Spatial Motion Doodles: Sketching Animation in VR Using Hand Gestures and Laban Motion Analysis

Maxime Garcia, Rémi Ronfard, Marie-Paule Cani

Spatial Motion Doodles: Sketching Animation in VR Using Hand Gestures and Laban Motion Analysis

Maxime Garcia
Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LJK
38000 Grenoble, France

Rémi Ronfard
Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LJK
38000 Grenoble, France

Marie-Paule Cani
Ecole Polytechnique, CNRS, LIX
9128, Palaiseau, France

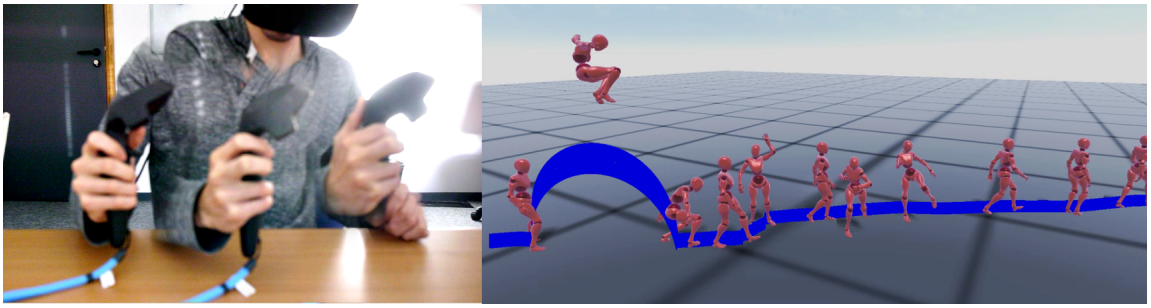


Figure 1: Left: A user drawing a spatial motion doodle (SMD) which is the six-dimensional trajectory of a moving frame (position and orientation), here attached to the HTC Vive controller. Right: The SMD is parsed into a string of motion tokens, allowing to recognize actions and extract the associated motion qualities. This information is transferred to an articulated character to generate an expressive 3D animation sequence.

ABSTRACT

We present a method for easily drafting expressive character animation by playing with instrumented rigid objects. We parse the input 6D trajectories (position and orientation over time) – called *spatial motion doodles* – into sequences of actions and convert them into detailed character animations using a dataset of parameterized motion clips which are automatically fitted to the doodles in terms of global trajectory and timing. Moreover, we capture the expressiveness of user-manipulation by analyzing Laban effort qualities in the input spatial motion doodles and transferring them to the synthetic motions we generate. We validate the ease of use of our system and the expressiveness of the resulting animations through a series of user studies, showing the interest of our approach for interactive digital storytelling applications dedicated to children and non-expert users, as well as for providing fast drafting tools for animators.

CCS CONCEPTS

• **Computing methodologies** → **Procedural animation**; **Motion processing**; *Virtual reality*.

KEYWORDS

Animation system, Character Animation, Laban movement analysis, Expressive animation, Gesture recognition.

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

ACM Reference Format:
Maxime Garcia, Rémi Ronfard, and Marie-Paule Cani. 2020. Spatial Motion Doodles: Sketching Animation in VR Using Hand Gestures and Laban Motion Analysis. In *Motion, Interaction and Games (MIG '19), October 28–30, 2019, Newcastle upon Tyne, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3359566.3360061

1 INTRODUCTION

Creating coordinated and expressive animation of multiple characters is a difficult but necessary step in visual storytelling and interactive game creation, and a bottleneck for novice storytellers and independent game developers not trained in 3D animation. The traditional 3D animation follows a strict pipeline: after the initial storyboarding and character design phases, articulated characters are modeled, rigged and roughly positioned in 3D, at which point the layout of their relative poses can be established. Then, their actions and movements are broken down into keyframes which are interpolated to produce the final animation. Skilled animators need to spend time and efforts carefully tuning animation curves for each character's degrees of freedom and each specific action to be specified; they also need to consistently sequence these actions over time in order to convey the desired character's intentions and personalities. Given the long learning curve of standard animation software, creating expressive animations with this pipeline is clearly out of reach for non-expert users.

Creating character animation directly from storyboard-like sketches or diagrams is a challenging task, and a promising research direction for democratizing computer graphics and animation. *Motion doodles* [Thorne et al. 2004] are interactive, hand-drawn sketches controlling the full-body motion of a character, which can be learned, parsed and interpreted to generate animations almost in real-time. They are however limited to planar, side view sketches.

Our research goal is to enable users to generate expressive 3D animation sequences by playing with tangible rigid objects as freely as in a child's make-believe game. Therefore, we extend the motion doodles paradigm to the context of spatial interaction with instrumented rigid objects. In this work, we use the HTC Vive controller and Valve Lighthouse tracking system, which provides a 6D motion signal at 60 fps. We record the motion of the objects as *spatial motion doodles (SMD)*, ie. trajectories of 6D rigid frames over time, and parse them into a sequence of actions which are translated into animations of articulated characters. Taking advantage of the natural embedding of SMDs in 3D space, we generate expressive animations guided by the coarse space trajectories and timings of the input gestures. Furthermore, we analyze the expressive qualities of the input gestures (light/strong, sudden/sustained) and transfer them to the corresponding character animations.

