Parameterized Animated Activities

Alba M. Rios Rodriguez
ETH Zurich

Steven Poulakos
DisneyResearch|Studios

Maurizio Nitti
DisneyResearch|Studios

Mattia Ryffel
DisneyResearch|Studios

Robert W. Sumner
DisneyResearch|Studios

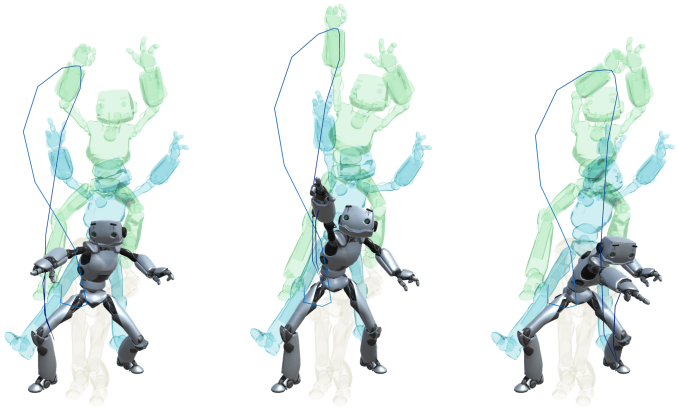

Figure 1: Pointing animation. The original animation (on left) and resulting variations generated by our method. The trajectory deformation is visualized with a time varying line color from light to dark over the animation sequence duration.

ABSTRACT

This work addresses the development of a character animation editing method that accommodates animation changes while preserving the animator's original artistic intent. Our goal is to give the artist control over the automatic editing of animations by extending them with artist-defined metadata. We propose a metadata representation that describes which aspects of an animation can be varied. To make the authoring process easier, we have developed an interface for specifying the metadata. Our method extracts a collection of trajectories of both effectors and objects for the animation. We approximate and parameterize the trajectories with a series of cubic Bézier curves. Then, we generate a set of high-level parameters for editing which are related to trajectory deformations. The only possible deformations are those that preserve the fine structure of the original motion. From the trajectories, we use inverse kinematics to generate a new animation that conforms to the user's edits while preserving the overall character of the original.

CCS CONCEPTS

• **Computing methodologies** → **Animation**; **Motion processing**; *Graphics systems and interfaces.*

KEYWORDS

character animation, motion editing, artistic intent

ACM Reference Format:
Alba M. Rios Rodriguez, Steven Poulakos, Maurizio Nitti, Mattia Ryffel, and Robert W. Sumner. 2019. Parameterized Animated Activities. In *Motion, Interaction and Games (MIG '19), October 28–30, 2019, Newcastle upon Tyne, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3359566.3360062

MIG '19, October 28–30, 2019, Newcastle upon Tyne, United Kingdom

ACM ISBN 978-1-4503-6994-7/19/10…$15.00
https://doi.org/10.1145/3359566.3360062

1 INTRODUCTION

Through subtle variations in motion, talented animators bring digital characters to life. Expressive character animation gives digital characters identity, mood, and personality. However, creating this expressive motion has always been a challenging task. In order to achieve the highest quality result, animators most often utilize keyframing tools, since it allows them to carefully craft and control all aspects of a character's movement. In this process, animators carefully construct the pose of digital characters at key points in the animation, which are interpolated by the animation software. Generating these key poses, achieving the appropriate timing [Whitaker and Halas 2013], and effectively incorporating the principles of animation [Johnston and Thomas 1981] is a time consuming and costly process. Unfortunately, once the investment has been made in creating an initial animation, few tools exist to reuse that investment on similar animation tasks.

We address this situation by developing a parameterization of humanoid animation with intent-preserving editing. Our work forms the basis of an animation modification solution that can make animation adjustment more efficient and enhance real-time applications like games that require run-time animation edits. It is

especially appropriate for adapting animations that involve object interaction. Our solution allows the digital artist to participate actively in the editing process by extending animations with artist-defined metadata. We explore the notion that the artistic intent is encoded in the spatiotemporal trajectories of both effectors, such as hands, and scene objects. The state of these elements over time is represented through events occurring along the animation's time frame. Thus, as part of the animation process, the animator creates metadata using a custom interface to indicate which effectors and which objects can be adjusted, as well as at which times during the animation timeline these adjustments are possible.

The animation modification system stores the relevant trajectories over time of all metadata elements. Internally, the trajectories are represented by cubic Bézier splines. The purpose of this representation is to help deform the trajectories while preserving the fine structure of the motion. Finally, we use the modified trajectory points as the targets of an inverse kinematics system to compute the character pose at each frame of the animation. Our method can generate different variations of the same animation that preserve the original intent of the artist. We demonstrate results showing various animation adjustments such as pointing, grabbing, pouring, and sitting that preserve the overall style of the original animation.

2 RELATED WORK

The creation of stylised motion remains an important challenge in animation. In particular, human motion frequently portrays physical and emotional traits of the character. However, the term 'style' has been defined in a wide variety of ways, and its meaning and scope depend on the judgement of the proposer. Hsu et al. [2005] describe style as variations in the execution of the same action, such as walking and running. Later, Min et al. [2010] incorporate emotion into the definition, differentiating between a neutral, sad or angry walk. They delve deeper into the difference between style and identity, determined by how a specific character performs an action. For example, an adult walks differently than a child. However, more often than not, we find that emotion and identity are included in the definition of style [Aristidou et al. 2017; Shapiro et al. 2006]. This is because stylistic variations are very subtle and often coupled in the motion data, which makes it difficult to make a distinction between them. In our work, we focus on artistic intent rather than style, which covers all of these aspects at the same time, and we explore the notion that it is encoded in the spatiotemporal trajectories of both effectors and objects.

