISSN 15240703. doi: 10.1016/j.gmod.2005.03.003.

[79] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović. Style-based inverse kinematics. *ACM Transactions on Graphics*, 23:522, 2004. ISSN 07300301. doi: 10.1145/1015706.1015755.

[80] N. Courty and E. Arnaud. Sequential Monte Carlo Inverse Kinematics. *Research Report*, 24:194947–1, 2008.

[81] C. Hecker, B. Raabe, R. W. Enslow, J. DeWeese, J. Maynard, and K. van Prooijen. Real-time motion retargeting to highly varied user-created morphologies. *ACM Transactions on Graphics*, 27(3):1, aug 2008. ISSN 07300301. doi: 10.1145/1360612.1360626.

[82] L. Wang and C. Chen. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 7(4):489–499, 1991. ISSN 1042-296X. doi: 10.1109/70.86079.

[83] D. Merrick and T. Dwyer. Skeletal Animation for the Exploration of Graphs. *Australasian Symposium on Information Visualisation, (invis.au'04)*, 35:61–70, 2004.

[84] M. P. Johnson. *Exploiting Quaternions to Support Expressive Interactive Character Motion*. PhD thesis, Massachusetts Institute of Technology, 2003.

[85] A. Aristidou and J. Lasenby. Inverse Kinematics: a review of existing techniques and introduction of a new fast iterative solver. *University of Cambridge*, 2009.

[86] P. Baerlocher and R. Boulic. Parametrization and range of motion of the ball-and-socket joint. *IFIP Advances in Information and Communication Technology*, 68:180–190, 2001. ISSN 18684238. doi: 10.1007/978-0-306-47002-8.

[87] C. Breazeal, A. Brooks, M. Hancher, J. Strickon, C. Kidd, J. McBean, and D. Stiehl. Public Anemone: An Organic Robot Creature. *ACM SIGGRAPH 2002 Conference Abstracts and Applications*, page 76, 2002. doi: 10.1145/1242073.1242111.

[88] A. Aristidou and J. Lasenby. FABRIK: A fast, iterative solver for the Inverse Kinematics problem. *Graphical Models*, 73(5):243–260, 2011. ISSN 15240703. doi: 10.1016/j.gmod.2011.05.003.

[89] A. Aristidou, Y. Chrysanthou, and J. Lasenby. Extending FABRIK with model constraints. *Computer Animation and Virtual Worlds*, 27(1):35–57, 2016. ISSN 1546427X. doi: 10.1002/cav.1630.

[90] J. Brown, J. C. Latombe, and K. Montgomery. Real-time knot-tying simulation. *Visual Computer*, 20(2-3): 165–179, 2004. ISSN 01782789. doi: 10.1007/s00371-003-0226-y.

[91] S. Starke. *A Hybrid Genetic Swarm Algorithm for Interactive Inverse Kinematics. MSc Thesis*. PhD thesis, University of Hamburg, 2016.

[92] K. Dautenhahn. Design spaces and niche spaces of believable social robots. *Proceedings. 11th IEEE International Workshop on Robot and Human Interactive Communication*, pages 192–197, 2002. doi: 10.1109/ROMAN.2002.1045621.

[93] B. Meerbeek, M. Saerbeck, and C. Bartneck. Iterative design process for robots with personality. In *Symposium on New Frontiers in Human-Robot Interaction*, pages 94–101, Edimburgh, 2009.

[94] C. L. Bethel. *Robots without faces : Non-verbal social human- robot interaction. PhD Thesis*. PhD thesis, 2009.

[95] M. Saerbeck and C. Bartneck. Perception of Affect Elicited by Robot Motion. *Journal of Personality*, pages 53–60, 2010. doi: 10.1145/1734454.1734473.

[96] G. Hoffman and W. Ju. Designing Robots With Movement in Mind. *Journal of Human-Robot Interaction*, 3 (1):89, 2014. ISSN 2163-0364. doi: 10.5898/JHRI.3.1.Hoffman.

[97] H. Knight. Expressive Motion for Low Degree-of-Freedom Robots. *PhD Thesis, CMU-RI-TR-16-51, Carnegie Mellon University*, 2016.

[98] T. Schulz, J. Torresen, and J. Herstad. Animation Techniques in Human-Robot Interaction User Studies: a Systematic Literature Review. 2018. URL `http://arxiv.org/abs/1812.06784`.

[99] M. Veloso, P. Rybski, S. Lenser, S. Chernova, and D. Vail. CMRoboBits: Creating an intelligent AIBO robot. *AI Magazine*, 27(1), 2006. ISSN 07384602. doi: http://dx.doi.org/10.1609/aimag.v27i1.1864.

[100] J. Osada, S. Ohnaka, and M. Sato. The scenario and design process of childcare robot, PaPeRo. *Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology - ACE '06*, page 80, 2006. doi: 10.1145/1178823.1178917.

[101] J. Y. Sung, R. E. Grinter, and H. I. Christensen. Domestic robot ecology: An initial framework to unpack long-term acceptance of robots at home. *International Journal of Social Robotics*, 2(4):417–429, 2010. ISSN 18754791. doi: 10.1007/s12369-010-0065-8.

[102] D. Rea, J. Young, and P. Irani. The Roomba mood ring: An ambient-display robot. *ACM/IEEE International Conference on Human-Robot Interaction - HRI '12*, pages 217–218, 2012.

[103] A. Singh and J. Young. Animal-inspired human-robot interaction: A robotic tail for communicating state. *ACM/IEEE International Conference on Human-Robot Interaction - HRI '12*, pages 237–238, 2012.

[104] J. Gray, G. Hoffman, S. O. Adalgeirsson, M. Berlin, and C. Breazeal. Expressive, interactive robots: Tools, techniques, and insights based on collaborations. In *ACM/IEEE International Conference on Human-Robot Interaction - HRI '10 - Workshop on What do Collaborations with the Arts Have to Say About Human-Robot Interaction*, 2010.

[105] C. Breazeal and B. Scassellati. How to build robots that make friends and influence people. (April 2013): 858–863, 2003. doi: 10.1109/iros.1999.812787.

[106] F. Tanaka, K. Noda, T. Sawada, and M. Fujita. Associated emotion and its expression in an entertainment robot QRIO. *Entertainment Computing–ICEC . . .*, pages 1–6, 2004.

[107] A. L. Thomaz, M. Berlin, and C. Breazeal. An embodied computational model of social referencing. *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, 2005:591–598, 2005. doi: 10.1109/ROMAN.2005.1513844.

[108] C. Breazeal, A. Brooks, D. Chilongo, J. Gray, G. Hoffman, C. Kidd, H. Lee, J. Lieberman, and A. Lockerd. Working collaboratively with humanoid robots. *IEEE/RAS International Conference on Humanoid Robots - Humanoids '04*, 1:253 – 272, 2004. doi: 10.1109/ICHR.2004.1442126.

[109] X. Shusong. From character animation to robot motion. *IEEE International Conference on Industrial Informatics - INDIN '08*, pages 57–62, jul 2008. doi: 10.1109/INDIN.2008.4618066.

[110] X. Shusong and H. Jiefeng. Robot behavior expression by illusion of life. *IEEE Conference on Robotics, Automation and Mechatronics - RAM '08*, pages 994–998, sep 2008. doi: 10.1109/RAMECH.2008.4690892.

[111] A. V. Breemen, X. Yan, and B. Meerbeek. iCat: An animated user-interface robot with personality. In *Proceedings of the International Conference on Autonomous Agents*, pages 17–18, 2005.

[112] I. Leite, G. Castellano, A. Pereira, C. Martinho, and A. Paiva. Empathic Robots for Long-term Interaction: Evaluating Social Presence, Engagement and Perceived Support in Children. *International Journal of Social Robotics*, 6(3):329–341, 2014. ISSN 18754805. doi: 10.1007/s12369-014-0227-1.

[113] H. Kozima, M. P. Michalowski, and C. Nakagawa. Keepon: A playful robot for research, therapy, and entertainment. *International Journal of Social Robotics*, 1(1):3–18, 2009. ISSN 18754791. doi: 10.1007/s12369-008-0009-8.

[114] S. Strohkorb, E. Fukuto, N. Warren, C. Taylor, B. Berry, and B. Scassellati. Improving Human-Human Collaboration Between Children With a Social Robot. In *IEEE International Symposium on Robot and Human Interactive Communication - RO-MAN '16*, pages 1–6, New York City, NY, USA, 2016.

[115] H. Knight and R. Simmons. Laban head-motions convey robot state: A call for robot body language. *IEEE International Conference on Robotics and Automation - ICRA '16*, 2016-June:2881–2888, 2016. ISSN 10504729. doi: 10.1109/ICRA.2016.7487451.

[116] E. C. Grigore, A. Pereira, I. Zhou, and D. Wang. Talk to Me : Verbal Communication Improves Perceptions of Friendship and Social Presence in Human-Robot Interaction. In *International Conference on Intelligent Virtual Agents - IVA'16*, Los Angeles, CA, USA, 2016.

[117] M. Siegel, C. Breazeal, and M. I. Norton. Persuasive Robotics : the influence of robot gender on human behavior. *IEEE/RSJ International Conference on Intelligent Robots and Systems - IROS '09*, pages 2563–2568, 2009.

[118] K. Wada and T. Shibata. Robot therapy in a care house - Change of relationship among the residents and seal robot during a 2-month long study. *IEEE International Symposium on Robot and Human Interactive Communication - RO-MAN '07*, 23(5):107–112, 2007. ISSN 1552-3098. doi: 10.1109/ROMAN.2007.4415062.

[119] N. A. Torres, N. Clark, I. Ranatunga, and D. Popa. Implementation of Interactive Arm Playback Behaviors of Social Robot Zeno for Autism Spectrum Disorder Therapy. *International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '12*, 3(January):21:1—-21:7, 2012. doi: 10.1145/2413097.2413124.

[120] G. Hoffman, R. Kubat, and C. Breazeal. A hybrid control system for puppeteering a live robotic stage actor. *IEEE International Symposium on Robot and Human Interactive Communication - RO-MAN '08*, pages 354–359, aug 2008. doi: 10.1109/ROMAN.2008.4600691.

[121] T. Ribeiro, D. Dooley, and A. Paiva. Nutty Tracks - Symbolic Animation Pipeline for Expressive Robotics. *ACM International Conference on Computer Graphics and Interactive Techniques Posters - SIGGRAPH '13*, page 4503, 2013.

[122] J. Kennedy, P. Baxter, E. Senft, and T. Belpaeme. Social Robot Tutoring for Child Second Language Learning. In *ACM/IEEE International Conference on Human-Robot Interaction - HRI '16*, 2016.

[123] A. Ramachandran, A. Litoiu, and B. Scassellati. Shaping Productive Help-Seeking Behavior During Robot-Child Tutoring Interactions. In *ACM/IEEE International Conference on Human-Robot Interaction - HRI '16*, 2016.