Three main challenges need to be solved for achieving our goals: firstly, we need to parse SMDs into the desired sequence of character actions, independently from position, orientation or scale of the input gesture; secondly, we need to recognize expressive qualities to be associated with each action; thirdly, we need to synthesize character animation sequences displaying the required actions with the corresponding qualities along the path of the input SMDs.

We solve these challenges for a pre-defined set of possible actions for which neutral-style 3D animations have been provided [1]. Our method works as follows: We first read the SMD in temporal order and transform it into a series of token indicating the changes of direction of the linear and angular components of motion. This enables us to search for regular expressions in terms of changes of directions that correspond to our recognizable actions. Then, each portion of the SMD corresponding to a detected action is assigned motion qualities from Laban's Effort classification, using a Bayesian identification method. Finally, we automatically assemble character animation sequences representing the selected actions by retargeting them to the coarse trajectory of the SMD and the desired motion qualities.

In summary, our three contributions are a new position, rotation and scale invariant pattern extraction method used to detect actions from spatial motion doodles (SMDs); a method for extracting Laban effort features from SMDs using Bayesian classification with a carefully selected set of geometrical features; and a new method for assembling character animation sequences along the path of the SMDs with suitable Laban motion qualities.

2 RELATED WORK

Spatial motion doodles borrow ideas from several lines of previous work, namely sketching animations, gesture recognition, spatial interaction, expressive animation and motion qualities recognition.

2.1 Sketching animations

The idea of using sketches either to time or to create animations is not new. Several existing methods deal with the problem of key framing using 2D strokes. [Terra and Metoyer 2004] use 2D drawing gestures to control and re-time the trajectories of 3D animated characters. [Choi et al. 2012] matches stick figures with existing animation clips from a database, allowing the creation of a key frame animation sequence. [Bai et al. 2016] use dynamics to fill in the details of low-dimensional keyframes, but their work is limited to 2D cartoon animation. [Guay et al. 2013] and [Mahmudi et al. 2016] use 2D lines of action for posing 3D characters or animals and creating keyframed animations. Going further, some methods enable to create and time animation from a single stroke. [Guay et al. 2015] introduce a space-time curve to animate a character body line in a single drawing gesture, while [Choi et al. 2016] proposes a way to edit existing animations using the same kind of input. Lastly, [Ciccone et al. 2017] proposes a new intuitive way of sketching and editing space-time curves for motion cycles.

While these methods can easily be used to create or edit selected parts of animations, they do not tackle the problem of creating an entire character animation sequence. Former methods such as [Thorne et al. 2004] and [Hyun et al. 2016] focus on this issue. Specifically, [Thorne et al. 2004] introduces the concept of *motion doodle*, ie. a 2D curve in side-view, abstracting the main trajectory of the motion sequence. This sketch is parsed into motion primitives and translated into 3D animations stored in a library. Close-to-planar motion is used, and retiming is inferred from the detection of singular points in the geometry of the curve. In contrast, [Hyun et al. 2016] address the design of 3D animations of basketball players from 2D diagrams. The latter are drawn in a floor-plan view, allowing complex multi-player actions to unfold in 3D. The technique however requires an intensive labeling of the input motion-capture dataset using an elaborate grammar of the basketball game. In this work, we would like instead to investigate 3D animation from expressive hand-gestures, where the input position, orientation and speed variations can be used for control.

2.2 Spatial interaction

Early attempts to use 3D interaction techniques for creating animated scenes have used space balls [Gobbetti and Balaguer 1995], full-body motion capture suits [Sturman 1998] or instrumented articulated puppets [Shin et al. 2001]. More recently, [Zhang et al. 2012] animated Chinese shadow characters with gestures recorded by a Kinect camera , [Furukawa et al. 2014] animated 3D characters using a kinect camera and [Jacobson et al. 2014] animated 3D characters using articulated physical puppets.

While such systems are useful for controlling the faces and gestures of virtual characters, they provide little support for animating virtual actors moving freely in large virtual spaces, which is the focus of our paper.

2.3 Expressive animation

Creating stylized animation given a set of expressive parameters such as emotions and attitudes has been explored for a long time in computer graphics. We can distinguish two families of methods: data-driven methods, which attempt to learn models of the chosen emotions and attitudes from motion capture databases [Rose et al. 1998] and stylization methods, which directly modify an input animation using aesthetic features [Neff and Fiume 2005] or more abstract motion descriptors such as Laban motion qualities [Aristidou et al. 2017; Chi et al. 2000; Durupinar et al. 2016; Masuda et al. 2010]. In addition methods such as [Xia et al. 2015; Yumer and Mitra 2016] allow stylization interpolation for heterogeneous movement. Data-driven methods come with the inconvenient of

[1] Our implementation contains 27 animations taken from the Mixamo repository at https://www.mixamo.com/