2.1 Motion Editing

We examine a variety of motion editing methods which aim to synthesize animations from existing motion data according to particular requirements. The results are usually short motions similar to the original data. This excludes a large body of work on motion synthesis, which is more versatile than motion editing as it can be used to generate a wider range of motions. The methods presented here are divided into four main categories: constraint-based, motion graphs, interpolation-based and statistical-based.

2.1.1 Constraint-Based Motion Editing. Contraint-based editing is a well-known problem that has been extensively explored. It consists of modifying a given animation to satisfy a set of constraints through the editing of specific keyframes. Constraints can offer intuitive control over the motion of a character, especially over the end-effectors. Our method falls into this category, as we also define constraints in the form of animation metadata. Motion Warping [1995] is an early work on this topic, where a new animation is generated from a set of sparse constraints. These correspond to character poses that are defined by the animator at specific frames. More recently, Shapiro et al. [2007] combine a given motion with a motion planner to produce a new character animation that avoids and manipulates objects. Ho et al. [2010] describe the spatial relationship between single or multiple characters, or between characters and a restricted environment, through a simple structure called an interaction mesh. This method is adequate for handling motions in which the spatial relationships between different body parts of characters are essential for capturing the scene semantics, such as dancing, fighting or getting into a car. Interaction meshes can be used to preserve the spatial relationships and reduce inappropriate interpenetrations, however, they are not applicable for preserving artistic intent or character style, because these do not always relate to spatial relationships between two distinct entities. In another line of work, Kim et al. [2009] propose high-level editing of synchronized human motion data by manipulation of spatial and temporal constraints. First, the constraints are specified in the form of annotations for each of the motion clips. Then, the authors employ a one-dimensional version of Laplacian mesh editing to deform the character trajectories in an as-rigid-as-possible manner. Similarly, we annotate the animations with metadata and we aim to preserve the underlying structure of the effector and object trajectories.

There are also sketch-based approaches to motion editing, which represent the motion constraints with user sketches. Guay et al. [2015] propose a method for animation design from a single user sketch. The generated animations can be further refined by providing new sketches. Motivated by this work, SketchiMo [2016] is a sketching interface to allow users to modify poses and edit joint trajectories by sketching curves through the interface.

2.1.2 Motion Graphs. This line of research was opened by Kovar et al. [2008] who developed a system for generating different styles of locomotion along user-specified 2D trajectories. The motion data is divided into fragments corresponding to character poses, which are then reassembled to create new animations. The major limitation is that motion data is not actually being modified, and therefore any generated animation is limited to a combination of the motions in the database. The approach also requires gathering a large amount of example data.

2.1.3 Statistical-Based Motion Editing. In this category of methods, the new motion sequences are generated by applying statistical and machine learning models. However, they need to construct an extensive motion database to train the neural networks. Despite being placed in different categories, there is no absolute boundary between these and motion graph-based methods, as the latter may use statistics in a particular step. The same applies to statistical models. A recent example is presented by Holden et al. [2016]. In this work, the authors propose a framework to synthesize human movement and to edit the generated motions on the space of the

motion manifold. Users can generate motion variations by inputting constraints such as the trajectory over terrain for the character to follow or trajectories for the end-effectors. Their system can produce high-quality animations with no pre-processing needed.

2.1.4 Interpolation-Based Motion Editing. This is a technique that extrapolates the new motions through interpolation of a broad set of example motions. Although this approach has been successfully demonstrated, it is significantly limited by the number of example animations. Rose et al. [1998] proposed to interpolate a set of annotated example motions by using radial basis functions. Later, this work is extended to be used in inverse kinematics problems [Rose III et al. 2001]. Kovar and Gleicher [2004] control the interpolation process by using weighted interpolation functions. In their work, they give the user control over a set of properties of the generated motions, such as end-effector locations or joint velocities. More recently, Huang and Kallmann [2010] use direct optimization to precisely satisfy spatial constraints in real time. Their work is demonstrated for both walking and upper-body actions, such as pointing to distant objects and pouring tea. Min et al. [2010] present a model for editing personalized motion styles, but extend the idea of modeling both the style and identity of the motion. To construct the model they use a large corpus of example motions annotated in terms of identity and style attributes.

2.2 Inverse Kinematics

In humanoid characters, inverse Kinematics (IK) is used to obtain a single pose given a set of constraints in the form of joint positions and rotations. The dominant trend in the past years is to use data-driven techniques that try to take advantage of the large volume of motion capture data available [Holden et al. 2016; Wei and Chai 2011]. In case there are time limitations, heuristic-based algorithms can be used for solving the IK problem with a low computational cost. The most popular heuristic-based algorithms are CCD [1991] and FABRIK [2011]. A comprehensive overview of inverse kinematics techniques can be found in [Aristidou et al. 2018].

Although the base IK problem has been well studied, the generated poses may often appear mechanical or unnatural. Over the years, there has been a growing effort to create more natural looking results. Grochow et al. [2004] propose a style-based IK that uses a learned model of human poses to generate the most likely pose satisfying a given set of constraints. The system can produce natural-looking results, but those are limited to poses that are close to the training data. Liu et al. [2005] extend this work by incorporating elements of locomotion such as gravity or preference for using certain joints rather than others. Although they demonstrate the efficacy of their method in style transfer, the system has the disadvantage of being computationally intensive. Later, Aristidou et al. [2017] propose a method to control the style of a given motion by correlating the human poses with emotion. In a more recent work, Xia et al. [2015] present a data-driven solution to generate stylized motions from plain, non-stylized motion data in real-time.

In our work, the choice of IK solver plays an important role in the quality of the results. We use an extension of FABRIK [2011] for biped characters. We chose this solver for its low computational cost, which makes it significantly faster than other methods, and allows us to achieve an immediate set up of the system by still achieving reasonable human poses.

