The practice of animation in robotics

Tiago Ribeiro and Ana Paiva

Abstract Robot animation is a new form of character animation that extends the traditional process by allowing the animated motion to become more interactive and adaptable during interaction with users in real-world settings. This paper reviews how this new type of character animation has evolved and been shaped from character animation principles and practices. We outline some new paradigms that aim at allowing character animators to become robot animators, and to properly take part in the development of social robots. In particular, we describe the 12 principles of robot animation, which describes general concepts that both animators and robot developers should consider in order to properly understand each other. We conclude with a description of some types of tools that can be used by animators, while taking a part in the development process of social robot applications, and how they fit into the rest of the system.

1 Introduction

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 [4].

Tiago Ribeiro
INESC-ID &
Instituto Superior Técnico, University of Lisbon, e-mail: me@tiagoribeiro.pt

Ana Paiva
INESC-ID &
Instituto Superior Técnico, University of Lisbon, e-mail: ana.paiva@inesc-id.pt

Since then these hand-drawn animated characters have been evolving and taking in many different forms and audiences. During the last thirty years, animated characters have become mainly computer-animated, and are being produced by many major animation studios such as Pixar, Walt Disney Animation Studios, Dreamworks or Blue Sky Studios.

Today we see robots becoming a new form of animated characters. However, this time the characters are jumping out of the big screens, powered by artificial intelligence (AI), and are becoming more interactive, and part of people's daily life. They are being developed in order to be used in social applications, in fields such as education, entertainment or assisted living. Given the technological background required for the creation of such characters, they are being developed by roboticists, software engineers and AI scientists, instead of by artists.

While this has been a necessary stage, we believe it is now time for robots and animated characters to reunite, by allowing artists and robot developers to work together, side by side, on the development of such characters. Animation artists have already been providing a contributing voice in the development of expressive, emotional and design traits of robots. However they typically get little to no access to the development of the actual interactive and intelligent behaviors that are performed with humans.

The goal of our work is to establish a solid bridge between these two worlds, which are intrinsically connected, but have been evolving separately, based on different perspectives, fundamental competencies, and end-goals. Such a connection will allow animators to take a new role as artists that are fully part of, and not just accessory, to the development of social robotic products. The same happened upon the emerging of computer animated cartoons and in particular, of 3D animated characters. 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 [18]. Not only were those principles transferred to 3D animated characters, but new tools and methodologies were also created to support the creative and development processes. Establishing robot animation as the new character animation medium will therefore require not only new theories, but also the integration of the technology with new tools and practices.

In this chapter we start by reviewing some character animation theories, along with existing proposals of how to adapt and use them with robots, and present some cases in which the animation process was considered and integrated in the development of socially interactive robots. We then outline a list of principles of animation for robots based on the current state of the art, and on our own previous and diverse experience. These principles are intended to provide thoughts on some general concepts that both animators and robot developers should consider, in order to properly understand each other, and to engage in successful collaborations. We complete the chapter by providing an overview on how the creative and technical tools and workflows may converge into an integrated robot animation pipeline, in

which both artists and engineers are able to work together from initial development to the finished product.

2 Character Animation

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 [30], the last two of Disney's Nine Old Men[1]. We summarize these principles further on, in section 2.1

However since The Golden Age of American Animation, Warner Bros. and MGM animators also definitely marked their position as masters of animated cartoons. These animators took exaggeration to another level, by giving special focus on physical exaggeration, in which we can actually identify common sub-types 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.

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 [7].

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. The EMYS robotic head is an example of how a 'Tex Avery' eyes-popping mechanism can be incorporated into a robot [23].

A common trait in character animation is that each character is made to be very unique and well adapted to its role. 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 [3]. 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.

Unfortunately, except for Disney-based ones, the practice of these animators is not very well documented. As they were generally jumping around from one studio to another, each animator may have followed different guidelines along his career, there are no compiled guidelines to describe their creative process. However, by viewing

[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).

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.

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' [8]. Henson's puppets (Figure 1) are generally very simple in movement. Most of them can only open and close their mouth, and wave their arms and body. It was impossible to actually convey human-like expressions with them, and that was not needed. 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.

Fig. 1 The Muppet Show's Kermit the Frog.

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.

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

In the realm of 3D animated characters, Walt Disney Animation Studios, Pixar, Dreamworks and Blue Sky have become established as the major studios. These studios have been leading teams of some of the best artists in the world to create critically acclaimed animation films such as 'Toy Story', 'Monsters, Inc.', 'Tangled', 'How to Train Your Dragon', 'Ice Age' and many more. In particular, one of Pixar's most popular films is WALL-E, which features a highly expressive animated robot as the main character. Once again John Lasseter, who is a cornerstone in the shift from hand-drawn to 3D animation, has took the time to present some tips for traditional animators to learn how to adapt and animate characters in the 3D world [19].

2.1 Disney's Twelve Principles of Animation

For reference, we present a small summary of the original Twelve Principles of Animation defined in 'The Illusion of Life' [30].

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 natural 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 connected by smooth in-betweenings (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, unless it is clearly intended not to. Slow out can be confused with follow-through; however, follow-through extends the action, while the slow-out finishes it smoothly.

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 typical robotic movement would make the head rotate only 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 life-likeness. 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 some 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 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 [26].

Figure 2 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 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.

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. Figure 3 shows the animation curve of the rotation of the pivot of a pendulum that is dropped from a

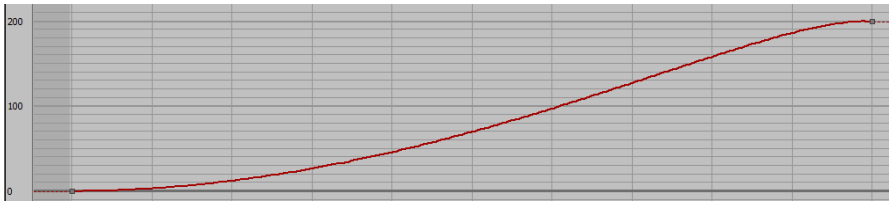

Fig. 2 The animation curve of the translation of a drag car accelerating until it reaches a top speed, and then decelerating until it halts. The vertical axis represents distance in generic units.

height of 40 degrees. It then balances several times while losing momentum due to friction and air resistance, until it stops.