[124] J. Greczek, K. Swift-Spong, and M. Matarić. Using Eye Shape to Improve Affect Recognition on a Humanoid Robot with Limited Expression. *University of Southern California Technical Reports*, pages 1–10, 2011.

[125] M. Veloso, J. Biswas, B. Coltin, S. Rosenthal, T. Kollar, C. Mericli, M. Samadi, S. Brandao, and R. Ventura. CoBots: Collaborative robots servicing multi-floor buildings. *IEEE/RSJ International Conference on Intelligent Robots and Systems - IROS '12*, pages 5446–5447, 2012. ISSN 21530858. doi: 10.1109/IROS.2012.6386300.

[126] K. Baraka and A. Paiva. Expressive Lights for Revealing Mobile Service Robot State Light Interface for State Revealing. pages 17–23, 2015.

[127] S. S. Srinivasa, D. Ferguson, C. J. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and M. V. Weghe. HERB: A home exploring robotic butler. *Autonomous Robots*, 28(1):5–20, 2010. ISSN 09295593. doi: 10.1007/s10514-009-9160-9.

[128] S. S. Srinivasa, D. Berenson, M. Cakmak, A. Collet, M. R. Dogar, A. D. Dragan, R. A. Knepper, T. Niemueller, K. Strabala, M. Vande Weghe, and J. Ziegler. Herb 2.0: Lessons learned from developing a mobile

manipulator for the home. *Proceedings of the IEEE*, 100(8):2410–2428, 2012. ISSN 00189219. doi: 10.1109/JPROC.2012.2200561.

[129] A. D. Dragan, S. Bauman, J. Forlizzi, and S. S. Srinivasa. Effects of Robot Motion on Human-Robot Collaboration. *ACM/IEEE International Conference on Human-Robot Interaction - HRI '15*, 1:51–58, 2015. ISSN 21672148. doi: 10.1145/2696454.2696473.

[130] H. Admoni. *Nonverbal Communication in Socially Assistive Human-Robot Interaction*. PhD thesis, Yale University, 2016.

[131] G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori. The iCub humanoid robot: an open platform for research in embodied cognition. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, pages 50–56, 2008. ISBN 9781605582931.

[132] E. H. Kim, S. S. Kwak, K. H. Hyun, S. H. Kim, and Y. K. Kwak. Design and Development of an Emotional Interaction Robot, Mung. *Advanced Robotics*, 23(6):767–784, apr 2009. ISSN 01691864. doi: 10.1163/156855309X431712.

[133] M. Lee, J. Forlizzi, P. Rybski, and F. Crabbe. The snackbot: documenting the design of a robot for long-term human-robot interaction. *ACM/IEEE International Conference on Human-Robot Interaction - HRI '09*, 2009.

[134] M. Lee, J. Forlizzi, and S. Kiesler. Personalization in HRI: A longitudinal field experiment. *ACM/IEEE International Conference on Human-Robot Interaction - HRI '12*, pages 319–326, 2012. ISSN 2167-2121. doi: 10.1145/2157689.2157804.

[135] G. Hoffman and G. Weinberg. Gesture-based human-robot Jazz improvisation. In *IEEE International Conference on Robotics and Automation - ICRA '10*, pages 582–587, 2010. ISBN 978-1-4244-5038-1. doi: 10.1109/ROBOT.2010.5509182.

[136] G. Hoffman. Dumb robots, smart phones: A case study of music listening companionship. *IEEE International Symposium on Robot and Human Interactive Communication - RO-MAN '12*, pages 358–363, sep 2012. doi: 10.1109/ROMAN.2012.6343779.

[137] R. Mead and M. J. Mataric. Automated Caricature of Robot Expressions in Socially Assistive Human-Robot Interaction. *ACM/IEEE International Conference on Human-Robot Interaction - HRI '10 - Workshop on What do Collaborations with the Arts Have to Say About Human-Robot Interaction*, mar 2010. ISSN 1573-3432.

[138] J. Kędzierski, R. Muszyński, C. Zoll, A. Oleksy, and M. Frontkiewicz. Emys—emotive head of a social robot. *International Journal of Social Robotics*, 5(2):237–249, Apr 2013. ISSN 1875-4805. doi: 10.1007/s12369-013-0183-1.

[139] A. Pereira, R. Prada, and A. Paiva. Improving social presence in human-agent interaction. In *ACM Conference on Human Factors in Computing Systems - CHI '14*, pages 1449–1458. ACM, 2014. ISBN 9781450324731.

[140] F. Correia, P. Alves-Oliveira, N. Maia, T. Ribeiro, S. Petisca, F. S. Melo, and A. Paiva. Just follow the suit! Trust in human-robot interactions during card game playing. *IEEE International Symposium on Robot and*

Human Interactive Communication (RO-MAN), (September):507–512, 2016. doi: 10.1109/ROMAN.2016.7745165.

[141] J. Saldien, B. Vanderborght, K. Goris, M. Van, and D. Lefeber. A Motion System for Social and Animated Robots. *International Journal of Advanced Robotic Systems*, page 1, 2014. ISSN 1729-8806. doi: 10.5772/58402.

[142] M. Faria and A. Costigliola. Follow me : Communicating Intentions with a Spherical Robot. In *IEEE International Symposium on Robot and Human Interactive Communication - RO-MAN '16*, number August, 2016. ISBN 9781509039289.

[143] H. Yin, F. S. Melo, A. Billard, A. Paiva, and U. D. Lisboa. Synthesizing Robotic Handwriting Motion by Learning from Human Demonstrations. *International Joint Conference on Artificial Intelligence - IJCAI '16*, pages 3530–3537, 2016.

[144] M. Faria, R. Silva, F. S. Melo, and A. Paiva. Me and you together: A study on collaboration in manipulation tasks. *AAAI Fall Symposium Series*, FS-16-01 -(PG - 32-35):32–35, 2016.

[145] A. A. M. Setapen. Creating robotic characters for long-term interaction. *Thesis (MSc) - Massachusetts Institute of Technology, School of Architecture and Planning, Program in Media Arts and Sciences*, 2012.

[146] E. Short, K. Swift-Spong, J. Greczek, A. Ramachandran, A. Litoiu, E. C. Grigore, D. Feil-Seifer, S. Shuster, J. J. Lee, S. Huang, S. Levonisova, S. Litz, J. Li, G. Ragusa, D. Spruijt-Metz, M. Mataric, and B. Scassellati. How to train your DragonBot: Socially assistive robots for teaching children about nutrition through play. *IEEE International Symposium on Robot and Human Interactive Communication - RO-MAN '14*, pages 924–929, 2014. doi: 10.1109/ROMAN.2014.6926371.

[147] T. Klamer, S. Allouch, and D. Heylen. "Adventures of Harvey"–Use, Acceptance of and Relationship Building with a Social Robot in a Domestic Environment. *Human-Robot Personal Relationships*, (2):74–82, 2011. doi: 10.1007/978-3-642-19385-9.

[148] P. Wallis. A robot in the kitchen. *Association For Computational Linguistics ACL '10 - Workshop on Companionable Dialogue Systems - CDS '10*, 2010. ISSN 00189235. doi: 10.1109/MSPEC.2010.5434835.

[149] S. A. Moubayed. Towards rich multimodal behavior in spoken dialogues with embodied agents. *4th IEEE International Conference on Cognitive Infocommunications, CogInfoCom 2013 - Proceedings*, pages 817–822, 2013. doi: 10.1109/CogInfoCom.2013.6719212.

[150] S. A. Moubayed, J. Edlund, and J. Beskow. Taming Mona Lisa. *ACM Transactions on Interactive Intelligent Systems*, 1(2):1–25, 2012. ISSN 21606455. doi: 10.1145/2070719.2070724.

[151] I. Aaltonen, A. Arvola, P. Heikkilä, and H. Lammi. Hello Pepper, May I Tickle You? pages 53–54, 2017. doi: 10.1145/3029798.3038362.

[152] A. Rossi, F. Garcia, A. C. Maya, K. Dautenhahn, K. L. Koay, M. L. Walters, and A. K. Pandey. Investigating the Effects of Social Interactive Behaviours of a Robot on People's Trust During a Navigation Task. *Lecture*

Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11649 LNAI(July):349–361, 2019. ISSN 16113349. doi: 10.1007/978-3-030-23807-0_29.

[153] J. Lafaye, D. Gouaillier, and P.-B. Wieber. Linear model predictive control of the locomotion of Pepper, a humanoid robot with omnidirectional wheels. *IEEE/RAS International Conference on Humanoid Robots - Humanoids '14*, (November):336–341, 2014. doi: 10.1109/HUMANOIDS.2014.7041381.

[154] E. Goldberg. *Character Animation Crash Course!* Silman-James Press, 2008. ISBN 978-1-879505-97-1.

[155] G. Hoffman, O. Zuckerman, G. Hirschberger, M. Luria, and T. Shani Sherman. Design and Evaluation of a Peripheral Robotic Conversation Companion. *ACM/IEEE International Conference on Human-Robot Interaction*, 2015-March:3–10, 2015. ISSN 21672148. doi: 10.1145/2696454.2696495.

[156] Blue Frog Robotics. Buddy robot. URL `http://www.bluefrogrobotics.com/robot`. Accessed: 2019-10-21.

[157] T. Ribeiro and A. Paiva. Animating the Adelino Robot with ERIK. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 388–396, Glasgow, UK, 2017. ACM. ISBN 9781450355438.

[158] P. Alves-Oliveira, P. Arriaga, A. Paiva, and G. Hoffman. Guide to build YOLO, a creativity-stimulating robot for children. *HardwareX*, 6(July):e00074, 2019. ISSN 24680672. doi: 10.1016/j.ohx.2019.e00074. URL `https://doi.org/10.1016/j.ohx.2019.e00074`.

[159] P. Alves-Oliveira, S. Gomes, A. Chandak, P. Arriaga, G. Hoffman, and A. Paiva. Software architecture for YOLO, a creativity-stimulating robot. 2019. URL `http://arxiv.org/abs/1909.10823`.

[160] M. Suguitan and G. Hoffman. Blossom: A Handcrafted Open-Source Robot. *ACM Trans. Hum.-Robot Interact.*, 8(1), 2019.

[161] LG Electronics. CLOi robots. URL `http://www.lg.com/global/lg-thinq-appliances/cloi`. Accessed: 2019-10-21.

[162] I. El Makrini, S. A. Elprama, J. Van den Bergh, B. Vanderborght, A. Knevels, C. I. C. Jewell, F. Stals, G. De Coppel, I. Ravyse, J. Potargent, J. Berte, B. Diericx, T. Waegeman, and A. Jacobs. Working with walt: How a cobot was developed and inserted on an auto assembly line. *IEEE Robotics Automation Magazine*, 25(2): 51–58, June 2018.