2.3 Animation in Games

In the game industry, creating compelling animation for game characters is an essential part of the development process. Although there has been extensive research on human motion editing, the standard is to generate massive animation libraries and to enforce strict metrics on the environment. Animation editing comes down to procedural post-processing of the animation data and the extent to which animations are modified after their creation is relatively small. If the results are not satisfactory, solutions involve either going back and editing the animation data or solving the issue in the post-processing layer [Dowsett and Harrower 2015; Laidacker 2013]. IK Rig [Bereznyak 2016], as one of the most advanced animation technologies in the game industry, is designed to transfer animation data between rigs of both different sizes and skeletal hierarchies, as well as amending the animations to represent the size or weight of the rig.

2.4 Learning from Demonstration

Learning from Demonstration (LfD) is a technique used on robotic applications to teach robots how to correctly execute a specific behavior through a demonstration. The first step is to obtain the examples from the teacher. A relevant approach is to record the robot trajectory during the demonstration [Bagnell et al. 2007; Ratliff et al. 2006]. Then, a common strategy to generate new behaviors is to include teacher annotations. Similarly, Nicolescu and Mataric [2003] present a method that allows the teacher to provide verbal instructions and informative cues beyond the demonstration experience. Finally, van Lent et al. [2001] propose to annotate traces with the times at which goals are achieved or abandoned. A comprehensive survey of the existing techniques in LfD can be found in [Argall et al. 2009].

In summary, LfD shares common aspects with our work, such as representing behaviors as a set of spatiotemporal trajectories. Additionally, the use of artist-generated metadata to define character operations plays a similar role to teacher annotations.

3 METADATA REPRESENTATION

The task of creating an animated motion should ideally stop after obtaining the initial animation. However, if we want to re-use the motion data for a different action, we need to make alterations to it. There are a number of issues that can make this editing a challenging task: we have nothing but the original data, there is little sign of what the key properties of the motion are and - if working with motion capture data - the artist may not be familiar with the intent of the original performance. In this section, we propose a metadata representation that enables artists to specify the important elements to retain in an animation while performing motion editing. The metadata defines common animation constraints, as well as the modifiable elements of the animation, that can be specified at specific frames. It is independent of the animation technique used to generate the original motion.

The proposed metadata representation, including a short description of each item, is shown in Listing 1. The artist can specify two categories of editable elements: objects and effectors. The supported effectors are: pelvis, left/right shoulder, left/right thigh, left/right hand and left/right foot. The metadata language includes constraints between elements, including object and end-effector position and look at, and can be used to identify for each frame the type of editable elements and the type of edition possible. The set of constraints at a specific frame is what we call an animation event. The metadata language can be easily extended as needed to encode additional data.

```
{
 "objects":[{
   "objectName":"The object's name",
   "offsetName":"The offset point from pivot, if any",
   "isModifiable":true,
   "isReplaceable":true
  },
  ...
 ],
 "effectors":[
  {
   "effectorName":"The end-effector's name"
  },
  ...
 ],
 "events":[{
   "keyframe":0,
   "modifiableEffectors":[
    "The name of the first modifiable effector",
    "The name of the second modifiable effector",
    ...
   ],
   "constraints":[{
     "constrainedElement":"The constrained element's name",
     "constrainingElement":"The constraining element's name",
    },
    ...
   ],
   "lookAt":"The name of the element to look at"
  }
 ]
}
```

Listing 1: Example metadata stored in text-format.

In addition to the above, we developed a Unity plugin for metadata authoring (Figure 2). A preview of the selected animation is shown at the bottom of the window. In the preview area, the user can playback the animation or select specific frames. The first step is to specify the parameterizable objects and effectors. Then, the timeline can be used to specify the animation constraints at specific frames. Details about the constraints at the selected frame are displayed under the timeline area.

4 ANIMATION EDITING

Once the metadata for the animation has been authored, the animation/metadata pair can be used as input to the animation editing system. Figure 3 shows an overview of the steps involved in the animation editing process and the relationship between them. The first goal is to compute the trajectories over time of all parameterizable elements, which can be both objects and effectors, in the original animation. To compute such trajectories, we first retrieve the parameterizable elements from the metadata, and then we sample the state of the elements at every frame of the original animation. Then, we generate a set of Bézier splines to approximate each of the trajectories. The purpose of the splines is to help deform the trajectories while preserving the structure of the motion. In the next step, the system will determine a set of parameters to control the modification, as well as constraints on what should not be modified during the editing process. Finally, we find a set of poses that

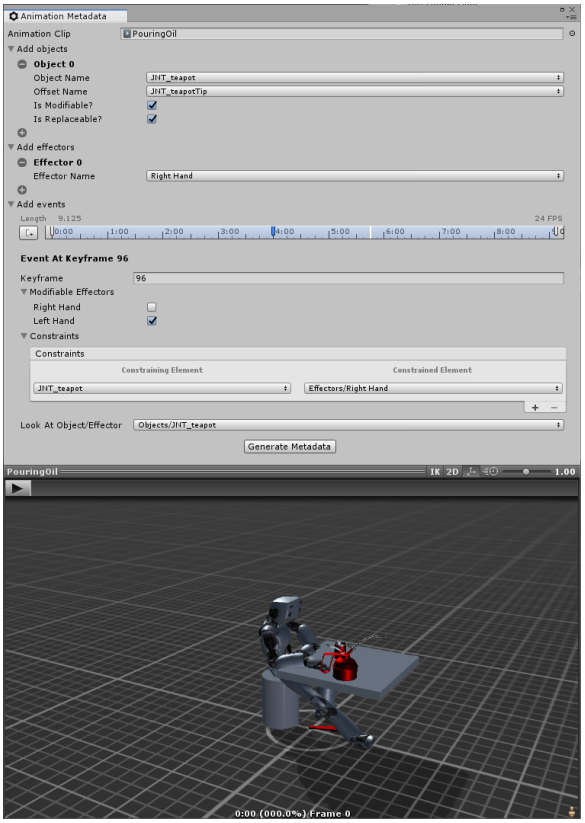

Figure 2: The metadata authoring interface built in Unity with the pouring tea animation selected.

generate a new animation that meets the requirements provided in both the metadata and the new trajectories. As a result, the output of our algorithm is a modified version of the original animation that preserves its artistic intent.