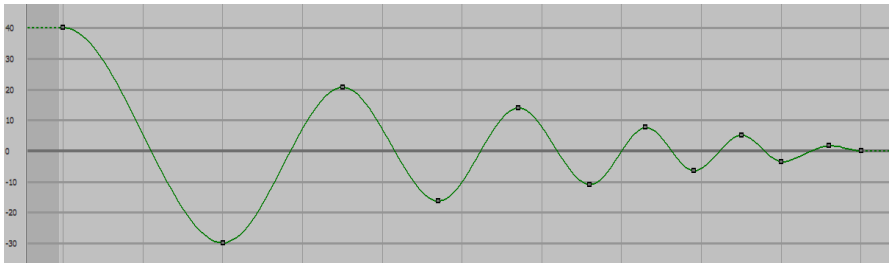

Fig. 3 The animation curve of the rotation of a pendulum that is dropped from 40 degrees and balances until it stops. The vertical axis represents the angle in degrees.

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

By looking at each key-frame, we see that the angle goes from 40 degrees to -30, then to about 20, 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 each spline between two key-frames is steeper at this point, than closer to the key-frames. As the pendulum loses energy and balances less, the steepness becomes lower, which reflects a lower speed, until it comes to a stop.

Animation curves therefore stand as a very important tool for representing, analyzing and adjusting animations. They can also be computationally processed just like a signal, in order to warp the animation and create animation effects. More importantly, the animation curves represent a concept that both animators and engineers can understand, and can use it to connect their thoughts, requirements and obstacles. Furthermore, they provide a technical interface that animators can use, and that can faithfully and mathematically model motion for robots.

3 Related Work

Various authors have previously worked 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' [6]. 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 into account when animating robots, which do not accurately follow the ones from Disney [32]. 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 closely related to 'Anticipation' and 'Engagement' refers to 'Staging'. Furthermore it is discussable whether or not Eyes must be part of robots at all.

Mead and Mataric also addressed the principles of Staging, Exaggeration, Anticipation and Secondary Action to improve the understanding of a robot's intentions by autistic children [20]. For exaggeration, they were inspired in a process used for the generation of caricatures, by exaggerating the difference from the mean.

Hoffman & Ju have presented some guidelines and techniques, especially based on previous experiences, about designing robots with their expressive movement in mind [15]. They provide 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 just as an after-step.

3.1 Use of animation concepts and techniques in robots

In 2003, Breazeal and colleagues presented the Interactive Theatre [5]. 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 system to dynamically change the appearance of its motion depending on events captured by a vision system [13].

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 [16]. It is mounted on a workbench and controlled through a hybrid control system that allows it to be used for live puppeteering, in order to allow the robot to be expressive while also being responsive. In AUR, the motion is controlled by extensively trained puppeteers, and 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 overridden

by an inverse kinematics solution using CCD [31]. 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.

Shimon is a gesture based musical improvisation robot created by Hoffman & Weinberg that plays a real marimba [17]. Its behaviour is a mix between its functionality as a musician, for which it plays the instrument in tune and rhythm, and being part of a band, for which it performs expressive behaviour by gazing towards its band mates during the performance.

Travis is a robotic music listening companion also created by Hoffman, that acts as an interactive expressive music dock for smart phones [14]. The system allows a user to dock a smart-phone 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.

More recently, Suguitan & Hoffman have created Blossom, a flexible, hand-crafted social robot that abides several principles of animation such as squash and stretch, slow in/out and follow-through animation [28]. The robot was built using an innovative compliant tensile structure that allows it to be flexible even in the inside. The exterior has a soft woven cover that can deform and shift freely, thus accentuating its organic movement.

Various interactive social robots have been created at MIT's MediaLab that build on animation concepts and techniques [13]. In particular the AIDA[2] 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.

Takayama, Dooley & Ju have explored the use of animation principles using the PR-2 robot[3]. This is a large mobile robot with two arms, that can navigate in a human environment. The authors focused on the use of Anticipation, Engagement, Confidence and Timing to enhance the readability of a robot's actions [29]. Once again, the authors refer to 'Engagement', when in fact they follow the 'Staging' principle. 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 stage. Doug Dooley, a professional animator from Pixar Animation Studios, 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 in the core of character animation, and in the portrayal of the illusion of life. In this work, the authors also argue for the need of both functional and expressive behaviors, i.e., that some of the robot's behaviours would be related with accomplishing a given task (e.g. picking up an object; opening a door), and that another part would concern its expressiveness in order to convey thought and emotion.

---

[2] `http://robotic.media.mit.edu/portfolio/aida` (accessed March 02, 2019)

[3] `http://www.willowgarage.com/pages/pr2/overview` (accessed March 02, 2019)

Gielniak et al. have successfully 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 [12]. The same authors have also presented techniques to simulate the principles of Secondary Motion [10] and of Anticipation [11] in robot motion.

Walt[4] is a social collaborative robot that that helps factory workers assemble cars. Walt 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.

Several works by Ribeiro & Paiva aim at creating software technology and tools that allow animators and robot developers to work together. In particular, they have created Nutty Tracks, an animation engine and pipeline, aimed at providing an expressive bridge between an application-specific artificial intelligence, the perception of user and environment, and a physical, animated embodiment [22]. It is able to combine and blend multi-modal expressions such as gazing towards users, while performing pre-designed animations, or overlaying expressive postures over the idle- and gazing- behaviour of a robot[5]. Furthermore, Nutty Tracks can also be used or adapted as a plug-in in animation software such as Autodesk 3ds Max[6] and Maya[7], SideFX Houdini[8] or even the open-source Blender software[9]. The composing of animation programs in the Nutty Tracks GUI follows a box-flow type of interface greatly inspired by other programming tools commonly used by artists, such as the Unreal Engine[10], Pure Data[11] or Houdini[8]. Figure 4 shows the Nutty Tracks GUI. Animation Controllers are connected into a chain of execution that generates and composes animation either procedurally or using animations and postures that were pre-designed using other animation software. The convergence between animation tools and a robot animation engine allows researchers to explore the use of animation principles in such autonomous interactions with humans by focusing, however, on the behaviour selection and management mechanisms, and on pre-designing particular animations that were solely selected and played back on the robots. The development pipeline for Nutty Tracks has also been briefly exemplified with the Keepon robot[12] [25].