[163] Zoetic AI. Kiki robot. URL `https://www.kiki.ai/`. Accessed: 2019-10-21.

[164] R. Wistort. Only robots on the inside. *interactions*, 17(2):72–74, 2010.

[165] R. Murphy, T. Nomura, A. Billard, and J. Burke. Human–Robot Interaction. *IEEE Robotics & Automation Magazine*, 17(2):85–89, 2010. ISSN 1070-9932. doi: 10.1109/MRA.2010.936953. URL `http://ieeexplore.ieee.org/document/5481144/`.

[166] T. Fong. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42(3-4):143–166, mar 2003. ISSN 09218890. doi: 10.1016/S0921-8890(02)00372-X.

[167] P. Baxter, J. Kennedy, E. Senft, S. Lemaignan, and T. Belpaeme. From characterising three years of HRI to methodology and reporting recommendations. In *Proceedings of the 11th ACM/IEEE International Conference on Human-Robot Interaction*, number December 2017, pages 391–398, Christchurch, New Zealand, 2016. IEEE. ISBN 9781467383707. doi: 10.1109/HRI.2016.7451777.

[168] P. Alves-Oliveira, D. Küster, A. Kappas, and A. Paiva. Psychological science in HRI: Striving for a more integrated field of research. *Proceedings of the 2016 AAAI Fall Symposium Series on AI-HRI*, 2016. ISSN 1878-1705 (Electronic). doi: 10.1016/j.intimp.2016.04.032.

[169] M. Suguitan and G. Hoffman. Blossom: A Tensile Social Robot Design with a Handcrafted Shell. In *Companion of the 13th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 249–250. ACM, 2018. ISBN 9781450356152. doi: 10.1080/00064246.1978.11414007.

[170] D. Sonnenschein. *Sound Design: The Expressive Power of Music, Voice and Sound Effects in Cinema*. Michael Wiese Productions, 2013. ISBN 9781615932023.

[171] T. Ribeiro, A. Pereira, E. Di Tullio, P. Alves-Oliveira, and A. Paiva. From Thalamus to Skene: High-level behaviour planning and managing for mixed-reality characters. In *Intelligent Virtual Agents - Workshop on Architectures and Standards for IVAs*, 2014.

[172] L. Riek. Wizard of Oz Studies in HRI: A Systematic Review and New Reporting Guidelines. *Journal of Human-Robot Interaction*, 1(1):119–136, aug 2012. ISSN 21630364. doi: 10.5898/JHRI.1.1.Riek.

[173] P. Sequeira, P. Alves-Oliveira, T. Ribeiro, E. Di Tullio, S. Petisca, F. S. Melo, G. Castellano, and A. Paiva. Discovering social interaction strategies for robots from restricted-perception wizard-of-oz studies. *ACM/IEEE International Conference on Human-Robot Interaction*, 2016-April(March):197–204, 2016. ISSN 21672148. doi: 10.1109/HRI.2016.7451752.

[174] T. Akenine-Möller, E. Haines, N. Hoffman, A. Pesce, M. Iwanicki, and S. Hillaire. *Real-Time Rendering 4th Edition*. A K Peters/CRC Press, Boca Raton, FL, USA, 2018. ISBN 978-1-13862-700-0.

[175] BeatBots LLC. Keepon robot. URL `http://beatbots.net/my-keepon`. Accessed: 2016-11-22.

[176] F. Correia, P. Alves-Oliveira, T. Ribeiro, F. S. Melo, and A. Paiva. A social robot as a card game player. *13th AAAI conference on artificial intelligence and interactive digital entertainment*, pages 23–29, 2017.

[177] P. Alves-Oliveira, S. Petisca, F. Correia, N. Maia, and A. Paiva. Social robots for older adults : Framework of activities for aging in place with robots. In *International Conference on Social Robotics*, 2015.

[178] S. Mascarenhas, M. Guimarães, R. Prada, J. Dias, P. A. Santos, K. Star, B. Hirsh, E. Spice, and R. Kommeren. A Virtual Agent Toolkit for Serious Games Developers. *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–7, aug 2018.

[179] F. Correia, S. Petisca, P. Alves-Oliveira, T. Ribeiro, F. S. Melo, and A. Paiva. "I Choose... YOU!" Membership preferences in human–robot teams. *Autonomous Robots*, 43(2):359–373, feb 2019. ISSN 0929-5593. doi: 10.1007/s10514-018-9767-9.

[180] C. Harms and F. Biocca. Internal Consistency and Reliability of the Networked Minds Measure of Social Presence. *Seventh Annual International Workshop: Presence 2004*, pages 246–251, 2004.

[181] A. Aron, E. N. Aron, and D. Smollan. Inclusion of other in the self scale and the structure of interpersonal closeness. *Journal of Personality and Social Psychology*, 63(4):596–612, 1992.

[182] M. Heerink, B. Kröse, V. Evers, and B. Wielinga. Assessing acceptance of assistive social agent technology by older adults: The almere model. *International Journal of Social Robotics*, 2(4):361–375, 2010. ISSN 18754791. doi: 10.1007/s12369-010-0068-5.

[183] C. Bartneck, D. Kulić, E. Croft, and S. Zoghbi. Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots. *International Journal of Social Robotics*, 1 (1):71–81, 2009. ISSN 18754791. doi: 10.1007/s12369-008-0001-3.

[184] C. M. Carpinella, A. B. Wyman, M. A. Perez, and S. J. Stroessner. The Robotic Social Attributes Scale (RoSAS). *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction - HRI '17*, (March):254–262, 2017. ISSN 21672148. doi: 10.1145/2909824.3020208.

Appendix A

ERIK Algorithm

A.1 Algorithmic Specification

One of this paper's major contribution is to share the full algorithmic specification that allows to implement ERIK. However it would be impractical and even unintelligible to present the whole algorithm here in detail. Instead we have done our best to describe in detail only the major parts of it, while presenting either a textual description or a mathematical formulation for the parts that are less particular to ERIK, and which may be understood and implemented by someone with appropriate CGI animation knowledge.

Aiming at a more comprehensible reading experience, we have shifted all the detailed algorithms to the next section of the appendix, A.2. The entry point to the algorithm is the **CalculateERIK** function, outlined in Algorithm 3. This function takes as input the ERIK Parameters (Π), and Hyperparameters (Λ), which have been described in detail in Section 6.3.5.

Some of the macros or functions used in the algorithms are briefly described in the next section. For simplicity, all the quaternions used are rotation quaternions, i.e, quaternions of unit length. As an additional reminder, please note that the child link of the end-point link refers to the posture's SuperPoint (Section 6.3.8).

A.1.1 Description of functions used throughout the algorithms

This section outlines a short description and/or mathematical formulation for some of the auxiliary functions and operations used within ERIK.

EmptySolution(Sk) Return an empty solution for skeleton Sk.

SafeAngle($k, \theta, bCycle = False$) Returns θ' as an angle that is safe for joint k given its maximum and minimum angle limits, while allowing the angle to cycle instead of purely clamping:

$$\theta' = \begin{cases} min(k_{\text{Max}\theta}, max(k_{\text{Min}\theta}, 2\pi \cdot (\frac{\theta}{2\pi} - \lfloor \frac{\theta}{2\pi} \rceil))) & if bCycle \\ min(k_{\text{Max}\theta}, max(k_{\text{Min}\theta}, \theta)) & otherwise \end{cases}$$

SetOriFromParent(k, Θ) Sets k's basis orientation from its parent:

$$\Theta_{k_Q} = \begin{cases} \Theta_{k_{\text{Parent}\,Q}} \cdot \Theta_{k_{\text{Parent}\,L}} & ifnot\ k_{\text{IsRoot}} \\ I & otherwise \end{cases}$$

SetPosFromParent(k, Θ) Sets k's basis position from its parent:

$$\Theta_{k_\rho} = \begin{cases} \Theta_{k_{\text{Parent}\,\rho}} + \text{RotVQ}(k_{\text{Parent}\,\vec{\sigma}}, \Theta_{k_Q}) & ifnot\ k_{\text{IsRoot}} \\ \vec{0} & otherwise \end{cases}$$

SetFrameFromParent(k, Θ) Call $SetOriFromParent(k, \Theta)$ and $SetPosFromParent(k, \Theta)$ and returns the new Θ.

ApplyFK$(\Theta, k = \Theta_{\textbf{Root}})$ Performs Forward Kinematics calculus on solution Θ starting from node k (optional).

RotVQ$(\vec{v}, Q)$ Returns vector V rotated by quaternion Q.

QAA$(\vec{v}, \alpha)$ Returns a normalized quaternion that represents a rotation of α about the axis $\vec{v}$ (axis-angle).

VDiffAsQ$(\vec{v_1}, \vec{v_2})$ Returns the orientation difference between $\vec{v_1}$ and $\vec{v_2}$ as a Quaternion.

QDiff(Q_1, Q_2) Returns rotation difference between Q_1 and Q_2 as a Quaternion.

TBasis(k, Q') Transforms the world-space basis of link k globally by Q' ($k_Q = Q' \cdot k_Q$).

TBasisRoll(k, Q') Transforms the world-space basis of link k locally by Q' ($k_Q = k_Q \cdot Q'$).

VecAngle$(\vec{v_1}, \vec{v_2})$ Angle θ between $\vec{v_1}$ and $\vec{v_2}$ using atan2.

VecAngle$(\vec{v_1}, \vec{v_2}, \vec{r})$ Angle θ between $\vec{v_1}$ and $\vec{v_2}$ using atan2 with $sign(\theta) = -1$ **if** $r \cdot (\vec{v_1} \times \vec{v_2}) < 0$ **else** 1.

LALUT(k, λ) Queries joint k's LALUT for latitude λ.

TargetLatitude$(k, \vec{\tau})$ Calculates the latitude λ for target $\vec{\tau}$ on k's joint model:

$$\varsigma = -1 \textbf{ if } \hat{\tau} \cdot k_{POA} < 0 \textbf{ else } 1$$

$$\lambda = max(k_{\text{Bottom}\lambda}, min(k_{\text{Top}\lambda}, \frac{\hat{\tau} \cdot \hat{\boldsymbol{Y}} + 1}{2}))$$

PitchRA(k) Returns $\vec{r}$ such that:

$$\vec{r} = \begin{cases} k_{\text{Child}\,RA} & \textbf{if } \text{IsTwister}(k) \textbf{ and } k_{\text{IsRoot}} \\ k_{\text{Parent}\,RA} & \textbf{if } \text{IsTwister}(k) \textbf{ and not } k_{\text{IsRoot}} \\ k_{RA} & otherwise \end{cases}$$

EPA$(Q, \vec{u})$ Ensures Positive Axis on quaternion Q based on a rotation axis $\vec{u}$:

$$EPA(Q) = -Q \textbf{ if } \vec{u} \cdot \vec{Q_v} < 0 \textbf{ else } Q$$

IsTwister(L) Returns $True$ if link L's rotation axis is aligned with its own segment.