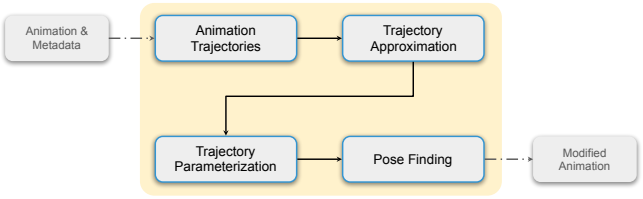

Figure 3: The steps involved in the animation editing process. The input is the animation/metadata pair. The output is a modified version of the original animation that preserves the original artistic intent.

4.1 Animation Trajectories

Similar to previous animation editing methods, we represent the animation through spatiotemporal trajectories [Choi et al. 2016; Kim et al. 2009]. The first step of this process is to retrieve the parameterizable elements from the metadata. In the metadata, the parameterizable elements are stored in two lists, based on element

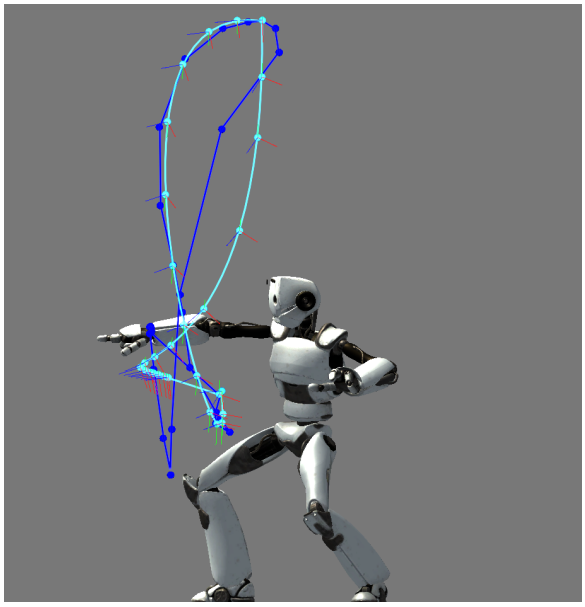

Figure 4: The trajectory and corresponding spline of the right hand effector in the pointing animation. The trajectory and the spline are displayed in blue and cyan respectively. The coordinate-system axes of each trajectory point are displayed in red (x-axis), green (y-axis) and blue (z-axis).

type. The next step is to play back the animation at every frame. For a frame i, we sample the state of all the parameterizable elements. Besides, at frame i we compute a rule for the look-at target, which can be described as a triple parameter $< from, to, weight >$. The weight is uniformly distributed through the frames between the previous and following events. After sampling the animation, we have a spatiotemporal trajectory for each of the parameterizable elements, consisting of the element state at each frame plus a look-at rule. The trajectory of an effector is shown in Figure 4.

4.2 Trajectory Approximation

The first step is to split the trajectories. Each trajectory is divided into $N-1$ sub-trajectories, where N corresponds to the number of metadata events. A sub-trajectory contains all the element positions that are between two consecutive events.

We have a set of M data points, $(M_0, \ldots, M_j, \ldots, M_{m-1})$, and we want to find a cubic Bézier curve that meets the following conditions

(1) The first and last data points (i.e., M_0 and M_{m-1}) must correspond to the curve endings.
(2) Then, the curve must approximate the data polygon using least squares.

The first step towards this goal is to determine a set of parameters, $(t_0, \ldots, t_j, \ldots, t_{m-1})$, in the domain $[0, 1]$ to fix the data points at particular values . Note that there is one t parameter per data point. There are many parameterization methods, such as uniform, chord length or centripetal, each with advantages and disadvantages. For cubic splines, it has been proven that the centripetal method produces results that are closer to the data polygon [Floater 2008]. Therefore, we use such method to generate our set of parameters.

Once we have determined the parameters, we proceed with the task of approximating the sub-trajectories with Bézier curves. We are interested in cubic Bézier curves (i.e. $d = 3$), which are defined by four points $< P_0, P_1, P_2, P_3 >$. We want the curve to pass through the first and last data points, which gives $P_0 = B(0) = M_0$ and $P_3 = B(1) = M_{m-1}$. There are only two unknowns, which are control points P_1 and P_2. Hence, the least squares error that we are seeking to minimize is

$$f(P_1, P_2) = \sum_{j=1}^{m-2} |M_j - B(t_j)|^2 \tag{1}$$

Our goal is to find a pair of values, P_1 and P_2, that minimizes equation (1). Solving the equation gives us the solution

$$P_1 = (A_1 B_1 - A_{12} C_2)/(A_1 A_2 - A_{12}^2) \tag{2}$$

$$P_2 = (A_1 B_2 - A_{12} C_1)/(A_1 A_2 - A_{12}^2) \tag{3}$$

where

$$A_1 = 9 \sum_{j=0}^{m-2} t_j^2 (1 - t_j)^4$$

$$A_2 = 9 \sum_{j=0}^{m-2} t_j^4 (1 - t_j)^2$$

$$A_{12} = 9 \sum_{j=0}^{m-2} t_j^3 (1 - t_j)^3$$

$$B_1 = \sum_{j=0}^{m-2} 3 t_j (1 - t_j)^2 [p_j - (1 - t_j) M_0 - t_j^3 M_{m-1}]$$

$$B_2 = \sum_{j=0}^{m-2} 3 t_j^2 (1 - t_j) [p_j - (1 - t_j) M_0 - t_j^3 M_{m-1}]$$

By following these steps, we have approximated each subtrajectory with a Bézier curve defined by parameters $< P_0, P_1, P_2, P_3 >$. The curves are joined end-to-end, forming a point-wise continuous Bézier spline. This can be seen in Figure 4. In this case, the continuity order, along with other curve properties, is not relevant as the spline will be only used as a tool for deforming the trajectories.