More recently the same authors have created ERIK, a new inverse kinematics technique that allows an articulated robot with multiple DoFs (such as a manipulator), to exhibit an expressive posture while aiming towards any given direction [24]. The technique is demonstrated using the custom built, and DIY[13] inspired low-fidelity

[4] `http://robovision.be/offer/#airobots` (accessed March 02, 2019)

[5] `http://vimeo.com/67197221` (accessed March 02, 2019)

[6] `https://www.autodesk.com/products/3ds-max/overview` (accessed March 02, 2019)

[7] `https://www.autodesk.com/products/maya/overview` (accessed March 02, 2019)

[8] `https://www.sidefx.com/products/houdini` (accessed March 02, 2019)

[9] `https://www.blender.org` (accessed March 02, 2019)

[10] `http://www.unrealengine.com` (accessed March 02, 2019)

[11] `http://puredata.info` (accessed March 02, 2019)

[12] `https://vimeo.com/155593476` (accessed March 02, 2019)

[13] 'Do-it-yourself'

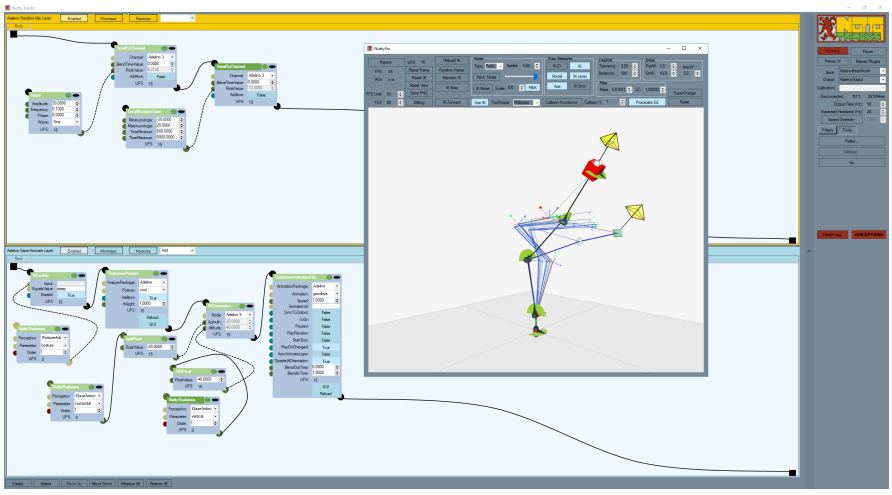

Fig. 4 The Nutty Tracks standalone GUI, used for composing animation programs, and to execute them in both a virtual window (for diagnostics) and on the real robot.

craft robot Adelino[14]. The purpose of ERIK is to allow complex robots to interact with humans while exhibiting artistically-crafted expressions. By allowing simple, artist-designed expressive postures to be warped in real-time and turned to face any direction, while maintaining continuous movement that complies with the robot's mechanical constraints, the technique brings robot animation a step closer to typical artist-centered character animation pipelines.

4 Robot Animation

Before we move on to define our principles of animation for robots, 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 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 AI, and that

[14] `https://vimeo.com/232300140` (accessed March 02, 2019)

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.

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 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 (that would fall into the field of animatronics). Instead it is about creating techniques, systems and interfaces that allow animation artists to design, specify and program ***how*** the motion will be generated, shaped and composed throughout an interaction, based on the behaviour descriptions that are computed by the AI.

One such common and basic behaviour we take as 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). Therefore in the context of robot animation, such gazing behaviour should consider not only an orientation but also the expressivity portrayed through the behaviour in an interactive manner. 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 e.g. the manipulator's endpoint to take on the expressive role of being the character's head, i.e. taking inspiration on an animated snake.

4.1 Principles of Robot Animation

A general list of Principles of Robot Animation should also address principles related to human-robot interaction. In this list however, we refrain from deepening such topic that is already subject of intensive study [21, 9, 2, 1]. 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[15], 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 [15, 28], we can also create a squash and stretch effect by using poses and body movement.

In Figure 5 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 6 presents a snapshot from the video[15] illustrating how this principle looks like on the NAO robot.

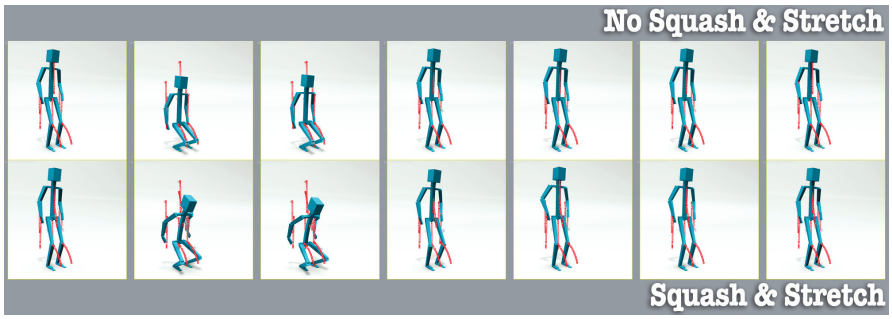

Fig. 5 An animation sequence denoting the principle of Squash & Stretch. The red marks represent the trajectory of the most relevant joints.

[15] https://vimeo.com/49122495 (accessed March 02, 2019)

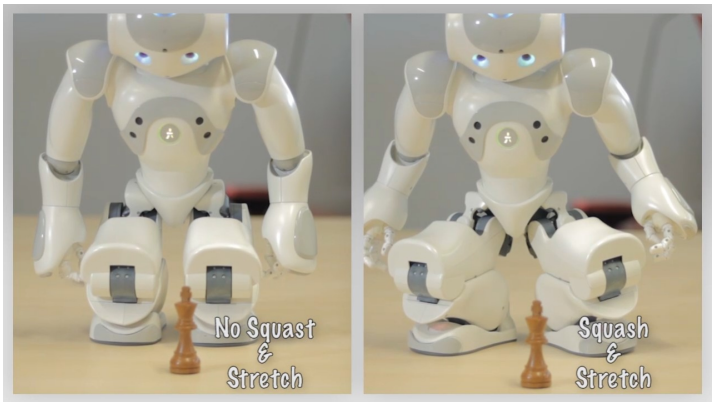

Fig. 6 The principle of Squash & Stretch shown on the NAO robot.