AvoidJointEdge(k, δ) Performs an angular offset of $\pm\delta$ on joint k if it is currently at its minimum ($+\delta$) or maximum ($-\delta$) value.

AvoidPostureJointEdges(Ψ, δ) Runs $AvoidJointEdge(k, \delta)$ on each joint k in posture Ψ.

BWCD($\Psi|\Theta, \vec{\tau}, \Lambda$) Performs BWCD (Section 6.3.2) on posture Ψ or solution Θ, towards the target direction $\vec{\tau}$.

CCD($\Theta, \vec{\tau}, \Lambda$) Finds a new solution that turns the current solution Θ's end-point towards direction $\vec{\tau}$ using CCD.

NonConversionDetected(aux, Λ) Checks whether or not the current aux_{Θ_ε} is converging towards a fixed value or behaving as a cyclic function, and therefore not converging.

NonConvOffsetTrick(aux, k) Shifts the current Target Orientation to a slightly different direction choosing link k and child as the expected offset solvers, by applying:

$$\Omega(\Theta, k, \delta) = \begin{cases} QAA(RotVQ(k_{RA}, \Theta_{k_Q}), \delta) & \textbf{if } |\Theta_{k_\theta} - k_{\text{Min}\theta}| > |\Theta_{k_\theta} - k_{\text{Max}\theta}| \\ QAA(RotVQ(k_{RA}, \Theta_{k_Q}), -\delta) & otherwise \end{cases}$$

$$\delta = \Lambda_{\text{Disturbance}\theta}$$

$$k = \Lambda_{\text{Sk}_{\text{Root}}}$$

$$aux_\tau = \Omega(aux_\Theta, k_{\text{Child}}, \delta) \cdot \Omega(aux_\Theta, k, \delta) \cdot aux_\tau$$

$$aux_{\text{TriedNonconvOffset}}) = True$$

SelectBestSolution(aux) Returns Θ such that:

$$\Theta = \begin{cases} aux_\Theta & \textbf{if } aux_{\Theta_\varepsilon} \leq aux_{\text{best}\Theta_\varepsilon} \\ aux_{\text{best}\Theta} & otherwise \end{cases}$$

A.2 Detailed Algorithms

We start by presenting a map of the algorithms in Figure A.1, which serves as a visual index to know where to find each of the pieces, and where they are invoked. Please refer back to Tables 6.2–6.4 in Section 6.3.5 and to the description of functions in Section A.1.1 while following or implementing these algorithmic descriptions.

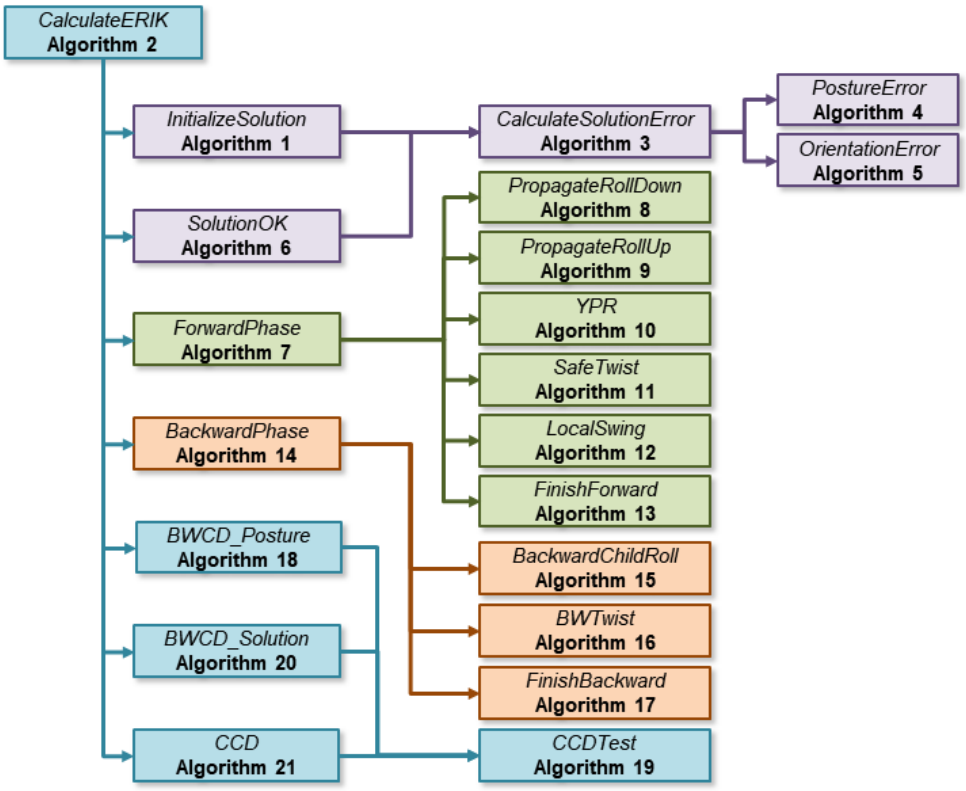


Figure A.1: A map of the algorithmic description of ERIK, to be used as a visual index throughout this section. Each arrow means that the algorithm from where it departs invokes the algorithm at which it arrives.

Algorithm 2: InitializeSolution

input : τ, Ψ, Λ // Orientation, Posture, Hyperparams
output : aux // container of execution variables

begin
  $aux \leftarrow \varnothing$
  $aux_{\tau} \leftarrow \tau$ // save a working copy of orientation
  if *IsTwister*$(\Lambda_{Sk_{EE}})$ **then**
    $aux_{\tau} \leftarrow \tau \cdot \text{RotVQ}(\hat{\boldsymbol{Y}}, \Psi_{EndPoint_{\theta}})$
  $aux_{\Psi} \leftarrow \Psi$ // save a working copy of posture
  $aux_{previous\Theta} \leftarrow \text{EmptySolution}(\Lambda_{\text{Sk}})$
  `CalculateSolutionError`$(aux_{previous\Theta}, \tau, \Psi, \Lambda_{\phi})$
  $aux_{best\Theta} \leftarrow aux_{\text{previous}\Theta}$
  $aux_{\Theta} \leftarrow aux_{\text{previous}\Theta}$
  return aux
end

Algorithm 3: CalculateERIK

input : Π, Λ // Parameters, Hyperparameters
output : Θ // Solution

begin
  $aux \leftarrow$ `InitializeSolution`$(\Pi_\tau, \Pi_\Psi, \Lambda)$
  $aux_\Psi \leftarrow$ `BWCD_Posture`$(aux_\Psi, aux_{\vec{\tau}}, \Lambda)$
  if $\Lambda_{\Xi_{AvoidJointEdges}}$ **then**
    AvoidPostureJointEdges$(aux_\Psi, \Lambda_{\text{Disturbance}\theta})$
  for $i \leftarrow 1$ **to** $\Lambda_{MaxERIKIterations}$ **do**
    for $k \leftarrow \Lambda_{Sk_{EE}}$ **to** $\Lambda_{Sk_{Root}}$ **do**
      $\tau \leftarrow aux_\tau$ **if** $k_{IsEndPoint}$ **else** $aux_{\Theta_{k_{Q_{Child}}}}$
      `ForwardPhase`$(k, \tau, aux_\Psi, aux_\Theta, \Lambda)$
    end
    for $k \leftarrow \Lambda_{Sk_{Root}}$ **to** $\Lambda_{Sk_{EE}}$ **do**
      `BackwardPhase`*(aux,* k, Π, Λ`)`
    end
    if `SolutionOK`$(\Pi, \Lambda,$ *aux*`)` **then**
      return aux_Θ
    $aux_\Theta \leftarrow$ `BWCD_Solution`$(aux_\Theta, aux_{\vec{\tau}}, \Lambda)$
    if `SolutionOK`$(\Pi, \Lambda,$ *aux*`)` **then**
      return aux_Θ
    if *NonConversionDetected*(aux, Λ) **then**
      if $\Lambda_{\Xi_{NonConvOffsetTrick}}$ ***and*** $\neg aux_{TriedNonconvOffset}$ **then**
        $aux \leftarrow$ NonConvOffsetTrick(aux, Λ)
        continue
      else if $\Lambda_{\Xi_{NonConvCCDTrick}}$ **then**
        $aux_\Theta \leftarrow$`CCD`$(aux_\Theta, aux_\tau, \Lambda)$
        if `SolutionOK`$(\Pi, \Lambda,$ *aux*`)` **then**
          return aux_Θ
        else
          $aux \leftarrow$ `CCD`*(EmptySolution*$(\Lambda_{Sk}), aux_\tau, \Lambda$`)` // Try from a new empty solution.
          if `SolutionOK`$(\Pi, \Lambda,$ *aux*`)` **then**
            return aux_Θ
        return *SelectBestSolution*(aux)
  end
  return *SelectBestSolution*(aux)
end

Algorithm 4: CalculateSolutionError

input : $\Theta, \tau, \Psi, \Lambda$ // Solution, Target Orientation, Target Posture, Hyperparameters
output : The solution's combined error Θ_ϵ

begin
  $\epsilon \leftarrow \Lambda_{\text{OrientationErrorWeight}} \cdot$ `OrientationError`$(\Theta_{EE}, \tau, \Lambda)$ + $\Lambda_{\text{PostureErrorWeight}} \cdot$ `PostureError`(Θ, Ψ, Λ)
  return ϵ
end

Algorithm 5: OrientationError

input : Φ, τ, Λ `// End-Effector Solution, Target Orientation, Hyperparameters`
output : The solution's orientation error $\epsilon_{\text{Orientation}}$

begin
  $\omega_\epsilon \leftarrow \frac{\min(|\tau - \Phi_{\text{EE}_\Omega}|, |\tau + \Phi_{\text{EE}_\Omega}|)}{\sqrt{2}}$ `// Absolute distance between quaternions`
  if $\Lambda_{\Xi_{SymmetricEndpoint}}$ **then**
    $\xi \leftarrow \text{QAA}(\text{RotVQ}(\hat{\boldsymbol{Y}}, \Phi_{\text{EE}_\Omega}), \pi) \cdot \Phi_{\text{EE}_\Omega}$ `// Rotate symmetrically about its rotation axis`
    $\xi_\epsilon \leftarrow \frac{\min(|\tau - \xi|, |\tau + \xi|)}{\sqrt{2}}$ `// Absolute distance between quaternions`
    $\omega_\epsilon \leftarrow \min(\omega_\epsilon, \xi_\epsilon)$
  return ω_ϵ
end