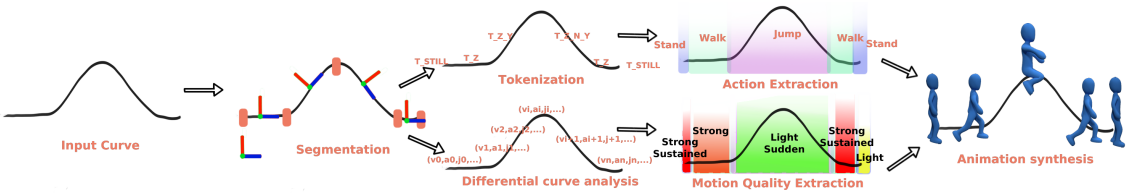


Figure 2: Spatial motion doodle pipeline: We first segment the input curve (spatial component of the SMD) based on changes on motion direction using a moving local frame, and label the segments with motion tokens. We then recognize actions as regular expressions over motion tokens, together with their motion qualities. Finally, we synthesize corresponding animations following the input SMD trajectory, timing and motion qualities.

relying on an animation database and cannot infer a style for a non registered action. Stylization methods on the other hand make it more difficult to ensure that the stylized motion remain plausible.

Our work belongs to the general family of motion graph methods, where existing character animations are re-targeted to user-controlled actions and to the geometry of the scene [Furukawa et al. 2014; Heck and Gleicher 2007; Kovar et al. 2002; Safonova and Hodgins 2007]. Interestingly, motion graphs have been extended to include continuous style variations using a data-driven approach [Min and Chai 2012]. We instead choose a stylization method to change the Laban motion qualities of the animation clips during retargeting.

2.4 Motion quality recognition

Motion qualities are recognizable features of a motion sequence, which should be preserved while translating an input action into an output animation. Motion qualities have been researched extensively in the 1940s by Rudolf Laban for the purpose of dance notation [Laban 1950] and previous work in the computer graphics community has attempted to recognize Laban motion qualities from natural human motion [Bouchard and Badler 2007; Lorenzo Torresani 2007] and to generate 3D character animation with recognizable Laban motion qualities [Bishko 2014; Chi et al. 2000], or both [Aristidou et al. 2017]. In our work, we attempt to recognize Laban motion qualities of weight (light/strong) and time (sudden/sustained) from spatial motion doodles - i.e. motions of a rigid frame moving and rotating over time - and transfer them to full character animation.

Extracting Laban effort qualities from low dimensional motion was explored in the context of wrist motion data [Fdili Alaoui et al. 2017] where they used multi-modal input, including wrist position, speed, acceleration and muscle activation. In contrast, we take as input a single rigid body motion and extract a rich set of geometric and kinematic features, such as Euclidean and equi-affine velocity, acceleration, jerk, curvature and torsion.

3 OVERVIEW

In this paper, we focus on using the position and orientation of a tangible object over time as input. Our system creates an expressive animation sequence from the trajectory of the rigid body, or *spatial motion doodle* (SMD).

In practice, capturing a SMD ie. tracking the moving frame of a manipulated object can be done using a variety of available devices, including video and Kinect cameras, inertial measurement units (IMU), magnetic sensors or optical tracking systems [Gupta et al. 2014; LaViola and Keefe 2011]. In this work, we used the SteamVR Lighthouse tracking system with HTC Vive trackers and controllers.

The processing pipeline from the SMD to the final animation consists of several steps, depicted in Figure 2. We first process the SMD in sequential order in order to assign each point over time to a direction token, using a local frame that moves and orients itself according to the 6D SMD position and orientation data. We switch to a new token when either the direction of the frame's translation or the direction of the frame's rotation changes. The resulting token list is then read in temporal order and analyzed by a state machine. The latter runs an approximate string matching algorithm that recognizes actions from a library of stored patterns. When an action match is found, the state machine transfers the corresponding portion of the SMD to an animation state machine able to convert it into a detailed animation sequence.

In contrast with previous work, our system also extracts Laban effort qualities such as weight (light/strong) and time (sudden/sustained) from the input spatial motion, and transfers them to the detailed animation sequence, enabling us to make it expressive.

The following sections present the three main components of our system, namely the method for parsing spatial motion doodles into sequences of actions (Section 4), the extraction of Laban effort qualities (Section 5), and the assembly of a 3D animation sequence matching the trajectory, timing and qualities of the input spatial motion doodle (Section 6).

4 ACTION RECOGNITION

In this section, we propose a solution for segmenting a continuous motion sequence into a discrete sequence of actions. The latter are represented as regular expressions over a set of spatial motion tokens, defined in the space of all possible local velocity directions in the character's frame.

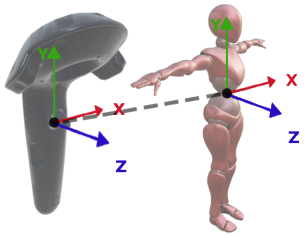


Figure 3: Controller and Character Moving frames. Z axis points towards the front direction, Y points axis up and X points left.