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 [29].

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.

In Figure 7 we can see how a humanoid character that is going to crouch may first slightly stretch upwards.

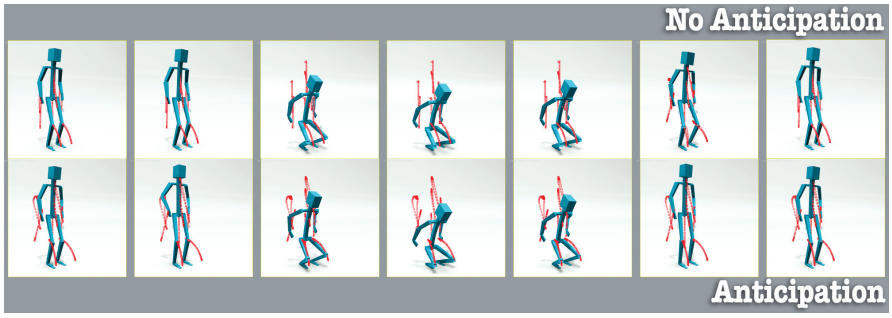

Fig. 7 An animation sequence denoting the principle of Anticipation. The red marks represent the trajectory of the most relevant joints.

The concept can be better explained by looking at a simple animation curve example. Figure 8 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.

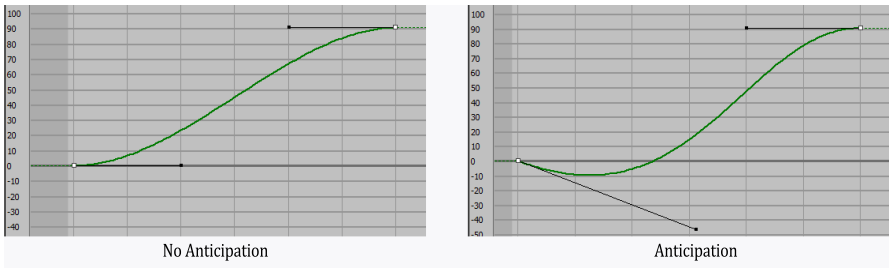

Fig. 8 Animation curves demonstrating anticipation. The left curve does not have anticipation; The right curve does.

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 [29]. In either cases, the key here is showing the intention of the robot.

We can see in Figure 9 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.

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.

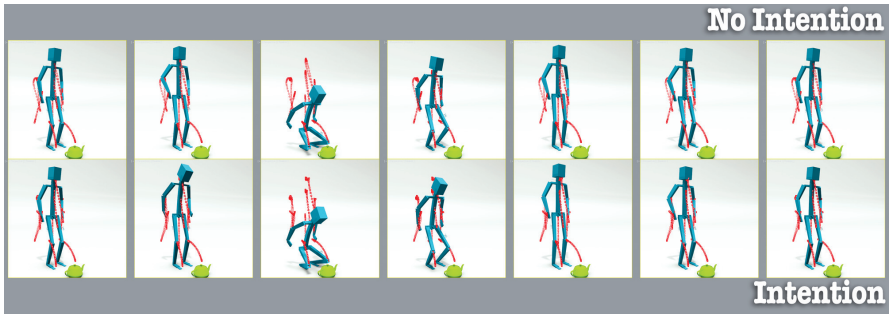

Fig. 9 An animation sequence denoting the principle of Intention. The red marks represent the trajectory of the most relevant joints.

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.

In Figure 10 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 [29]).

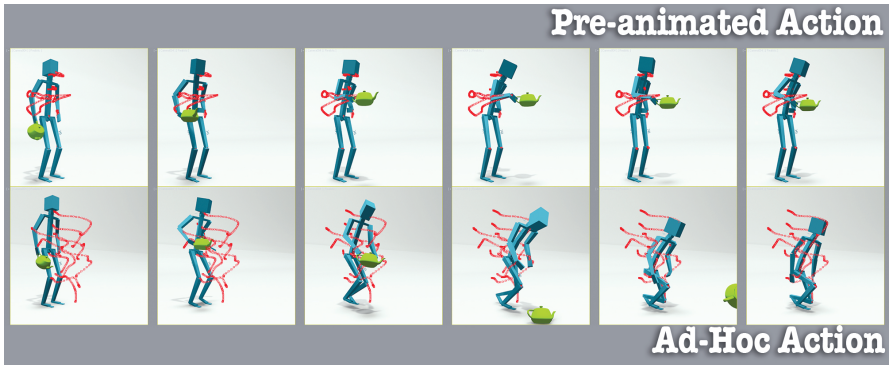

Fig. 10 An animation sequence denoting the principles of Pre-animated and Ad-hoc Action. The red marks represent the trajectory of the most relevant joints.

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 [6]. 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 11 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.

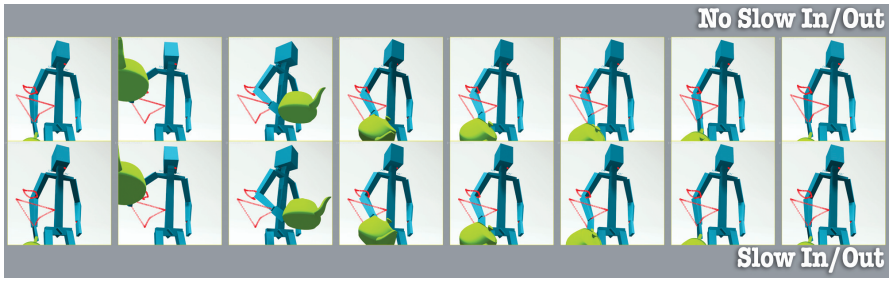

Fig. 11 An animation sequence denoting the principle of Slow In/Out. The red marks represent the trajectory of the most relevant joints. Notice how more frames are placed at the points of the trajectory where the motion changes in direction, in particular within the triangular-shaped portion. More spacing between points, using a fixed time-step, yields a faster motion.

This is more noticeable if we look at the animation curves. Figure 12 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.