Algorithm 6: PostureError

input : Θ, Ψ, Λ `// Solution, Target Posture, Hyperparameters`
output : The solution's posture error measure $\epsilon_{\text{Posture}}$

begin
  $\epsilon \leftarrow 0, \quad i \leftarrow 0$
  $\vec{s} \leftarrow \vec{t} \leftarrow \Lambda_{\text{Sk}_{\text{Root}_\sigma}}$
  $a \leftarrow \Lambda_{\text{ErrorAggravation}}$
  for $k \leftarrow \Lambda_{Sk_{Root}}$ **to** $\Lambda_{Sk_{EE}}$ **do**
    if $\neg$*IsTwister*(k) **then**
      $\vec{u} \leftarrow \Theta_{k_{\text{Child}_\rho}} - \Theta_{k_\rho}$
      $\vec{v} \leftarrow \Psi_{k_{\text{Child}_{\vec{\rho}}}} - \Psi_{k_{\vec{\rho}}}$
      $d_{su} \leftarrow 1 - \frac{(1 + \vec{s} \cdot \vec{u})}{2}$
      $d_{tv} \leftarrow 1 - \frac{(1 + \vec{t} \cdot \vec{v})}{2}$
      $\epsilon \leftarrow \epsilon + a^i * |d_{tv} - d_{su}|$
      $i \leftarrow i + 1, \quad \vec{s} \leftarrow \vec{u}, \quad \vec{t} \leftarrow \vec{v}$
  end
  return $\frac{\epsilon}{\Lambda_{PostureNorm}}$
end

Algorithm 7: SolutionOK

input : Π, Λ, aux `// Parameters, Hyperparameters`
output : Is aux_Θ acceptable (Boolean)

begin
  `CalculateSolutionError(`$aux_\Theta, \Pi_\tau, \Pi_\Psi, \Lambda_\phi$`)`
  if $aux_{\Theta_\varepsilon} \leq aux_{best\Theta_\varepsilon}$ **then**
    $aux_{\text{best}\Theta} \leftarrow aux_\Theta$
  return $aux_{\Theta_\varepsilon} \leq \Lambda_{Threshold\varepsilon}$
end

Algorithm 8: ForwardPhase

input : $k, \tau, \Psi, \Theta, \Lambda$ `/* Joint, Target Orientation, Target Posture, (aux) Solution, HypParams */`

output : Intermediate solution Θ after forward phase

begin
  if $k_{IsEndPoint}$ **then** $\Theta_{k_\rho} \leftarrow \Psi_{k_{\vec{\rho}}}$
  if ***not*** k_{IsRoot} **then**
    $\vec{s} \leftarrow \text{RotVQ}(k_{\vec{\sigma}}, \Theta_{k_Q})$
    $\text{TBasis}(k, \text{VDiffAsQ}(\vec{s}, \Psi_{k_{\vec{\rho}}} - \Psi_{k_{\text{Parent}\,\vec{\rho}}}))$
    $\vec{r_p} \leftarrow \text{RotVQ}(\text{PitchRA}(k), \Psi_{k_Q})$
    $\vec{r_s} \leftarrow \text{RotVQ}(\text{PitchRA}(k), \Theta_{k_Q})$
    $\theta \leftarrow \text{VecAngle}(\text{proj}_{\vec{r_s}, \vec{\Theta_{k_d}}}, \text{proj}_{\vec{r_p}, \vec{\Theta_{k_d}}}, \vec{\Theta_{k_d}})$
    $\text{TBasisRoll}(k, \text{QAA}(\hat{\boldsymbol{Y}}, \theta))$
  if ***not*** *IsTwister*(k) **then**
    $\vec{s} \leftarrow \Psi_{k_{\text{Child}\,\vec{\rho}}} - \Psi_{k_{\vec{\rho}}}$
    $\vec{r_p} \leftarrow \vec{0}, \;\; n \leftarrow k, \;\; flipped \leftarrow False$
    while $\|\vec{r_p}\| \approx 0$ ***and*** $n \neq \varnothing$ **do**
      $m \leftarrow n$
      $\vec{p} \leftarrow n_{\vec{\sigma}}$ **if** n_{IsRoot} **else** $(\Psi_{n_{\vec{\rho}}} - \Psi_{n_{\text{Parent}\,\vec{\rho}}})$
      $\vec{r_p} \leftarrow \hat{p} \times \hat{s}$
      if n_{IsRoot} ***and not*** $flipped$ **then**
        $\vec{s} \leftarrow \Psi_{k_{\text{Child Child}\,\vec{\rho}}} - \Psi_{k_{\text{Child}\,\vec{\rho}}}$
        $flipped \leftarrow True, \;\; n \leftarrow n_{\text{Child}}$
      else $\vec{s} \leftarrow \vec{p}, \;\; n \leftarrow n_{\text{Parent}}$
    $\vec{s} \leftarrow \text{RotVQ}(\text{PitchRA}(m), I$ **if** m_{IsRoot} **else** $\Theta_{m_{\text{Parent}\,Q}})$
    $\vec{p} \leftarrow \text{RotVQ}(\text{PitchRA}(m), \Theta_{k_Q})$
    if $(\vec{p} \cdot \vec{r_p} < 0) \neq (\vec{r_p} \cdot \vec{s} < 0)$ **then** $\vec{r_p} \leftarrow -\vec{r_p}$
    $\theta \leftarrow \text{VecAngle}(\text{proj}_{\vec{s}, \vec{\Theta_{m_d}}}, \text{proj}_{\vec{r_p}, \vec{\Theta_{m_d}}}, \vec{\Theta_{m_d}})$
    `PropagateRollDown(`$QAA(\hat{\boldsymbol{Y}}, \theta), k, \Theta$`)`
  if ***not*** $k_{IsEndPoint}$ **then**
    $\vec{a} \leftarrow \vec{k_{RA}}$ **if** $\text{IsTwister}(k_{\text{Child}})$ **else** $\vec{k_{\text{Child}\,RA}}$
    $\vec{r} \leftarrow \text{RotVQ}(\vec{a}, \Theta_{k_Q})$
    $\vec{r_c} \leftarrow \text{RotVQ}(\vec{a}, \Theta_{k_{\text{Child}\,Q}})$
    if *IsTwister*(k_{Child}) ***and*** $\vec{r_c} \cdot \vec{r} < 0$ **then**
      `PropagateRollUp(`$QAA(\hat{\boldsymbol{Y}}, \pi), k, \Theta, True$`)`
    else if $\|proj_{\vec{r_c}} \vec{\Theta_{k_d}}\| \approx 0$ ***or*** $\|proj_{\vec{r}} \vec{\Theta_{k_d}}\| \approx 0$ **then**
      $\theta \leftarrow \text{VecAngle}(\text{proj}_{\vec{r_c}} \vec{\Theta_{k_d}}, \text{proj}_{\vec{r}} \vec{\Theta_{k_d}}, \vec{\Theta_{k_d}})$
      `PropagateRollUp(`$QAA(\hat{\boldsymbol{Y}}, \theta), k, \Theta, \theta \approx \pi$`)`
  $Q_y, Q_p, Q_r \leftarrow YPR(\text{QDiff}(\Theta_{k_Q}, \tau), k_{RA})$
  if $k_{IsEndPoint}$ ***and*** $Q_{r_\theta} > \frac{\pi}{2}$ **then**
    $Q_r \leftarrow \text{QAA}(\vec{Q_{r_v}}, \pi - Q_{r_\theta})$
  else if $k_{IsEndPoint}$ ***and*** $Q_{r\theta} < -\frac{\pi}{2}$ **then**
    $Q_r \leftarrow \text{QAA}(\vec{Q_{r_v}}, -\pi - Q_{r_\theta})$
  if *IsTwister*(k) **then**
    $\Theta_{k_\theta} \leftarrow$ `SafeTwist(`$k, Q_{y\theta} + Q_{r\theta}$`)`
  else
    $\Theta_{k_\theta} \leftarrow$ `LocalSwing(`$k, RotVQ(\vec{\tau}, \Theta_{k_{Q^*}})$`)`
    if $Q_r \neq \boldsymbol{I}$ **then** `PropagateRollDown(`Q_r, k, Θ`)`
  return `FinishForward(`k, Θ, Λ`)`
end

Algorithm 9: PropagateRollDown

input : Q, k, Θ `// Roll, Start link, Solution`
output : Solution Θ with Roll propagated down the kinematic chain.

begin
  repeat
    $k \leftarrow k_{\text{Parent}}$
    $\Theta_{k_Q} \leftarrow \Theta_{k_Q} \cdot Q$
  until k_{IsRoot}
  return Θ
end

Algorithm 10: PropagateRollUp

input : $Q, k, \Theta, bFlip = False$
output : Solution Θ with Roll propagated up the kinematic chain, with pitch angles flipped if $bFlip = True$.

begin
  while $\neg k_{IsEndPoint}$ **do**
    $\Theta_{k_{\text{Child}\,Q}} \leftarrow \Theta_{k_{\text{Child}\,Q}} \cdot Q$
    if $bFlipPitch$ ***and not*** *IsTwister*(k_{Child}) **then**
      $\Theta_{k_{\text{Child}\,\theta}} \leftarrow -\Theta_{k_{\text{Child}\,\theta}}$
    $k \leftarrow k_{\text{Child}}$
  end
  return Θ
end

Algorithm 11: YPR

input : $Q, \vec{R}$ `// Quat orientation, Rotation axis`
output : Q_y, Q_p, Q_r such that $Q = Q_y \cdot Q_p \cdot Q_r$

begin
  $\vec{u} \leftarrow \hat{\boldsymbol{Y}}, \;\; \vec{x} \leftarrow \vec{R}$
  $\vec{y_Q} \leftarrow \vec{u} \cdot Q_M$
  $\vec{x_Q} \leftarrow \vec{x} \cdot Q_M$
  $\vec{N} \leftarrow \vec{u} \times \vec{y_Q}$
  if $\|\vec{N}\| = 0$ **then**
    if $\vec{u} \cdot \vec{y_q} \approx 1$ ***or*** $\|\vec{y_Q}\| = 0$ **then**
      $Q_y \leftarrow Q_p \leftarrow \boldsymbol{I}$
      $Q_r \leftarrow \text{QAA}(\vec{u}, Q_\theta$ **if** $\vec{u} \cdot \vec{Q_v} \approx 1$ **else** $-Q_\theta)$
    else
      $Q_y \leftarrow \boldsymbol{I}$
      $Q_p \leftarrow \text{QAA}(\vec{x}, -\pi)$
      $Q_r \leftarrow \text{QAA}(\vec{u}, \text{VecAngle}(\vec{x}, \vec{x_Q}, \vec{y_Q}))$
  else
    if $\vec{N} \cdot \vec{x} < 0$ **then** $\vec{N} \leftarrow -vecN$
    $Q_y \leftarrow \text{QAA}(\vec{u}, \text{VecAngle}(\vec{x}, \hat{N}, \vec{u}))$
    $Q_p \leftarrow \text{QAA}(\vec{x}, \text{VecAngle}(\vec{u}, \vec{y_Q}, \hat{N}))$
    $Q_r \leftarrow \text{QAA}(\vec{u}, \text{VecAngle}(\hat{N}, \vec{x_Q}, \vec{y_Q}))$
  return $EPA(Q_y, \vec{u}), EPA(Q_p, \vec{x}), EPA(Q_r, \vec{u})$
end