4.3 Trajectory Parameterization

As described in the previous section, we approximate each animation trajectory by a series of Bézier curves that are connected end-to-end forming a Bézier spline. The curve parameters t play an essential role in the trajectory deformation. These can be used to calculate the matching curve point to any given trajectory point by merely evaluating the curve function at the corresponding parameter. Our goal is to identify a way of parameterizing the trajectory points with respect to their corresponding curve in the spline. We seek to facilitate the modification process by applying the desired adjustments to the spline rather than directly to the trajectory.

We have adopted the method of *curve skeleton skinning* presented by Yang et al. [2006] to suit the problem of trajectory parameterization. The original method is used to generate realistic skin deformation by representing the relationship between the movement

of the skeleton and skin deformation in a non-linear continuous fashion. In our case, this approach can be used to generate deformations on the animation trajectories from local transformations of the Bézier curve points. The deformed trajectories preserve the original structure of the motion thereby preserving the artistic intent. This technique is easy to use and allows full control over the deformation process.

As shown in Figure 4, we construct a complete coordinate-system around each spline point. The curves that form the spline are defined by four parameters $< P_0, P_1, P_2, P_3 >$, where P_0 and P_3 correspond to points in the original trajectory. Following this, a coordinate system can be constructed in three steps

(1) First of all, we must compute the coordinate-system of the spline points that are associated with data points in the original trajectory. The spline is composed by a set of $N-1$ curves, $(B(t)_0, \ldots, B(t)_k, \ldots, B(t)_{n-2})$, each with two endings P_0^k and P_3^k, which correspond to the points that we are looking for. The coordinate-system at these points can be defined from the curve geometry

$$\begin{aligned} a &= \frac{\partial}{\partial t} B(t)_k \\ b &= z\, a \\ c &= a\, b \end{aligned} \tag{4}$$

where

$$z = (0, 0, 1)$$

(2) Next, we must compute the coordinate-system of the rest of the spline points, which are the points inside the curve endings. The coordinate-system at these points can be calculated by linear interpolation between the coordinate-systems at the curve endings computed in 1. Note that we have a set of parameters t, each corresponding to a curve point. Thus, we have

$$(a, b, c) = t(a_1, b_1, c_1) + (1 - t)(a_2, b_2, c_2) \tag{5}$$

(3) We can further create a 4x4 transformation matrix A by arranging the vectors of the coordinate-system as column vectors.

Once we have defined a coordinate-system for all the spline points, we must transform each trajectory point $M_j = (x, y, z)$ to the local coordinate-system $E_j = (a, b, c, o)$. To do this, we can use the previously defined transformation matrix A_j.

After the trajectory points are bound with the Bézier spline, deforming the trajectory is straightforward. First, we modify the spline and re-calculate the coordinate-systems at each spline point. Then, we can directly transform the local coordinates of each trajectory point with the associated local coordinate-system to obtain the new trajectory positions. We only allow modifications to points shared between the spline and the trajectory. To ensure the preservation of C^1 continuity, the tangents between the points are maintained during the deformation process.

4.4 Pose Finding

The goal of this section is to generate a humanoid animation that meets the requirements provided in both the metadata and our new trajectories. In the case of objects, the solution is straightforward. At each frame, the desired positioning is already encoded in the new trajectory of the object.

The placement of effectors is a more complicated issue because moving an effector affects the whole body of the character. In this case, we use a full body inverse kinematics solver to find a pose for the character in which its parameterizable effectors are in the desired trajectory positions. At each trajectory point, we go through the state of all the parameterizable elements and use them as input for the IK solver. Then, we make an additional call to the solver to satisfy the look-at constraints.

After repeating the same process for all trajectory points, we get a set of poses in which the modifiable effectors are placed as desired and the character is either looking at a specific position or in the process of doing so. Finally, if we put together the poses and object states in their proper consecutive order, we obtain the new animation. This animation is baked and can be used at run-time.

5 RESULTS

This section includes a series of results generated with our animation generation solution. The metadata definitions can be found in the supplemental material.

We have built a simple user interface to visually modify the animation trajectories. The user can click on the modifiable trajectory points and update their position in the scene, which automatically deforms the corresponding trajectory. Once the trajectories have been set to the desired values, the user can generate the new animation clip. Even an untrained user can control and modify the animation trajectories with confidence and ease.

In Figure 1, we generate a variation of a pointing animation. We modify the position of the right hand at the last frame of the animation to make the character point to a different location.

In Figure 5, we generate multiple variations of a pouring tea animation. Figure 5(A) is the original animation. In Figure 5(B), we scale and translate the teacup to a different position on the table. Furthermore, we adjust the position of the left hand so that it does not collide with the newly positioned cup. In Figure 5(C), we rotate the teapot. In Figure 5(D), we replace the teapot by a similar one with a much shorter spout. Our solution generates the correct result due to a metadata constraint between the teapot and the teacup. Directly replacing the teapot in the original animation results in an incorrect pouring position.

In Figure 6, we generate a variation of a sitting animation. We modify the position of the pelvis to make the character sit at a lower height. The upper body posture of the character is maintained due to a metadata constraint between both hands and the pelvis.