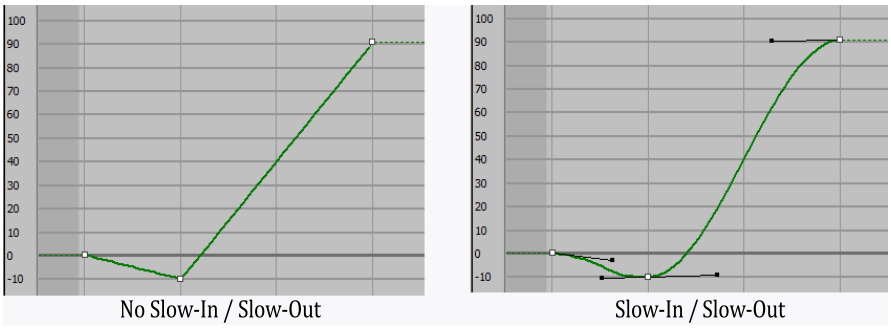

Fig. 12 Animation curves demonstrating Slow In and Slow-Out. The left curve does not have Slow In / Out; The right curve does.

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 13.

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 14 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.

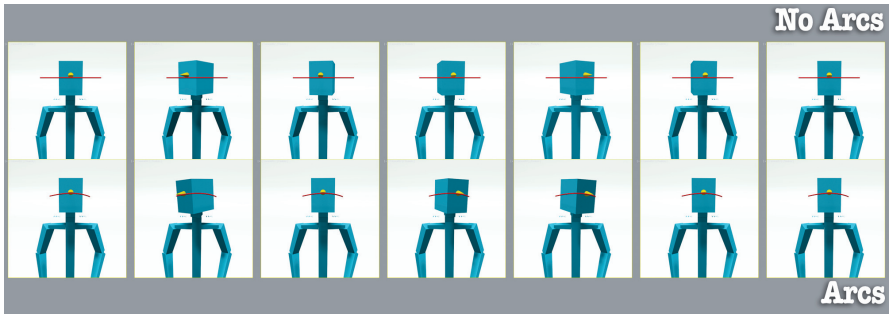

Fig. 13 An animation sequence denoting the principle of Arcs. The red marks represent the trajectory of the most relevant joints.

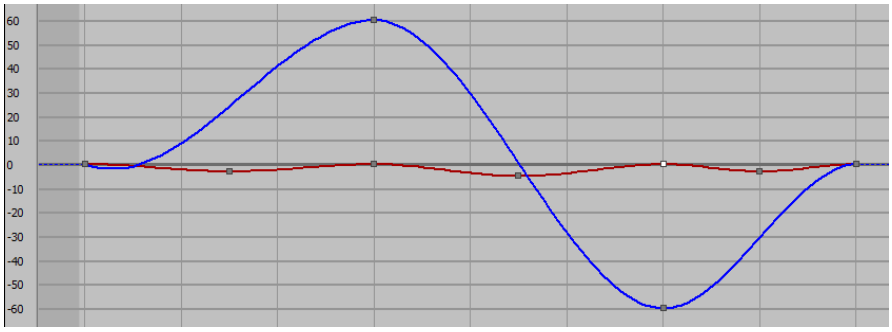

Fig. 14 Animation curves demonstrating Arcs. The blue curve is the Panning DoF, rotating from the rest pose, to its left (60 degrees) and then to its right (-60 degrees), and then back to rest. During this motion, the Pitch joint (red curve) slightly waves between those key-frames.

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 movements. It is actually a feature that can be implemented in animation systems by contrasting the motion signal [12].

Figure 15 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 16 presents a snapshot from the video[15] illustrating how this principle looks like on the NAO robot, while Figure 17 show the same for the EMYS robot.

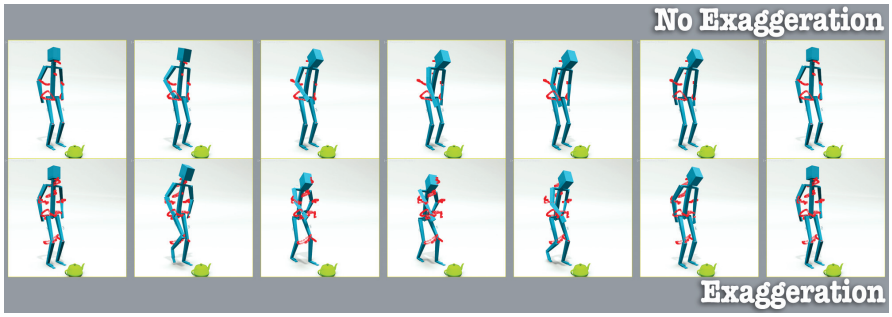

Fig. 15 An animation sequence denoting the principle of Exaggeration. The red marks represent the trajectory of the most relevant joints.

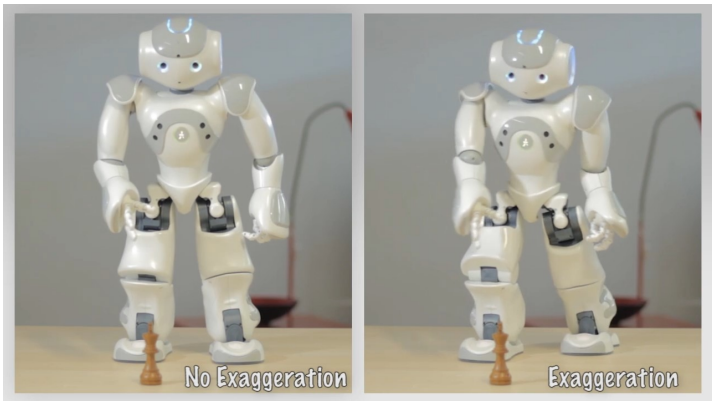

Fig. 16 The principle of Exaggeration exemplified on the NAO robot.

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 18 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

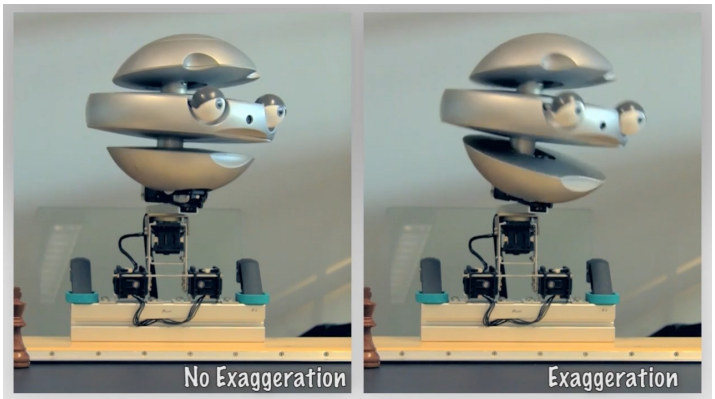

Fig. 17 The principle of Exaggeration exemplified on the EMYS robot.