4.1 Tokenization

The input spatial motion doodle (or SMD) $\{p_i\}, p_i = (x_i, y_i, z_i, \theta_i, \phi_i, \gamma_i)$, stores the position of the center of mass and the orientation over time of the tangible object in a global world frame, where y denotes the vertical direction. To be able to consistently recognize actions (eg. jump versus side-step), we need to evaluate velocity directions with respect to the character's frame. Figure 3 shows how the tangible object and the character frame relate to each other where x, y and z axis respectively represent the LEFT, UP and FRONT directions.

After each capture, the SMD is first de-noised (in our implementation, we use a third order Savitzky-Gole method [Savitzky and Golay 1964]). We then segment it into *spatial motion tokens*. The latter represent short snippets of 6D motion where velocity remains in the same bin in terms of velocity direction with respect to the character's frame. More precisely, token values represent bins partitioning the 6D space of linear and angular velocity directions. For instance, the *T_Z* token corresponds to a forward character motion (positive z direction) while *T_N_Z* token corresponds to a backward motion. In our implementation, we use 27 bins in translation and 27 bins in rotation.

For each p_i, we compute the signs of the six linear and angular velocities with respect to each axis, to identify the right bin. This is done in the local character's frame, defined by the current rotation matrix $R_i = R_z(\gamma_i)R_y(\phi_i)R_x(\theta_i)$ at each point $p_i = (x_i, y_i, z_i, \theta_i, \phi_i, \gamma_i)$ of the SMD. The local linear velocity direction $\vec{v}$ relative to the character's orientation is then given by:

$$\vec{v} = R_i(p_i - p_{i-1}). \tag{1}$$

Similarly, we compute the angular velocity relative to the character's orientation $\vec{\omega} = (\theta_i - \theta_{i-1}, \phi_i - \phi_{i-1}, \gamma_i - \gamma_{i-1})$. Evaluating these velocities enables us to partition the SMD into segments with constant velocity directions.

Finally, we identify segments along the SMD where $\|\vec{v}\|^2 + \|\vec{\omega}\|^2 \leq \epsilon$ ($\epsilon = 0.05$ in our case) and assign them to the *T_STILL* token symbolizing immobility.

4.2 Approximate matching

Once the SMD is converted into a sequence of tokens, we use a state machine for approximate matching of the regular expressions associated with actions in our database. Each action is represented as a regular expression of spatial motion tokens containing only $*$, + and | operators, where $*$ means zero or more occurrences, + means one or more occurrences and | is the logical or operator. We provide as supplementary material, a figure showing a subset of actions in our library together with their spatial motion doodles, regular expressions and animation keyframes.

We use approximate matching, rather than exact matching, to allow small errors in the spatial motion tokens drawn by the user. This is done by searching for the action with the smallest Levenshtein distance to the current token string, using a variant of Levenshtein automata [Schulz and Mihov 2002]. Noting that regular expressions in our database are insensitive to repetitions of atomic tokens, we remove all + from action regular expressions. Then, we enumerate all possible sub-expressions of non-repeating tokens by traversing all $*$ and | symbols in the regular expressions. This allows us to customize very compact and efficient Levenshtein automata suitable to actions in our database.

Our action detection method brings several benefits. It is translation and rotation invariant as all computations are performed in a moving frame (in contrast with [Thorne et al. 2004] where the drawing plane frame had to be used). In addition, the method is also scale invariant in space as each token of regular expression is followed by a + or a $*$ making them infinitely repeatable. These properties give more freedom to users: whatever the current position and rotation of the prop, they will be able to execute any recognizable action; furthermore each action can be executed at their preferred size, ie. either within a small or a wide area.

4.3 Learning Regular Expression

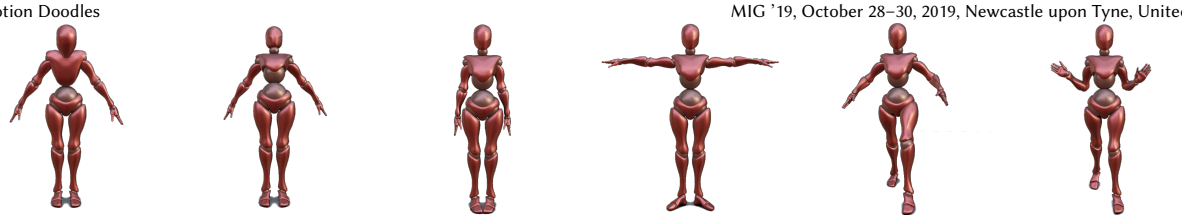
We propose a heuristic approach to learn user dependent action regular expressions from a small number of spatial motion doodle examples. Usually, three examples are sufficient. Learning a regular expression from examples is an ill-posed problem which can accept the composition (or operator) of all the examples as a solution. In our approach, we build a compact regular expression which recognizes both examples as well as inputs that are very similar to them.

Given n examples representing the same action, we compute an average space time doodle using dynamic time warping with l_2 norm as distance function, following an approach first proposed by [Ciccone et al. 2017]. Using this technique, we find point correspondences between the input examples and then compute the average doodle by taking the mean of each correspondences. We then convert the n input doodles and the average doodle into $n + 1$ token sequences $\{X_i\}$.