In Figure 7, we generate a variation of an examine-with-lamp animation. We moved the right hand at a specific trajectory point so that the character examines a higher position. The lamp follows the movement of the hand due to a metadata constraint between them.

6 EVALUATION

We have conducted a preliminary evaluation of the authoring aspect of the system by having a digital artist generate a few metadata files for different animations, visualize the results and then provide feedback. All of the evaluated components have been implemented in

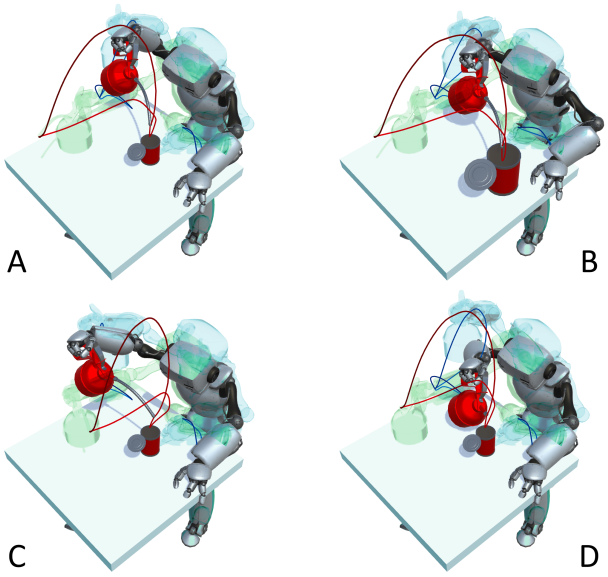

Figure 5: Pouring tea animation. Comparison between original animation and three resulting animations generated by our method.

Figure 6: Sitting animation. Comparison between original animation (on left) and resulting animation generated by our method.

the Unity game engine. The goal of the evaluation is to demonstrate that our system can support a digital artist through the animation generation process. We hypothesize that the system, in combination with artist-defined metadata, is able to generate animation modifications that preserve the original intent of the artist. We focus on the evaluation of the authoring interface and the quality of the generated results. The evaluation process is divided into three phases: an introduction to the metadata interface, several authoring tasks for the artist and a final interview to collect feedback.

Phase 1: Introduction. First, the artist was given an overview of the interface and the relationship with metadata properties. Then, we demonstrated the generation of metadata for the "examine with lamp" animation visualized in Figure 7. To illustrate the importance of the metadata, we authored two different metadata representations for the same animation. The artist witnessed the whole authoring process, from the definition of objects and effectors to the specification of particular events.

Figure 7: Examine-with-lamp animation. Comparison between original animation (on left) and resulting animation generated by our method.

Phase 2: Authoring Task. Later, the artist was given the task to generate metadata for the "pointing" animation in Figure 1. The target of this task was to generate metadata that allowed to modify the pointing target. Once the first task was completed, we moved onto the more complex "pouring tea" animation in Figure 5. In this case, the target of the task was to allow for modification of both the teapot and the cup. The expectation was that the authoring interface would allow the artist, newly introduced to the system, to create metadata intuitively and with a minimal learning curve. We encountered no serious problems during the evaluation. However, by observing the artist using the interface, we have gathered valuable feedback about the system. We believe that we are going in the right direction with the current metadata representation, but that the authoring interface can still be improved.

Phase 3: Interview. To conclude the evaluation process, we interviewed the artist and collected feedback. Initially, we asked about the clarity and understanding of the metadata representation. We attained the most insightful feedback from this question. The artist mentioned that the concept of globally modifiable objects is not intuitive enough and asked for further clarification. The artist also pointed out that the implications of adding both objects and effectors would be better understood if we displayed their trajectories in the animation preview. In this way, the user would be given a hint of the animation modification method. The last proposal was to update the representation of metadata events. Currently, the events are global, so they have an impact on all the parameterizable elements. The artist pointed out that we could improve this representation by making the events local to the elements. In other words, each element would be influenced only by the events in which it participates. We agree that all of these are valid observations and that they would improve the representation. Finally, we showed the artist a series of animations generated with the system. The feedback was that most improvements could be achieved in

the *pose finding* step. The desired improvements are all related to the modification of walking motions. For these motions, the artist hopes for a custom solver with basic reasoning of physics from the effector trajectories.

At last, we asked for suggestions or ideas for improving the system. The one suggestion was to reason about the pull vectors of both the elbows and knees of the character when solving for the poses. The artist believes that this could make a big difference in the quality of the results and that the poses would look more natural.

7 LIMITATIONS AND FUTURE WORK

The presented method experiences problems when dealing with animations that do not have an explicit mapping to our metadata representation. Because we want to give as much control as possible to the artist, the quality of the results is significantly affected by the defined metadata. Therefore, if the artist does not find an adequate mapping, the results may be unpredictable. For interactions, such as pointing or pouring tea, we can easily find spatiotemporal constraints between the character and the environment. More complex motions, such as walking, are left for future work.

The animation editing system generates the modified character poses by using an inverse kinematics heuristic solver [Aristidou and Lasenby 2011]. The low computational cost of the solver compromises the quality of the resulting poses, which may look unnatural. By adding bio-mechanical constraints in the solver, we can limit the motion of the joints and avoid unnatural poses [Maurel and Thalmann 2000]. Still, such improvement does not address more dynamic physical properties, such as momentum conservation or force constraints. More effort is required to address this issue of physically plausible motions [Rabbani and Kry 2016; Shapiro and Lee 2011].

Independently of the solver, the input of the inverse kinematics problem is the desired set of positions for the effectors, which are called targets. A target is said to be unreachable if it is located further than the character can reach or if there is no way to pose the character to reach it. A significant amount of processing time could be saved by computing the reachable zone of the character. Additionally, there is a considerable benefit in constraining the animation parameters to the character-reachable zone. In Figure 8 we can see examples exceeding the maximum reachable zone of the character for the pointing and the pouring tea animations.