Algorithm 12: SafeTwist

input : k, θ `// Joint, Angle`
output : Safe θ as cyclic local twist for ERIK's Joint Model

begin
  $\theta' \leftarrow$ SafeAngle$(k, \theta, bCycle \leftarrow True,)$
  $\beta \leftarrow (\theta - \theta') \bmod \pi$
  if $|\beta| > 0$ ***and*** $\neg k_{IsEndPoint}$ **then**
    if $\theta \leq k_{Min\theta}$ **then** $\theta' \leftarrow -k_{Min\theta} + \beta$
    else if $\theta \geq k_{Max\theta}$ **then** $\theta' \leftarrow -k_{Max\theta} + \beta$
  return θ'
end

Algorithm 13: LocalSwing

input : $k, \vec{\tau}$ `// Joint, Target direction (local)`
output : θ local swing angle using ERIK's LALUT

begin
  $\lambda \leftarrow$ TargetLatitude$(k, \vec{\tau})$
  return *SafeAngle*$(k,$ *LALUT*$(k, \lambda))$
end

Algorithm 14: FinishForward

input : k, Θ, Λ `// Link, Partial Solution, Hyperparams`
output : Solution Θ after Forward Phase

begin
  if $\Lambda_{\Xi_{AvoidEdges}}$ **then**
    $\Theta_{k_\theta} \leftarrow$ AvoidJointEdge$(k, \Lambda_{\text{Disturbance}\theta})$
  if $\neg k_{IsEndPoint}$ **then**
    $\Theta_{k_Q} \leftarrow$ EPA(QDiff$(\Theta_{k_\Omega}, \Theta_{k_{\text{Child}\,Q}} \cdot \Theta_{k_Q}), k_{RA},)$
    $\Theta_{k_\rho} \leftarrow \Theta_{k_{\text{Child}\,\rho}} -$ RotVQ$(k_{\vec{\sigma}}, \Theta_{k_{\text{Child}\,Q}})$
    $\Theta \leftarrow$ ApplyFK$(\Theta, k,)$
end

Algorithm 15: BackwardPhase

input : k, τ, Θ, Λ `/* Joint, Target Orientation, (aux) Solution, Hyperparameters */`
output : (Final) Solution Θ after backward phase

begin
  $\Theta \leftarrow$SetFrameFromParent(k, Θ,
) **if** *IsTwister*(k) **then**
    if $\neg k_{\textit{IsEndPoint}}$ **then**
      $\Theta \leftarrow$`BackwardChildRoll`(k, Θ, Λ)
    $\theta \leftarrow$`BWTwist`$(\tau_M, \Theta_{k_{\Omega_M}}) + \Theta_{k_\theta}$
    if $k_{\textit{IsEndPoint}}$ ***and*** $\Lambda_{\Xi_{\textit{SymmetricEndpoint}}}$ **then**
      $\theta' \leftarrow$`BWTwist`$(\tau_M \cdot QAA(\hat{\boldsymbol{Y}}, \pi), \Theta_{k_{\Omega_M}}) + \Theta_{k_\theta}$
      if $|\theta'| < |\theta|$ **then** $\theta \leftarrow \theta'$
    $\Theta_{k_\theta} \leftarrow \theta$
  else
    $Q \leftarrow \Theta_{k_{Q^*}} \cdot \tau$
    $\lambda \leftarrow$ TargetLatitude$(k, \vec{Q}_y)$
    $\theta \leftarrow$ SafeAngle$(k,$ LALUT$(k, \lambda))$
    if $\neg k_{\textit{IsEndPoint}}$ **then**
      $\theta' \leftarrow$ SafeAngle$(k,$ LALUT$(k, -\lambda))$
      $\vec{\sigma} \leftarrow$ RotVQ$(k_{\vec{\sigma}}, \Theta_{k_Q} \cdot$ QAA$(k_{RA}, \theta))$
      $\vec{\sigma'} \leftarrow$ RotVQ$(k_{\vec{\sigma}}, \Theta_{k_Q} \cdot$ QAA$(k_{RA}, \theta'))$
      $\vec{d} \leftarrow$ RotVQ$(k_{\vec{\sigma}}, \Theta_{k_{\text{Child}\,\Omega}})$
      if $\vec{\sigma'} \cdot \vec{d} > \vec{\sigma} \cdot \vec{d}$ **then**
        Swap(θ, θ')
        Swap(σ, σ')
      $\vec{d} \leftarrow$ RotVQ$(k_{\vec{\sigma}}, \tau)$
      if $\vec{\sigma'} \cdot \vec{d} > \vec{\sigma} \cdot \vec{d}$ **then**
        $\theta \leftarrow \theta'$
    $\Theta_{k_\theta} \leftarrow \theta$
    if $\neg k_{\textit{IsRoot}}$ ***and*** *IsTwister*$(k_{\textit{Parent}})$ **then**
      $\vec{r_t} \leftarrow$ RotVQ(k_{RA}, τ)
      $\vec{r_k} \leftarrow$ RotVQ(k_{RA}, Θ_{k_Q})
      $\vec{r_p} \leftarrow$ RotVQ$(k_{\text{Parent}\,RA}, \Theta_{k_{\text{Parent}\,Q}})$
      if $|\vec{r_t} \cdot \vec{r_p}| = 1$ ***or*** $|\vec{r_k} \cdot \vec{r_p}| = 1$ **then**
        $\vec{r_t} \leftarrow$ RotVQ(k_{OA}, τ)
        $\vec{r_k} \leftarrow$ RotVQ(k_{OA}, Θ_{k_Q})
      $\vec{p} \leftarrow \text{proj}_{\vec{r_t}} \vec{r_p}$
      if $\neg k_{\textit{IsEndPoint}}$ **then**
        $q \leftarrow$ QAA$(\Theta_{k_{\text{Child}\,d}},$ VecAngle$(\vec{r_t}, \vec{p}, \Theta_{k_{\text{Child}\,d}}))$
        $\vec{p} \leftarrow \text{proj}_{\text{RotVQ}(k_{RA}, q \cdot \tau)} \vec{r_p}$
      $\gamma \leftarrow$ VecAngle$(\text{proj}_{\vec{r_k}} \vec{r_p}, \vec{p}, \vec{r_p})$
      $\Theta_{k_{\text{Parent}\,\theta}} \leftarrow \gamma + \Theta_{k_{\text{Parent}\,\theta}}$
      $\Theta \leftarrow$SetFrameFromParent$(k, \Theta$,
)
  return `FinishBackward`(k, Θ)
end

Algorithm 16: BackwardChildRoll

input : k, Θ, Λ
output : Θ with $\Theta_{k_{\text{Child}\,RA}}$ aligned with plane $\perp$ to $\Theta_{k_{\text{Child}\,d}}$

begin
  if $\neg$*RotVQ*$(k_{ChildRA}, \Theta_{k_{ChildQ}}) \cdot \Theta_{k_{Qy}} = 0$ **then**
    $\vec{c_D} \leftarrow \Theta_{k_{\text{Child}\,d}} \quad c_B \leftarrow \Theta_{k_{\text{Child}\,Q_M}}$
    $axis \leftarrow$'y'
    if $c_{B_y} \cdot \vec{c_D} \approx 1$ ***or*** $c_{B_y} \cdot \vec{c_D} \approx -1$ ***or***
    $\Theta_{k_d} \cdot \vec{c_D} \approx 1$ ***or*** $\Theta_{k_d} \cdot \vec{c_D} \approx -1$ **then**
      if $c_{B_z} \cdot \vec{c_D} \approx 1$ ***or*** $c_{B_z} \cdot \vec{c_D} \approx -1$ ***or*** $c_{B_x} \cdot \Theta_{k_d} \approx 0$ **then** $axis \leftarrow$'x'
      else $axis \leftarrow$'z'
    $\vec{c_v} \leftarrow c_{B_{axis}}, \quad \vec{p_v} \leftarrow \Theta_{k_{Q_{axis}}}$
    $\vec{c_p} \leftarrow \text{proj}_{\vec{c_v}} \vec{c_D}, \quad \vec{p_p} \leftarrow \text{proj}_{\vec{p_v}} \vec{c_D}$
    $q \leftarrow \text{QAA}(\vec{c_D}, \text{VecAngle}(\vec{c_p}, \vec{p_p}, \vec{c_D})) \cdot \Theta_{k_{\text{Child}\,Q}}$
    $q' \leftarrow \text{QAA}(\vec{c_D}, \text{VecAngle}(\vec{c_p}, \vec{p_p}, -\vec{c_D})) \cdot \Theta_{k_{\text{Child}\,Q}}$
    if $q_{axis} \cdot \vec{p_v} < q'_{axis} \cdot \vec{p_v}$ **then**
      $\Theta_{k_{\text{Child}\,Q}} \leftarrow q'$
    else
      $\Theta_{k_{\text{Child}\,Q}} \leftarrow q$
  end
  return Θ
end

Algorithm 17: BWTwist

input : τ, Ω `//Target, Current Orientation`
output : Angle the joint should twist to achieve the given target τ based on its current orientation Ω

begin
  $d_{zy} \leftarrow \tau_z \cdot \Omega_y \quad d_{xy} \leftarrow \tau_x \cdot \Omega_y$
  if $d_{zy} \approx 1$ ***or*** $d_{zy} \approx -1$ ***or*** $|d_{xy}| < |d_{zy}|$ **then**
    return *VecAngle(Ω_x,proj$_{\tau_x}\Omega_y$,Ω_y)*
  else
    return *return VecAngle(Ω_z,proj$_{\tau_z}\Omega_y$,Ω_y)*
end

Algorithm 18: FinishBackward

input : k, Θ `// Link, Partial Solution`
output : Solution Θ after BackwardPhase

begin
  if $\neg k_{IsRoot}$ ***and*** *IsTwister*(k_{Parent}) **then**
    $\Theta \leftarrow$`FinishBackward(`k_{Parent}, Θ`)`
  $\Theta \leftarrow$ SetOriFromParent(k, Θ)
end

Algorithm 19: BWCD_Posture

input : $\Psi, \vec{\tau}, \Lambda$ `// Posture to solve, Target Direction, Hyperparameters`
output : Ψ result of the BWCD algorithm as a new posture.