Our learning algorithm is aimed to align tokens that are repeated more than twice for each sequences, considering them as mandatory in the final learned results (translated as a + in a regular expression). Other tokens are considered as optional or unintentional gestures from the user (translated as a $*$ in a regular expression). Thus, from each sequence $X_i = T_{0i}...T_{ni}$ we extract two subsequences : $M_i = m_{0i}...m_{pi}$ composed of mandatory tokens (preserving the original order) and $O_i = o_{0i}...o_{li}$ composed of optional tokens. For each o_i, we store the index of the next mandatory token in the X_i sequence as well. Then, our system parses mandatory sequences in parallel and build the first part of the final regular expression $R = r_0 + ... r_n +$ where $r_i = (m_{i0}|...|m_{in})$. Note that each mandatory sequence may have different length, consequently if $i \geq \min_j length(M_j)$, r_i will be treated as optional and r_i+ is replaced by r_i*. Finally, we add the remaining optional tokens into the final sequence using the following method: before each token r_i we insert the optional sequence $\widetilde{o_i} = ((o_{k_i 0}...o_{(k_i+p_i)0})|...|((o_{k_i n}...o_{(k_i+p_i)n}))*$ where $(o_{k_i j}...o_{(k_i+p_i)j})$ is the subsequence of X_j composed of optional tokens located between the $m_{(i-1)j}$ and m_{ij} mandatory token. Note that if all subsequences are equal then this optional token sequence is treated as a quick intentional (mandatory) gesture. Finally, we emphasize that, by construction, each training example is contained in the final regular expression R.

5 MOTION QUALITY RECOGNITION

For each detected action, we also extract motion qualities from the corresponding part of the user's input gesture. This is done



(a) Sinking Shape (b) Rising Shape (c) Enclosing Shape (d) Spreading Shape (e) Advancing Shape (f) Retreating Shape

Figure 4: Illustration of Laban's *Shape* category descriptors.

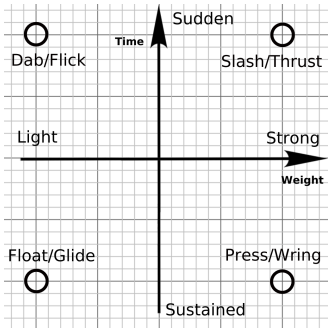


Figure 5: Laban's *Effort* verbs (drives) projected in the time and weight axes.

using Laban Movement Analysis. After presenting the necessary background, we explain how we select features from a spatial motion doodle and train a classifier to recognize some of the main Laban qualities from them.

5.1 Laban movement analysis

Laban Movement Analysis (LMA) [Laban 1950] is a method for characterizing quality in human movements in terms of timing, space used, posture and intention. The full LMA uses five description categories, namely Effort, Shape, Phrasing, Body and Space, to classify motion. In this work, we focus on the first three categories of Effort, Shape and Phrasing, which we now proceed to define briefly.

Effort describes the dynamic quality of motion and is based on the 4 following dimensions (with the associated extreme values): Space (direct - indirect), Time (sustained - sudden), Weight (light - strong) and Flow (bound - free) making the Effort category isomorphic to $[-1;1]^4$. Expert studies on Laban effort expression [Bishko 2014] observed that Space, Time and Weight are often sufficient to describe actions, either two or three of them being active at the same time. Motions involving two descriptors are defined as States and those involving three of them as Drives. Figure 5 illustrates the Laban Effort space of Time and Weight, omitting the Space and Flow dimensions. In this work, we restrict our analysis to the two dimensions of Time and Weight, which best capture the input gesture styles. This section explains how we recognize those styles and transfer them to the corresponding animations.

Shape characterizes body posture during motion. This feature directly influences the movement's convex hull. Similarly to Effort, it is isomorphic to a $[-1;1]^3$ cube and evolves according to 3 factors: Horizontal (enclosing - spreading), Vertical (sinking - rising) and Sagittal (retreating - advancing) as shown in Figure 4.

As reported in [Durupinar et al. 2016] and [Bishko 2014], Effort and Shape can be combined together, as each feature from the first category has expressiveness affinity with the second. For maximal expressiveness, Bishko [Bishko 2014] suggests to associate light effort with a rising shape, strong effort with a sinking shape, sustained effort with an advancing shape and sudden effort with a retreating shape. In Section 6.4, we use this coupling between Effort and Shape to emphasize weight (strong/light) and time (sustained/sudden) qualities by changing the shapes of the 3D character in the corresponding animations.

Phrasing describes the relative durations of the five main stages in human movements (Figure 6, see [Bishko 2014] for details). Among them, *Preparation* is when the character mentally prepares to execute the movement; *Anticipation* is an energy accumulation phase during which the character moves in the opposite direction from the main, subsequent motion; *Execution* corresponds to this main motion; *Follow-through* represents all the extra movements at the end of the main action; *Transition* is the end stage, which leads either to a rest pose or to another action. In practice, some of these stages may be removed or masked depending on the current motion and its context.

5.2 Feature extraction

We train classifiers for Effort qualities of Time and Weight in segments of the SMD corresponding to each detected action. To do so, the first step is to choose candidate features to be computed from those SMD segments.