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.

Fig. 18 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 behavior, 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 19 shows a character portraying another Principle - Idle Behavior, while also standing asymmetrically. This Idle Behavior is performed by the simulation of breathing and by slightly waving its arms like if they were mere pendulums.

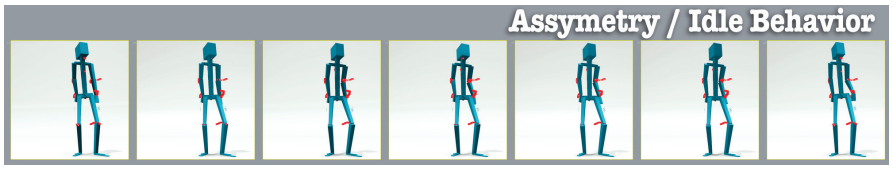

Fig. 19 An animation sequence denoting the principles of Asymmetry and Idle Behavior. The red marks represent the trajectory of the most relevant joints.

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 20 shows two characters performing the same kind of behavior, 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' [32], and his arguments have inspired us to agree. He considers that the design and behavior 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[16] 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.

[16] `www.pleoworld.com` (accessed March 02, 2019)

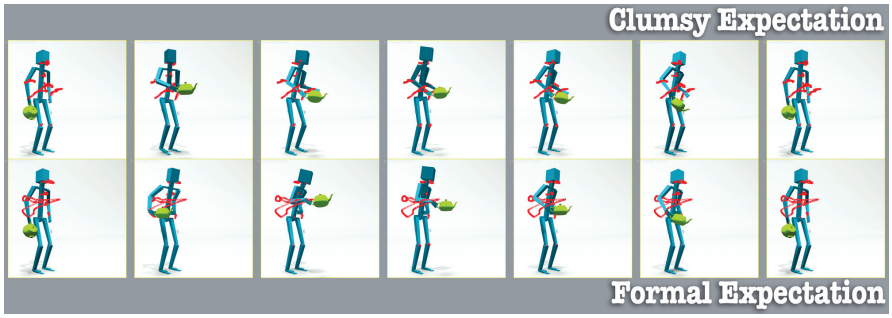

Fig. 20 An animation sequence denoting the principle of Expectation. The red marks represent the trajectory of the most relevant joints. Notice how the clumsy version balances the teapot around instead of holding it straight, and waves around its left ar instead of holding it closer to its body, delivering a feeling of discourtesy.

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 what it's doing [27, 29].

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 21 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.

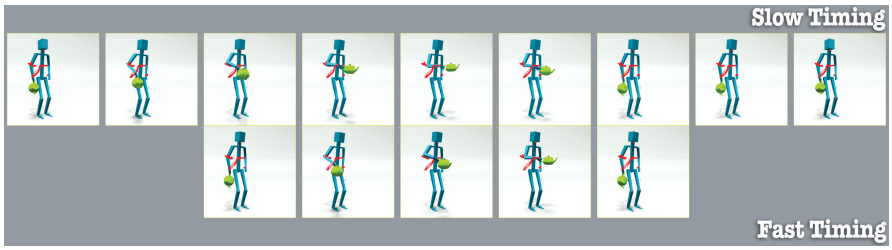

Fig. 21 An animation sequence denoting the principle of Timing. The red marks represent the trajectory of the most relevant joints.

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 [28]. 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.

5 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 designated 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[6], Maya[7], Houdini[8] or Blender[9]. These existing animation packages allow to export animation files using general-purpose formats such as Autodesk FBX[17]. 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.

5.1 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 22 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

[17] `https://www.autodesk.com/products/fbx/overview` (accessed March 02, 2019)

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.

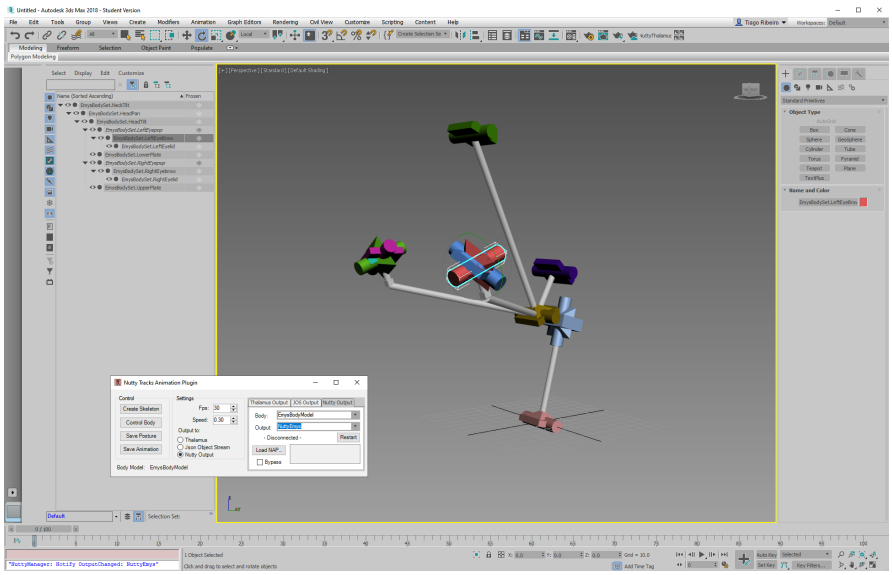

Fig. 22 A screenshot of the Nutty Tracks plug-in for Autodesk 3dsmax, illustrating the skeletal animation rig created by the plug-in. An animator can generate this rig through the simple click of a button, and then use the plug-in to export the final animation to a Nutty-compatible animation file.

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 23 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 23. 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.