In addition to the above, there is a lot to be gained by extending the system to deal with collision avoidance. In Figure 9 we can see two examples of possible collisions with the environment in the pouring tea animation. Tao and Yang [2017] propose an approach for collision-free motion based on the FABRIK algorithm [2011]. Another alternative to solve the collision problem can be found in [Brown et al. 2004]. Along the same line, Unzueta et al. [2008] present an IK algorithm that includes a self-collision detection step.

The presented method aims to serve as the basis for an animation modification solution that can be fully integrated into applications that have time constraints. This target imposes an essential constraint upon the computational cost of the inverse kinematics solver. As a preliminary step, we use a method that decimates an animation trajectory to a similar trajectory with fewer points. The tolerance

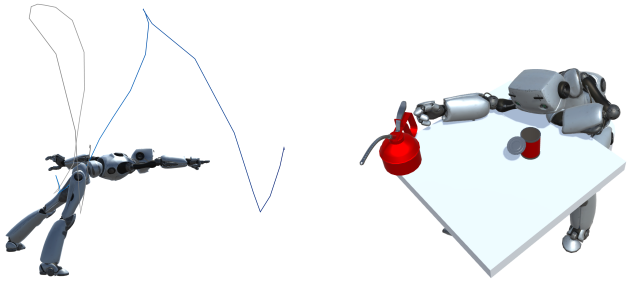

Figure 8: Maximum character-reachable zone in the pointing and pouring tea animation. The inverse kinematic targets are placed outside of the reachable zone which results in unsolvable character poses.

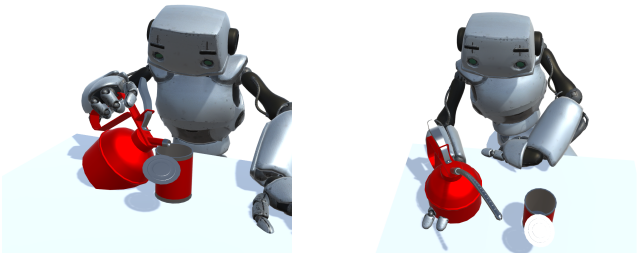

Figure 9: Example collisions in the pouring tea animation.

Table 1: Comparison of the execution time (seconds) of the *animation editing* system with and without trajectory simplification.

	Pointing		Pouring tea	
	Simplif.	No simplif.	Simplif.	No simplif.
Pose Finding	0.0052s	0.0070s	0.0341s	0.0390s
Total	0.0199s	0.0231s	0.1732s	0.2008s

of the method should be chosen with care because it is a parameter that affects the final shape of the curve. Only the remaining points are used in the *pose finding* step. A comparison of the execution time of the *pose finding* step before and after applying this trajectory simplification method can be seen in Table 1. Additional improvements in this direction as well as performance validation will be provided as future work.

8 CONCLUSION

There has been extensive research on human motion editing, including style-based editing, and yet the standard in the industry is to generate massive animation libraries and to enforce strict metrics on the environment. The extent to which animations are modified after their creation is relatively small and typically includes only minor touch-ups using inverse kinematcs in a post-processing step.

For applications with limited resources to devote to animation quality, animation sequences can end up looking robotic, and the overall visual quality can be negatively affected. In recent years,

the growing availability of online motion libraries has given rise to editing methods based on captured data. These methods often seek to reduce the artist's work in the animation generation process. On the contrary, our goal is to have artists actively participate in the post-processing of animations by extending the modification process with artist-defined metadata. While our method still requires manual annotation for each animation clip, the proposed metadata provides artists and developers a common tool to edit motions while retaining the artist's intent.

By focusing on limitations in the current development process, we have been able to come back to more simple solutions than the contemporary line of research in motion editing. Our method is capable of generating different variations of the same animation while preserving the original intent of the artist. The method uses metadata to encode the artistic intent of the animation. The results reflect a good compromise between intent preservation and animation quality. The feedback collected during preliminary evaluation provides guidance on how to proceed further and enhance the quality of our results.

In summary, we have built a system that can generate variations of the same animation in a fast and straightforward manner. The system saves time and resources by avoiding the cost of going back and forth in the animation process.

ACKNOWLEDGMENTS

We would like to thank Loic Ciccone, Miriam Tschanen, Michael Neff, Alexander Shoulson, and Stelian Coros for the enlightening conversations about character animation, motion editing and preserving artistic intent.

REFERENCES

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems* 57, 5 (2009), 469–483.

Andreas Aristidou and Joan Lasenby. 2011. FABRIK: A fast, iterative solver for the Inverse Kinematics problem. *Graphical Models* 73, 5 (2011), 243–260.

Andreas Aristidou, Joan Lasenby, Yiorgos Chrysanthou, and Ariel Shamir. 2018. Inverse Kinematics Techniques in Computer Graphics: A Survey. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 35–58.

Andreas Aristidou, Qiong Zeng, Efstathios Stavrakis, KangKang Yin, Daniel Cohen-Or, Yiorgos Chrysanthou, and Baoquan Chen. 2017. Emotion control of unstructured dance movements. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 9.

JA Bagnell, Joel Chestnutt, David M Bradley, and Nathan D Ratliff. 2007. Boosting structured prediction for imitation learning. In *Advances in Neural Information Processing Systems*. 1153–1160.

Alexander Bereznyak. 2016. IK Rig: Procedural Pose Animation. https://www.gdcvault.com/play/1018058/AI-Postmortems-Assassin-s-Creedm

Joel Brown, Jean-Claude Latombe, and Kevin Montgomery. 2004. Real-time knot-tying simulation. *The Visual Computer* 20, 2-3 (2004), 165–179.