We use the first three derivatives (velocities, acceleration and jerk) of the six components of the SMD for a total of 18 features. Those can be computed efficiently using finite differences. As for the SMD segmentation into tokens, these features are expressed in the local frame of the character (with the up - down, left - right, and front - back axes) for them to be independent to the orientation if the input gesture.

In addition, we compute the curvature and torsion of the trajectory of the center of the moving frame, as well as its Euclidean and *equi-affine* velocities and their derivatives (accelerations). Let us explain the concept of equi-affine velocity.

In 2D, the equi-affine velocity of a point moving along a planar trajectory is defined as $V_a = |\dot{r}, \ddot{r}|^{\frac{1}{3}}$ and is related to the Euclidean velocity with the formula $V_a = V_e \kappa^{\frac{1}{3}}$, where $\kappa = \frac{|\dot{r}, \ddot{r}|}{\|\dot{r}\|^3}$ is the curvature of the trajectory. Equi-affine velocity is an important in movement studies because is has been shown that human subjects spontaneously draw with constant equi-affine velocity. This is known as the 1/3 power law [Pollick and Sapiro 1997].

In 3D, equi-affine velocity is defined in a similar fashion using the triple scalar product $V_a = |\dot{r}, \ddot{r}, \dddot{r}|^{\frac{1}{6}}$ which is again related to

the Euclidean velocity with the formula $V_a = V_e \kappa^{\frac{1}{3}} \tau^{\frac{1}{6}}$ where κ is the curvature of the stroke as before and $\tau = \frac{|\dot{r}, \ddot{r}, \dddot{r}|}{\|\dot{r} \times \ddot{r}\|^2}$ is the torsion of the stroke. Similarly to the $2D$ case, recent work has shown that human subjects describe spatial movements performed with equi-affine velocity as uniform [Pollick et al. 2009] and this is known as the $1/6$ power law. Those findings make spatial equi-affine velocity a likely candidate to measure subjective motion qualitites and we therefore include it in our feature set. We also compute equi-affine acceleration as the first derivative of V_a. In total, we measure 18 features in moving frame coordinates and 7 features in world coordinates (Euclidean velocity, acceleration and jerk, curvature, torsion, equi-affine velocity, and equi-affine acceleration). We then take the mean values and standard deviations for all of them over the duration of an action to obtain a total number of 50 candidate features.

5.3 Feature selection

In order to validate our choice of features, we created a dataset of spatial motion doodles labeled with Laban weight and time qualities, and computed the intra-class and inter-class variances for all features. We used a total of 200 training examples, captured by asking 5 users to move a rigid object in four different combination of the *Time* and *Weight* qualities, namely strong and sudden, strong and sustained, light and sudden, light and sustained. This was done ten times for each combination. The resulting examples were then used to train the two classifiers as follows.

Considering n classes ($n = 2$ in our case) and m_i examples in the i^{th} class, we computed for each feature f:

$$\overline{f} = \frac{\sum_{i \in [1:n]} \sum_{j \in [1:m_i]} f_{ij}}{\sum_i m_i} \tag{2}$$

$$\forall i \in [1, n], \overline{f_i} = \frac{\sum_{j \in [1:m_i]} f_{ij}}{m_i} \tag{3}$$

followed by the corresponding inter-variance and intra-variance:

$$\sigma_{inter}(f)^2 = \frac{\sum_{i \in [1:n]} m_i (\overline{f_i} - \overline{f})^2}{\sum_i m_i} \tag{4}$$

$$\sigma_{intra}(f)^2 = \frac{\sum_{i \in [1:n]} \sum_{j \in [1:m_i]} (f_{ij} - \overline{f}_i)^2}{\sum_i m_i} \tag{5}$$

The relevance of each feature f was then computed as the ratio $\frac{\sigma_{inter}(f)^2}{\sigma_{intra}(f)^2}$ (the higher the better). Using our training data, this enabled us to evaluate which features of the SMD are the most relevant for Laban classification.

Experimentally, we found that the most relevant features for recognizing *sudden* from *sustained* along the *Time* dimension are *local up speed, acceleration, jerk,* global *Euclidean acceleration* and *local left and right speed* . Remarkably, we also found that the most relevant features for classifying *light* from *strong* along the *Weight* dimension were *equi-affine speed, equi-affine acceleration*, *Euclidean acceleration, speed, jerk* and *torsion*, which confirmed our intuition that equi-affine speed was an important perceptual feature of the spatial motion doodles.

5.4 Bayesian classification

Using the above features and examples, we trained two naive Bayes classifiers to recognize Laban Effort qualities along the *Time* axis (sudden - sustained) and the *Weight* axis (light - strong).

Next, we compute the mean $\overline{f_i}$ and the standard deviation $\sigma^2_{f_i}$ of feature f for the i-th class. We then define, for an input SMD u:

$$P(u \in class_i | u) = \prod_f N(f_u, \overline{f_i}, \sigma^2_{f_i})$$

where f_u is the value of the f-th feature for spatial motion u.

Finally, in order to classify the input SMD u we select:

$$Class(u) = \text{argmax}_i (\prod_f N(f_u, \overline{f_i}, \sigma^2_{f_i}))$$

where the classes are chosen along either the *Time* axis (sudden - sustained) or the *Weight* axis (light - strong).