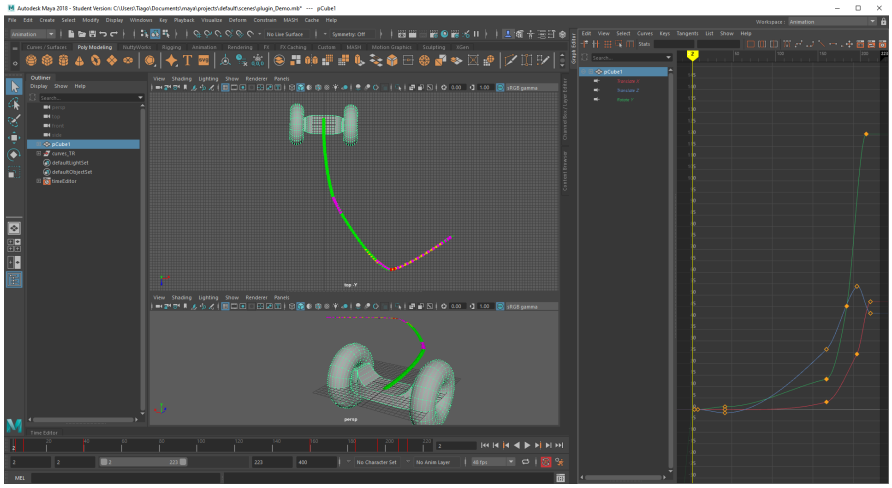

Fig. 23 A screenshot illustrating the robot-animation trajectory-helper feature implemented through a plug-in into Autodesk Maya. This feature draws the motion trajectory as a path directly into the scene of the animation software, and highlights the points of the trajectory that break any of the robot's kinematic limits.

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 23, 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[18], 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 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.

5.2 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

[18] `https://gagosian.com/exhibitions/2018/urs-fischer-play/` (accessed March 02, 2019)

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, animation programs are *programmable* animations. These can take in certain parameters that are provided throughout the interaction, and using motion sources such as animation files, static poses, or signal generators such as sine-waves and Perlin noise, compose them into a final resulting motion in a way that was both directed by an animator, and managed in real-time interaction by the AI, robotic and perceptual system.

Animation programs can, at a very basic level, be specified by some kind of mark-up code. However, taking inspiration from currently existing tools such as Autodesk's Slate material editor[19], or the Unreal Material Editor[20], which provide artist-friendly shader-programming interfaces, 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[21], which allows for the scripting of new interface tools. In this case, because a game engine such as Unity3D already provides 3d visualization and animation tools, it could be extended with a robot animation programming tool in order to become a fully-fledged robot animation designing, programming and pre-visualization tool.

Nutty Tracks provides an example of how such an animation-programming editor may be presented[22]. Its programmable animation GUI is also shown in Figure 24. 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 composed in real-time in Nutty Tracks, while allowing the parameters to be tinkered with, something which could not be properly visualized within the typical animation design software. 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

---

[19] 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 March 02, 2019)

[20] https://docs.unrealengine.com/en-us/Engine/Rendering/Materials/Editor/Interface (accessed March 02, 2019)

[21] https://www.unity3d.com (accessed March 02, 2019)

[22] https://vimeo.com/67197221(accessed March 02, 2019)

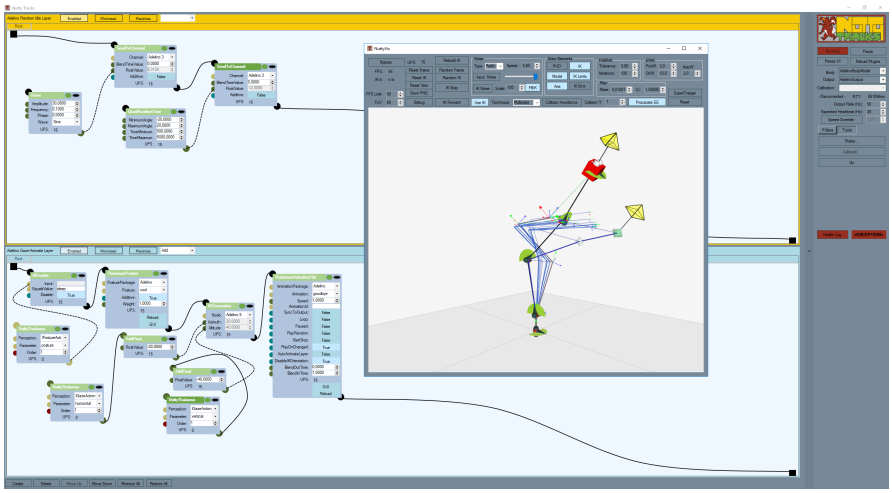

Fig. 24 The Nutty Tracks GUI, used for animation programming in a multi-layer, multi-block visual editor. Within the figure, we see several different animation blocks which either generate or operate on motion signals. The integrated 3D visualizer allows and artist to preview the output of the motion based on how he tinkers with the parameters. It additionally includes an Inverse Kinematics interactive visualizer which allows an animator to tweak the solver, in order to adjust the generated motion to the robot's kinematic capabilities.

programmable animations using operator- and generator-blocks may have a parallel with certain motion control nodes found in some animation software, but the way they are used and composed may 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.

6 Conclusion

Throughout this chapter we have presented our perspective on how robot animation can become an integral process in the development of social robots, based on theories and practices that have been created through the last century, in the fields of both traditional and 3D computer-graphics character animation. We have introduced and described the 12 principles of robot animation, as a foundation that aims at aiding the transfer of the previous character animation practices into the new robot animation ones. In the traditional character animation workflow, characters and their motions are designed to be faithfully played-back on screens. One of the most relevant steps in this transition is the ability to not only design, but also program how animations

should be shaped, merged and behave during interaction with human users. We must therefore introduce new techniques and methods that allow such artistically crafted animations to become not only interactive (such as in video-games), but to interact in the real world, with real users. Such new techniques and methods will be provided by new tools and workflows that are designed with artists in mind, and that aim at the technical requirements imposed by robotics. Upon establishing such techniques, such artists may become a new type of animators which we call robot animators. These are not only experts in traditional character animation, but also know how animation must be designed for robots, and how it should be adapted and shaped during real-world interactions. By following and implementing such paradigms, we expect that social robots may become more akin to animated characters, in a sense that they are able to interact with users in social settings while properly exhibiting the illusion of life.

Acknowledgements This work was supported by national funds through FCT - Fundação para a Ciência e a Tecnologia with references UID/CEC/50021/2019 and SFRH/BD/97150/2013.