begin
  for $i \leftarrow 1$ **to** $\Lambda_{MaxCCDIterations}$ **do**
    for $k \leftarrow \Lambda_{Sk_{Root}}$ **to** $\Lambda_{Sk_{EE}}$ **do**
      $\vec{pd} \leftarrow \text{RotVQ}(\Lambda_{\text{Sk}_{\text{EE}}\hat{\sigma}}, \Psi_{\text{Superpoint}_Q})$
      if `CCDTest(`$\vec{\tau}, \vec{pd}, \Lambda$`)` **then**
        return Ψ
      end
      $\vec{r} \leftarrow \text{RotVQ}(k_{RA}, \Psi_{k_Q})$
      $\vec{pdp} \leftarrow \text{proj}_{\vec{pd}}\vec{r}$
      $\vec{tdp} \leftarrow \text{proj}_{\vec{t}}\vec{r}$
      if $\|\vec{pdp}\| \neq 0$ ***and*** $\|\vec{tdp}\| \neq 0$ **then**
        $\alpha \leftarrow \text{VecAngle}(\vec{pdp}, \vec{tdp}, \vec{r})$
        if $\lfloor|\alpha|\rceil > 0$ **then**
          $q \leftarrow \text{EPA}(\text{QAA}(\vec{r}, \alpha), k_{RA})$
          $\vec{p} \leftarrow \Psi_{k_{\vec{p}}}$
          $\Psi_{k_\theta} \leftarrow \Psi_{k_\theta} + \alpha$
          for $j \leftarrow k$ **to** $\Lambda_{Sk_{EE}}$ **do**
            $\Psi_{j_{\vec{p}}} \leftarrow \vec{p}$
            $c \leftarrow j_{\text{Child}}$ **if not** $j_{\text{IsEndPoint}}$ **else** $\Psi_{\text{SuperPoint}}$
            $\Psi_{c_Q} \leftarrow q \cdot \Psi_{c_Q}$
            if $j_{IsEndPoint}$ **then**
              $\Psi_{\text{Superpoint}_{\vec{p}}} \leftarrow \vec{p} + \text{RotVQ}(j_{\vec{\sigma}}, \Psi_{c_Q})$
            else
              $\vec{p} \leftarrow \vec{p} + \text{RotVQ}(j_{\vec{\sigma}}, \Psi_{c_Q})$
          end
        end
      end
    end
    $\vec{pd} \leftarrow \text{RotVQ}(\Lambda_{\text{Sk}_{\text{EE}}\hat{\sigma}}, \Psi_{\text{Superpoint}_Q})$
    if `CCDTest(`$\vec{\tau}, \vec{pd}, \Lambda$`)` **then**
      return Ψ
    end
  end
  return Ψ
end

Algorithm 20: CCDTest

input : $\vec{t}, \vec{ee}, \Lambda$, returnError=False `// Joint, Target, EndEffector, Hyperparameters, (optional)`
output : Current $\vec{ee}$ is an acceptable solution for a CCD-based algorithm.

begin
  $\epsilon \leftarrow \lfloor -\frac{\vec{t} \cdot \vec{ee} - 1}{2} \rceil$
  if *returnError* **then return** $\epsilon \leq \Lambda_{CCDPrecision}, \epsilon$
  return $\epsilon \leq \Lambda_{CCDPrecision}$
end

Algorithm 21: BWCD_Solution

input : Θ, τ, Λ `// Solution to solve, Target Direction, Hyperparameters`
output : Θ' result of the BWCD algorithm as a new solution.

begin
  for $i \leftarrow 1$ **to** $\Lambda_{MaxCCDIterations}$ **do**
    $\vec{eod} \leftarrow \Theta_{\mathrm{EE}_d}$
    for $k \leftarrow \Lambda_{Sk_{Root}}$ **to** $\Lambda_{Sk_{EE}}$ **do**
      if `CCDTest(`$\vec{\tau}, \vec{eod}, \Lambda$`)` **then**
        $\Theta_{\mathrm{EE}_\theta} \leftarrow \Theta_{\mathrm{EE}_\theta}$ + `BWTwist(`$\tau_M, \Theta_{EE_{\Omega_M}}$`)`
        return Θ
      end
      $\vec{r} \leftarrow \mathrm{RotVQ}(k_{RA}, \Theta_{k_\Omega})$
      $\vec{top} \leftarrow \mathrm{proj}_{\vec{\tau}}\vec{r}$
      $\vec{eop} \leftarrow \mathrm{proj}_{\vec{eod}}\vec{r}$
      if $\|\vec{top}\| \neq 0$ ***and*** $\|\vec{eop}\| \neq 0$ **then**
        $\Theta_{k_\theta} \leftarrow \Theta_{k_\theta} + \mathrm{VecAngle}(\vec{eop}, \vec{top}, \vec{r})$
        $\mathrm{ApplyFK}(\Theta, k)$
        $\vec{eod} \leftarrow \Theta_{\mathrm{EE}_d}$
      end
    end
    if `CCDTest(`$\vec{\tau}, \vec{eod}, \Lambda$`)` **then**
      $\Theta_{\mathrm{EE}_\theta} \leftarrow \Theta_{\mathrm{EE}_\theta}$ + `BWTwist(`$\tau_M, \Theta_{EE_{\Omega_M}}$`)`
      return Θ
    end
  end
  $\Theta_{\mathrm{EE}_\theta} \leftarrow \Theta_{\mathrm{EE}_\theta}$ + `BWTwist(`$\tau_M, \Theta_{EE_{\Omega_M}}$`)`
  return Θ
end

Algorithm 22: CCD

input : Θ, τ, Λ `// Solution to solve, Target Orientation, Hyperparameters`
output : Θ result of the CCD algorithm as a new solution.

begin
  $\epsilon \leftarrow 10000, \text{last}_\epsilon \leftarrow 0$
  $\vec{td} \leftarrow \vec{\tau}$
  $\vec{ed} \leftarrow \Theta_{\text{EE}_{\vec{d}}}$
  for $i \leftarrow 1$ **to** $\Lambda_{MaxCCDIterations}$ **do**
    $\text{last}_\epsilon \leftarrow \epsilon$
    for $k \leftarrow \Lambda_{Sk_{EE}}$ **to** $\Lambda_{Sk_{Root}}$ **do**
      $\text{ok}, \epsilon \leftarrow$ `CCDTest(`$\vec{td}, \vec{ed}, \Lambda, returnError = True$`)`
      if *ok* **then**
        $\Theta_{\text{EE}_\theta} \leftarrow \Theta_{\text{EE}_\theta} +$ `BWTwist(`$\tau_M, \Theta_{EE_{\Omega_M}}$`)`
        return Θ
      end
      $r \leftarrow \text{RotVQ}(k_{RA}, \Theta_{k_Q})$
      $\vec{tdp} \leftarrow \text{proj}_{\vec{td}} r$
      $\vec{edp} \leftarrow \text{proj}_{\vec{ed}} r$
      if $\|\vec{tdp}\| \neq 0$ ***and*** $\|\vec{edp}\| \neq 0$ **then**
        $\Theta_{k_\theta} \leftarrow \Theta_{k_\theta} + \text{VecAngle}(\vec{edp}, \vec{tdp}, r)$
        if $\Lambda_{\Xi_{AvoidJointEdges}}$ **then**
          $\text{AvoidPostureJointEdges}(\Theta, \Lambda_{\text{Disturbance}\theta})$
        $\text{ApplyFK}(k,)$
        $\vec{ed} \leftarrow \Theta_{\text{EE}_{\vec{d}}}$
      end
    end
    $ok, \epsilon \leftarrow$ `CCDTest(`$\vec{td}, \vec{ed}, \Lambda, returnError = True$`)`
    if *ok* **then**
      $\Theta_{\text{EE}_\theta} \leftarrow \Theta_{\text{EE}_\theta} +$ `BWTwist(`$\tau_M, \Theta_{EE_{\Omega_M}}$`)`
      return Θ
    end
    if $|\epsilon - last_\epsilon| \leq \Lambda_{Precision}$ **then**
      $\Theta_{\text{EE}_\theta} \leftarrow \Theta_{\text{EE}_\theta} +$ `BWTwist(`$\tau_M, \Theta_{EE_{\Omega_M}}$`)`
      return Θ
    end
  end
  $\Theta_{\text{EE}_\theta} \leftarrow \Theta_{\text{EE}_\theta} +$ `BWTwist(`$\tau_M, \Theta_{EE_{\Omega_M}}$`)`
  return Θ
end

Appendix B

User-Study Questionnaires

In the following sections we present the integral questionnaires as they were administered in both the Ahoy and the AvantSatie studies. Both questionnaires are presented in the original Portuguese language.

B.1 Ahoy Study Questionnaire

Nas páginas seguintes encontrarás alguns questionários, todos eles diferentes.

Lê as instruções de cada questionário **com atenção** e chama o investigador se tiveres alguma questão.

Todos os questionários são **anónimos e confidenciais.**

Sê honesto nas tuas respostas, **não existem respostas certas ou erradas** para as questões.

Instruções:

- As seguintes questões referem-se aos **momentos do jogo em que o robot fez a mímica das palavras que tiveste de adivinhar.**
- Faz um **círculo à volta de um número de 1-6** que melhor representa a tua opinião.
- IMPORTANTE: Não faças um círculo à volta de "Discordo completamente" ou "Concordo completamente", mas apenas de um dos números.

O robot foi bom a fazer a mímica das palavras.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

Consegui pensar em palavras que poderiam estar a ser representadas pela mímica do robot.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

Os movimentos do robot durante as mímicas foram naturais.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O robot pareceu possuir vida própria.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O robot fez as mímicas de maneira a ser fácil para mim ver e entender o que ele estava a fazer.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O robot parecia entender o conceito das palavras que estava a mimicar.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O robot pensou em cada uma das palavras antes de fazer a mímica.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O robot lembrava os personagens animados que conheço do cinema.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

::

Instruções:

- As seguintes questões referem-se aos **momentos do jogo em que estavas a tentar adivinhar a palavra correcta, ANTES** de o robot te indicar se conseguiste ou não.
- Faz um **círculo à volta de um número de 1-6** que melhor representa a tua opinião.
- IMPORTANTE: Não faças um círculo à volta de "Discordo completamente" ou "Concordo completamente", mas apenas de um dos números.

O robot deu-me dicas à medida que eu tentava adivinhar a palavra correcta.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

Consegui perceber, através das dicas do robot, se estava perto ou longe da resposta certa.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

As dicas do robot pareciam ser coerentes com as minha tentativas.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

As dicas que o robot me ia dando ajudaram-me a adivinhar a palavra correcta.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O movimento do robot enquanto eu tentava adivinhar era suave e natural.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O robot movimentou-se em sintonia comigo enquanto eu tentava adivinhar a palavra certa.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O robot estava entusiasmado com as minhas tentativas de adivinhar a palavra correcta.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O robot queria que eu conseguisse adivinhar a palavra certa.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

::

Instruções:

- As seguintes questões referem-se **à interacção, no geral, que tiveste com o robot, durante o jogo.**
- Faz um **círculo à volta de um número de 1-6** que melhor representa a tua opinião.
- IMPORTANTE: Não faças um círculo à volta de "Discordo completamente" ou "Concordo completamente", mas apenas de um dos números.