We computed all 50 features for all labeled training examples, and incrementally trained classifiers for each task using the M most relevant features for that task with M ranging between 1 and 50. We then chose the classifier with the best accuracy for each task, resulting in 4 features for recognizing Time (local UP speed, acceleration and jerk and global Euclidean acceleration) and 2 features for recognizing Weight (equi-affine speed and equi-affine acceleration). The best classification accuracy for each task are reported in Section 7.

6 ANIMATION TRANSFER

Once all actions have been extracted from the input SMD together with their corresponding trajectories, our system classifies each of them into Laban Effort categories and computes an environment adapted trajectory for the animated character. Both trajectory and Laban qualities transfer rely on *phrasing principles* for expressive animation. In particular, we build on this decomposition for transferring Laban effort values in different ways according to the stage of motion among preparation, anticipation, execution, follow through, and transition (see Figure 6). This decomposition also enables us to process differently in place from the moving stages during the trajectory run through.

We first describe the pre-computation we use for segmenting the animation clips in our database according to phrasing, before describing how our system creates an adapted character trajectory as well as our operators for transferring expressive motion features.

6.1 Phrasing detection

The key for segmenting an animation clip, representing an action, is the fact that the end of each animation stage is highlighted by an extreme pose where speed and acceleration both vanish [Bishko 2014]. This enables us to use a semi-automatic method for preprocessing the animations in our database: We apply segmentation at each frame for which both speed and acceleration are null. Then depending on the number of segments, we either use manual or automatic labeling, based on the fact that the five motion stages presented in Figure 6 always take place in the same order.

In practice, some motion stages may not be present for some of the actions: we solve situations where only two extreme poses are

(a) Preparation step (b) Anticipation step (c) Execution step (d) Follow-Through step (e) Transition step

Figure 6: The five main steps of motion: preparation (intention of action), anticipation (energy gathering), execution (energy release through main action), follow-through (follow-up motion induced by the execution), transition (movement either back to hold pose or to connect to next action).

detected between the first and last key-frames by selecting either (anticipation, execution) or (execution, follow-through) for these extreme poses, depending on whether the main direction of motion changes or not between the animation segments ending at these extreme poses. Indeed, anticipation is an energy accumulation stage that makes the whole body move in the opposite direction compared to the execution of the action; in contrast, there is no major change of direction between the execution and the follow-through stages.

6.2 Trajectory transfer

Transforming the sketched doodle into a plausible trajectory for the 3D character is not straightforward. We take care of doing it while preserving the user intent as much as possible.

The initial SMD is firstly projected into the ground so that the character does not perform actions above or under the terrain. We then need to remove unused parts of the trajectory (such as vertical displacements) for actions that are to be performed in place. To do so, during the whole transfer, our system applies and increments a spatial offset s_{off} which account for the parts of the transferred doodle that are removed for in-place actions. For an action A_i and its corresponding trajectory $\{P_{ij}\}$, we apply the spatial offset to each $p \in \{P_{ij}\}$. Then, if the current action is an in-place action, our system increments the spatial offset s_{off} by $\|p_{in} - p_{i0}\|$ and merge all the points of its trajectory into its first point.

Each animation clip corresponding to a detected action is then extracted from the data-base and modified to an expressive animation using the desired Laban qualities (as detailed in Section 6.4). It is then re-timed so that it lasts as long as its corresponding chunk of the character's trajectory. While doing so, we treat cyclic actions such as walking as a special case, in order to prevent artifacts such as foot skating:

Each cyclic animation C_i is subdivided into m cycles $\{P_{ijk}\}_{k=1\ldots m}$ where $\|p_{ink} - p_{i0k}\| = C_{length}$, with C_{length}, the travelled distance during one animation cycle.

Additionally, our systems re-times each non-cyclic trajectory chunks such that the character's root node moves and follows the trajectory during execution, but stays in place during preparation, anticipation, follow-through and transition.It is especially the case for Jumps which are generally composed of in place anticipation and follow through phases as well as a moving execution phase. For a Jump animation J_i whose duration last as much as its corresponding trajectory $\{P_{ij}\}$,ensuring that the character stays in place during the anticipation and follow through phases (ending respectively at t_{a_i} and t_{f_i}), our system applies the following process: $\forall p \in \{P_{ij}\}, t_p \leq t_{a_i} \Rightarrow p = p_{i0}$ and $\forall p \in \{P_{ij}\}, t_p \geq t_{f_i} \Rightarrow p = p_{in}$

Finally, we use well established motion graph techniques to compute smooth animation transitions between sequences corresponding to different actions. Additionally, we detect collisions between the character's feet and the ground and perform simple IK corrections to prevent penetration.

6.3 Transfer operators

Let us now detail how the extracted Laban Effort features (Time and Weight dimensions) are re-targeted to the selected animation clips. Based on the segmentation of each clip into phases, we transfer the extracted qualities using a combination of three operators, namely retiming, rescaling and reshaping. We introduce these operators before explaining how they are used.