References

1. Alves-Oliveira, P., Kiister, D., Kappas, A., Paiva, A.: Psychological science in HRI: Striving for a more integrated field of research. AAAI Fall Symp. Tech. Rep. **FS-16-01 -**(PG - 2-5), 2–5 (2016). DOI 10.1016/j.intimp.2016.04.032
2. Baxter, P., Kennedy, J., Senft, E., Lemaignan, S., Belpaeme, T.: From characterising three years of HRI to methodology and reporting recommendations. ACM/IEEE International Conference on Human-Robot Interaction **2016-April**(December 2017), 391–398 (2016). DOI 10.1109/HRI.2016.7451777
3. Beck, J.: The Animated Movie Guide. Cappella Bks. Chicago Review Press (2005)
4. Bendazzi, G.: Cartoons: One Hundred Years of Cinema Animation. Indiana University Press (1994)
5. Breazeal, B.C., Brooks, A., Gray, J., Hancher, M., Mcbean, J., Stiehl, D., Strickon, J.: Interactive Theatre. Communications of the ACM **46**(7), 76–84 (2003)
6. Breemen, A.V.: Animation engine for believable interactive user-interface robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems - IROS '04, vol. 3, pp. 2873–2878 (2004). DOI 10.1109/IROS.2004.1389845
7. Canemaker, J.: Tex Avery: The MGM years, 1942-1955. Turner Publishing (1996)
8. Finch, C.: Jim Henson: The Works. Random House (1993)
9. Fong, T.: A survey of socially interactive robots. Robotics and Autonomous Systems **42**(3-4), 143–166 (2003). DOI 10.1016/S0921-8890(02)00372-X. URL `http://linkinghub.elsevier.com/retrieve/pii/S092188900200372X`
10. Gielniak, M.J., Liu, C.K., Thomaz, A.: Secondary action in robot motion. RO-MAN, 2010 IEEE pp. 3921–3927 (2010). DOI 10.1109/ICRA.2011.5980348
11. Gielniak, M.J., Thomaz, A.L.: Anticipation in Robot Motion. Roman, 2011 (2011)
12. Gielniak, M.J., Thomaz, A.L.: Enhancing interaction through exaggerated motion synthesis. ACM/IEEE International Conference on Human-Robot Interaction - HRI '12 p. 375 (2012). DOI 10.1145/2157689.2157813
13. Gray, J., Hoffman, G., Adalgeirsson, S.O., Berlin, M., Breazeal, C.: 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)

14. Hoffman, G.: Dumb robots, smart phones: A case study of music listening companionship. IEEE International Symposium on Robot and Human Interactive Communication - RO-MAN '12 pp. 358–363 (2012). DOI 10.1109/ROMAN.2012.6343779
15. Hoffman, G., Ju, W.: Designing Robots With Movement in Mind. Journal of Human-Robot Interaction **3**(1), 89 (2014). DOI 10.5898/JHRI.3.1.Hoffman
16. Hoffman, G., Kubat, R., Breazeal, C.: A hybrid control system for puppeteering a live robotic stage actor. IEEE International Symposium on Robot and Human Interactive Communication - RO-MAN '08 pp. 354–359 (2008). DOI 10.1109/ROMAN.2008.4600691
17. Hoffman, G., Weinberg, G.: Gesture-based human-robot Jazz improvisation. In: IEEE International Conference on Robotics and Automation - ICRA '10, pp. 582–587 (2010). DOI 10.1109/ROBOT.2010.5509182
18. Lasseter, J.: 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). DOI 10.1145/37402.37407
19. Lasseter, J.: Tricks to animating characters with a computer. ACM International Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '01 **35**(2), 45–47 (2001). DOI 10.1145/563693.563706. URL http://dl.acm.org/citation.cfm?id=563706
20. Mead, R., Mataric, M.J.: 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 (2010)
21. Murphy, R., Nomura, T., Billard, A., Burke, J.: HumanâĂŞRobot Interaction. IEEE Robotics & Automation Magazine **17**(2), 85–89 (2010). DOI 10.1109/MRA.2010.936953. URL http://ieeexplore.ieee.org/document/5481144/
22. Ribeiro, T., Dooley, D., Paiva, A.: Nutty Tracks: Symbolic Animation Pipeline for Expressive Robotics. ACM International Conference on Computer Graphics and Interactive Techniques Posters - SIGGRAPH '13 p. 4503 (2013)
23. Ribeiro, T., Paiva, A.: The Illusion of Robotic Life Principles and Practices of Animation for Robots. In: ACM/IEEE International Conference on Human-Robot Interaction - HRI '12, 1937, pp. 383–390 (2012)
24. Ribeiro, T., Paiva, A.: Animating the Adelino Robot with ERIK. In: Proceedings of the 19th ACM International Conference on Multimodal Interaction, pp. 388–396. ACM, Glasgow, UK (2017)
25. Ribeiro, T., Pereira, A., Di Tullio, E., Paiva, A.: The SERA ecosystem: Socially Expressive Robotics Architecture for Autonomous Human-Robot Interaction. In: AAAI Spring Symposium (2016)
26. Roberts, S.: Character Animation in 3D. Elsevier (2004)
27. Saerbeck, M., Bartneck, C.: Perception of Affect Elicited by Robot Motion. Journal of Personality pp. 53–60 (2010). DOI 10.1145/1734454.1734473. URL http://portal.acm.org/citation.cfm?doid=1734454.1734473
28. Suguitan, M., Hoffman, G.: Blossom: A Handcrafted Open-Source Robot. ACM Trans. Hum.-Robot Interact. **8**(1) (2019)
29. Takayama, L., Dooley, D., Ju, W.: Expressing thought. In: ACM/IEEE International Conference on Human-Robot Interaction - HRI '11, p. 69 (2011). DOI 10.1145/1957656.1957674
30. Thomas, F., Johnston, O.: The Illusion of Life: Disney Animation. Hyperion (1995)
31. Wang, L.C., Chen, C.: A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. IEEE Transactions on Robotics and Automation **7**(4), 489–499 (1991). DOI 10.1109/70.86079
32. Wistort, R.: Only robots on the inside. interactions **17**(2), 72–74 (2010). URL http://dl.acm.org/citation.cfm?id=1699792