Dei pela presença do robot.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

Os pensamentos do robot foram claros para mim.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

Foi difícil para o robot compreender-me.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O robot atraiu a minha atenção.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

Os meus pensamentos foram claros para o robot.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O robot deu pela minha presença.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

Foi fácil para o robot compreender-me.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

A presença do robot foi óbvia para mim.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

Eu atraí a atenção do robot.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

Foi fácil compreender o robot.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

A minha presença foi óbvia para o robot.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

Foi difícil compreender o robot.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

Achei que o robot foi capaz de se adaptar ao que eu precisei.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

Achei que o robot fez apenas o que eu precisei, em cada momento.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O robot ajudou-me quando eu achei necessário.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

::

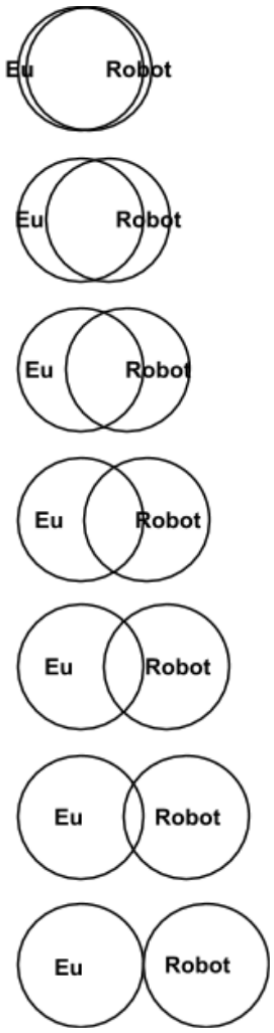
Instruções:

- **Selecciona o número** que achas que representa melhor o robot para cada uma das linhas abaixo.

Morto	1	2	3	4	5	6	Vivo
Estagnado	1	2	3	4	5	6	Vivacidade
Mecânico	1	2	3	4	5	6	Orgânico
Artificial	1	2	3	4	5	6	Naturale
Inerte	1	2	3	4	5	6	Interatico
Apático	1	2	3	4	5	6	Responsivo
Incompetente	1	2	3	4	5	6	Competente
Ignorante	1	2	3	4	5	6	Conhecedor
Irresponsável	1	2	3	4	5	6	Responsável
Pouco Inteligente	1	2	3	4	5	6	Inteligente
Insensato	1	2	3	4	5	6	Sensato
Antipático	1	2	3	4	5	6	Simpático
Hostil	1	2	3	4	5	6	Amigável
Cruél	1	2	3	4	5	6	Amável
Desagravável	1	2	3	4	5	6	Agradável
Horroroso	1	2	3	4	5	6	Atrativo

Instruções:

- Tendo em conta a interacção com o robot, indica **quão perto te sentiste em relação ao robot durante o jogo.**
- Faz uma cruz na resposta que melhor se adequa ao que sentiste.



Por último, completa com os teus dados demográficos (não adiciones o teu nome):

Idade:

Profissão:

Curso (se aplicável):

Ano de curso:

Familiaridade com robots (selecciona uma das opções abaixo)

Nunca tinha interagido com um robot ___

Já tinha interagido com um robot antes desta experiência ___ Quantas vezes? ___

Costumas jogar à mímica?

Sim ___ Não ___

Se sim,

Com que frequência?

Quando foi a última vez que jogaste?

(aproximadamente o mês e o ano)

Obrigado por teres feito parte deste estudo.

B.2 AvantSatie Study Questionnaire

Nas páginas seguintes encontrarás alguns questionários, todos eles diferentes.

Lê as instruções de cada questionário **com atenção** e chama o investigador se tiveres alguma questão.

Todos os questionários são **anónimos e confidenciais.**

Sê honesto nas tuas respostas, **não existem respostas certas ou erradas** para as questões.

Instruções:

- As seguintes questões referem-se aos momentos do jogo em que tentavas adivinhar as notas correctas, e não enquanto tu ou o robot estavam a repetir as músicas.
- Faz um **círculo à volta de um número de 1-6** que melhor representa a tua opinião.
- IMPORTANTE: Não faças um círculo à volta de "Discordo completamente" ou "Concordo completamente", mas apenas de **um** dos números.

O robot sabia onde estava cada uma das notas de cada música.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

Não teria conseguido entender o jogo sem o robô.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

Consegui descobrir as notas correctas graças ao robô.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

O robô queria que eu acertasse nas notas.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

Tive de olhar para o ecrã para saber o que tinha acontecido a cada momento.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

Não teria conseguido descobrir as músicas sem a ajuda do robô.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

O robô entendeu sempre bem a nota que eu tinha tocado.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

O robô seguia-me com o olhar enquanto eu tentava adivinhar as notas certas.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

O robô sabia onde estava cada uma das notas de cada música.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

As dicas que o robô me ia dando ajudaram-me a adivinhar as notas correctas.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O movimento do robô era suave e natural.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O robô pareceu possuir vida própria.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

Não entendia o jogo sem olhar para o ecrã.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

As dicas do robô eram coerentes com as minha tentativas de encontrar cada nota certa.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

Consegui entender, através do robô, quando tinha adivinhado cada nota correcta.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O robô lembrava os personagens animados que conheço do cinema.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O robô queria que eu conseguisse descobrir as músicas na totalidade.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O robô pensou em ajudar-me.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O robô conhecia bem cada uma das músicas.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

O ecrã de jogo continha a informação que eu precisava para entender o jogo.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

O robô estava entusiasmado com as minhas tentativas de adivinhar as notas correctas.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

O robô tinha as músicas todas na cabeça.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

O robô deu-me dicas à medida que eu tentava adivinhar cada nota certa.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

Consegui perceber, através das dicas do robô, se estava perto ou longe de cada nota correcta.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

O robô olhou para as teclas certas enquanto repetia cada nota.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

Tive de seguir o ecrã de jogo para entender o que fazer.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

O movimento do robô acompanhou o meu ritmo.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

::

Instruções:

- As seguintes questões referem-se **à interacção, no geral, que tiveste com o robot, durante o jogo.**
- Faz um **círculo à volta de um número de 1-6** que melhor representa a tua opinião.
- IMPORTANTE: Não faças um círculo à volta de "Discordo completamente" ou "Concordo completamente", mas apenas de **um** dos números.

Dei pela presença do robot.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

Os pensamentos do robot foram claros para mim.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

Foi difícil para o robot compreender-me.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O robot atraiu a minha atenção.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

Os meus pensamentos foram claros para o robot.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

O robot deu pela minha presença.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

Foi fácil para o robot compreender-me.

Discordo completamente	1	2	3	4	5	6	Concordo completamente

A presença do robot foi óbvia para mim.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

Eu atraí a atenção do robot.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

Foi fácil compreender o robot.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

A minha presença foi óbvia para o robot.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

Foi difícil compreender o robot.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

Achei que o robot foi capaz de se adaptar ao que eu precisei.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

Achei que o robot fez apenas o que eu precisei, em cada momento.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

O robot ajudou-me quando eu achei necessário.

Discordo completamente 1 2 3 4 5 6 Concordo completamente

::

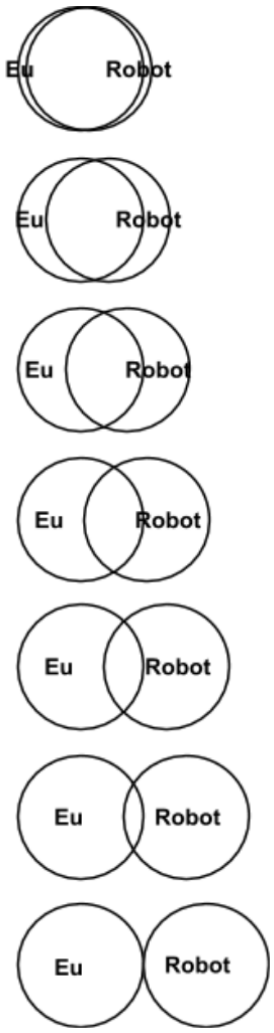
Instruções:

- Utilizando a escala abaixo, quanto associas cada palavra ao robot com que acabaste de interagir?
- Para cada palavra, faz um **círculo à volta de um número de 1-9** que melhor representa a tua opinião, em que **1** representa **Nada a ver**, e **9** representa **Tudo a ver**.

Nada a ver	1	2	3	4	5	6	7	8	9	Tudo a ver
Compassivo	1	2	3	4	5	6	7	8	9	
Embaraçoso	1	2	3	4	5	6	7	8	9	
Capaz	1	2	3	4	5	6	7	8	9	
Feliz	1	2	3	4	5	6	7	8	9	
Horrível	1	2	3	4	5	6	7	8	9	
Perigoso	1	2	3	4	5	6	7	8	9	
Confiável	1	2	3	4	5	6	7	8	9	
Assustador	1	2	3	4	5	6	7	8	9	
Competente	1	2	3	4	5	6	7	8	9	
Sensível	1	2	3	4	5	6	7	8	9	
Responsivo	1	2	3	4	5	6	7	8	9	
Agressivo	1	2	3	4	5	6	7	8	9	
Emocional	1	2	3	4	5	6	7	8	9	
Interativo	1	2	3	4	5	6	7	8	9	
Estranho	1	2	3	4	5	6	7	8	9	
Conhecedor	1	2	3	4	5	6	7	8	9	
Orgânico	1	2	3	4	5	6	7	8	9	
Social	1	2	3	4	5	6	7	8	9	

Instruções:

- Tendo em conta a interacção com o robot, indica **quão perto te sentiste em relação ao robot durante o jogo.**
- Faz uma cruz na resposta que melhor se adequa ao que sentiste.



Por último, completa com os teus dados demográficos (não adiciones o teu nome):

Género: ☐ Feminino ☐ Masculino ☐ Outro/Não quero dizer

Idade:

Profissão:

Curso (se aplicável):

Ano de curso:

Familiaridade com robots (selecciona uma das opções abaixo)

Nunca tinha interagido com um robot ☐

Já tinha interagido com um robot antes desta experiência ☐ Quantas vezes? ___

Em relação ao **teu** conhecimento e experiência musical,

- **Selecciona o número** que achas que te representa melhor para cada uma das linhas abaixo.

Não sei tocar nenhum instrumento musical	1	2	3	4	5	6	Consigo dar aulas de algum instrumento musical
Não sei ler pautas musicais	1	2	3	4	5	6	Leio fluentemente pautas musicais

Obrigado por teres feito parte deste estudo.