Byungkuk Choi, JP Lewis, Yeongho Seol, Seokpyo Hong, Haegwang Eom, Sunjin Jung, Junyong Noh, et al. 2016. SketchiMo: sketch-based motion editing for articulated characters. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 146.

Lee Dowsett and Geoff Harrower. 2015. Animation Bootcamp: UFC Animation System. https://www.gdcvault.com/play/1021794/Animation-Bootcamp-UFC-Animation

Michael S Floater. 2008. On the deviation of a parametric cubic spline interpolant from its data polygon. *Computer Aided Geometric Design* 25, 3 (2008), 148–156.

Keith Grochow, Steven L Martin, Aaron Hertzmann, and Zoran Popović. 2004. Style-based inverse kinematics. In *ACM transactions on graphics (TOG)*, Vol. 23. ACM, 522–531.

Martin Guay, Rémi Ronfard, Michael Gleicher, and Marie-Paule Cani. 2015. Space-time sketching of character animation. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 118.

Edmond SL Ho, Taku Komura, and Chiew-Lan Tai. 2010. Spatial relationship preserving character motion adaptation. In *ACM Transactions on Graphics (TOG)*, Vol. 29. ACM, 33.

Daniel Holden, Jun Saito, and Taku Komura. 2016. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 138.

Eugene Hsu, Kari Pulli, and Jovan Popović. 2005. Style translation for human motion. In *ACM Transactions on Graphics (TOG)*, Vol. 24. ACM, 1082–1089.

Yazhou Huang and Marcelo Kallmann. 2010. Motion parameterization with inverse blending. In *International Conference on Motion in Games*. Springer, 242–253.

Ollie Johnston and Frank Thomas. 1981. *The illusion of life: Disney animation*. Disney Editions New York.

Manmyung Kim, Kyunglyul Hyun, Jongmin Kim, and Jehee Lee. 2009. Synchronized multi-character motion editing. *ACM transactions on graphics (TOG)* 28, 3 (2009), 79.

Lucas Kovar and Michael Gleicher. 2004. Automated extraction and parameterization of motions in large data sets. In *ACM Transactions on Graphics (ToG)*, Vol. 23. ACM, 559–568.

Lucas Kovar, Michael Gleicher, and Frédéric Pighin. 2008. Motion graphs. In *ACM SIGGRAPH 2008 classes*. ACM, 51.

Aleissia Laidacker. 2013. AI Postmortems: Assassin's Creed III, XCOM: Enemy Unknown, and Warframe. https://www.gdcvault.com/play/1018058/AI-Postmortems-Assassin-s-Creedm

C Karen Liu, Aaron Hertzmann, and Zoran Popović. 2005. Learning physics-based motion style with nonlinear inverse optimization. In *ACM Transactions on Graphics (TOG)*, Vol. 24. ACM, 1071–1081.

Walter Maurel and Daniel Thalmann. 2000. Human shoulder modeling including scapulo-thoracic constraint and joint sinus cones. *Computers & Graphics* 24, 2 (2000), 203–218.

Jianyuan Min, Huajun Liu, and Jinxiang Chai. 2010. Synthesis and editing of personalized stylistic human motion. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*. ACM, 39–46.

Monica N Nicolescu and Maja J Mataric. 2003. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. ACM, 241–248.

Amir H Rabbani and Paul G Kry. 2016. PhysIK: Physically Plausible and Intuitive Keyframing.. In *Graphics Interface*. 153–161.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. 2006. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*. ACM, 729–736.

Charles Rose, Michael F Cohen, and Bobby Bodenheimer. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications* 18, 5 (1998), 32–40.

Charles F Rose III, Peter-Pike J Sloan, and Michael F Cohen. 2001. Artist-directed inverse-kinematics using radial basis function interpolation. In *Computer Graphics Forum*, Vol. 20. Wiley Online Library, 239–250.

Ari Shapiro, Yong Cao, and Petros Faloutsos. 2006. Style components. In *Proceedings of Graphics Interface 2006*. Canadian Information Processing Society, 33–39.

Ari Shapiro, Marcelo Kallmann, and Petros Faloutsos. 2007. Interactive motion correction and object manipulation. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*. ACM, 137–144.

Ari Shapiro and Sung-Hee Lee. 2011. Practical character physics for animators. *IEEE computer graphics and applications* 31, 4 (2011), 45–55.

Songqiao Tao and Yumeng Yang. 2017. Collision-free motion planning of a virtual arm based on the FABRIK algorithm. *Robotica* 35, 6 (2017), 1431–1450.

Luis Unzueta, Manuel Peinado, Ronan Boulic, and Ángel Suescun. 2008. Full-body performance animation with sequential inverse kinematics. *Graphical models* 70, 5 (2008), 87–104.

Michael van Lent, John E Laird, et al. 2001. Learning procedural knowledge through observation. In *K-CAP*, Vol. 1. 179–186.

L-CT Wang and Chih-Cheng Chen. 1991. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *IEEE Transactions on Robotics and Automation* 7, 4 (1991), 489–499.

Xiaolin Wei and Jinxiang Chai. 2011. Intuitive interactive human-character posing with millions of example poses. *IEEE Computer Graphics and Applications* 31, 4 (2011), 78–88.

Harold Whitaker and John Halas. 2013. *Timing for animation*. CRC Press.

Andrew Witkin and Zoran Popovic. 1995. Motion warping. In *Siggraph*, Vol. 95. 105–108.

Shihong Xia, Congyi Wang, Jinxiang Chai, and Jessica Hodgins. 2015. Realtime style transfer for unlabeled heterogeneous human motion. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 119.

Xiaosong Yang, Arun Somasekharan, and Jian J Zhang. 2006. Curve skeleton skinning for human and creature characters. *Computer Animation and Virtual Worlds* 17, 3-4 (2006), 281–292.