Retiming consists in reparametrizing and modifying an animation overall timing $\phi(t)$ which takes as input the time $t \in [start : end]$ and returns the time $t_{reparam}$ at which the animation will be computed. Additionally, this operator can extend or shorten animation phases using linear time scale. In order to preserve consistency between the animation phases, we use recursive functions to define the new time parameterization:

$$f_i(t) = f_{i-1}(end_{i-1}) + a_i.(t - start_i)^{b_i} \tag{6}$$

where i (from 1 to 5) relates to the animation phase, f_0 is the constant function returning time t_0 when the action starts, and a, b are parameters computed from the effort values to be transferred. We use this to apply parameterization different retimings strategies for each of the animation stages while maintaining consistent transitions.

Rescaling enables us to re-scale motion in an adapted way for each of the animation stages: parts of motion may be enhanced or made less salient depending on the needs. This is used for instance to down scale the anticipation part of sudden and sustained actions. Applying a spatial scale to a segment of the animation clip is done as follows: given the start and end keyframes s and e of an animation stage, we compute for each bone B_i its rotation $q_i(s)$ and its translation $t_i(s)$ at the start of the stage. Then we apply scaling using:

$$\begin{cases} q_i(t) & = scale(q_i(s)q_i(t)^{-1}, c)^{-1}q_i(s) \\ scale(q_i(t), c) & = Slerp(q_{unit},\ q_i(t), c) \\ t_i(t) & = t_i(s) + c(t_i(t) - t_i(s)) \end{cases} \tag{7}$$

where c is a parameter depending on the Effort values L_t and L_w, and q_{unit} is the unit quaternion whose angle is null. Note that an additional contact preserving step is performed after scaling using inverse kinematic constraint on contact effectors.

Reshaping enables us to visually enhance the weight dimension of efforts by applying an adequate tuning of the body posture during motion. We do this based on the relation between Laban *Effort*

and *Shape* categories, which states that a light motion is better expressed with a rising body posture, and a strong motion with a sinking posture.

6.4 Transferring Laban efforts

All the input animation clips in the database are chosen to represent neutral versions of the corresponding actions, with uniform keyframes sampling in time. We transfer the extracted *Time* $L_t \in [-1, 1]$ and *Weight* $L_w \in [-1, 1]$ features as follows.

6.4.1 Transferring Time. The aim is to give a feeling of impulse for sudden motions ($L_t > 0$), respectively a feeling of pose and control for sustained ones ($L_t < 0$). In both cases, this implies changing the relative duration of some of the action stages as well as their space amplitude. Preparation and anticipation phases are downscaled by a factor $c = \frac{1}{3|L_t|+1.0}$ using Equation (7) for sustained and sudden motions. Indeed, sudden motions feel unprecedented and should not display the intention of preparing an action while sustained motions do not need anticipation as their execution needs less effort. With respect to the feature selection results we obtained in section 5.3, we apply our retiming operator on both sustained and sudden motion making sustained motions execution longer and anticipation shorter while sudden motions have shorter execution phases and longer anticipations. Our retiming operator also induces an important acceleration phase at the beginning of sudden motions execution phase while inducing a deceleration for sustained motion executions.

Therefore, we use the following implementation of Equation (6.3):

$$\begin{cases} f_{ant}(t + end_{prep}) = \begin{cases} f_{prep}(t_{prep}) + \frac{t}{-0.5L_t+1} \text{ if } L_t \geq 0 \\ f_{prep}(t_{prep}) + (L_t 0.5 + 1)t \text{ otherwise} \end{cases} \\ f_{exec}(t + end_{ant}) = \begin{cases} f_{ant}(t_{ant}) + \frac{1}{2L_t+1} t^{-0.4L_t+1} \text{ if } L_t \geq 0 \\ f_{ant}(t_{ant}) + (-2L_t + 1)t^{-0.4L_t+1} \\ \text{otherwise} \end{cases} \end{cases}$$

where t_{prep}, t_{ant}, and t_{exec} are respectively the end times of the preparation, anticipation, execution, and follow-up stage. Note that for each stage reparametrization $t \in [0 : end_i]$.

6.4.2 Transferring Weight. When transferring Weight, we would like to make light motions ($L_w \leq 0$) feel more uniform and more subtle against more powerful ($L_w \geq 0$) for strong motions. To achieve this effect, we first re-time neutral animations in order to amplify (for strong motions) or flatten (for light motions) speed variations. Thus, we proceed to an arc length reparametrization $\gamma(l)$ of an input animation modifying its speeds.

At each frame i we compute the arc length $L^i = \left(\overline{V_e^i}\right)^{\alpha L_w}(t_i - t_{i-1}) + L^{i-1}$ ($\alpha = 1.5$ in our case) where $\overline{V_e^i}$ is the average joint velocity. Note that for light motions, as L_w is negative we clamp values between 0 and 1 to avoid divergence of the speed when it comes near 0. Finally, we define the Weight reparametrization function in the time domain $\phi(t) = \gamma(\frac{t*L}{T})$ where L is the total arc length and T the total duration of the animation. The resulting effect amplifies speed variations for strong motions and flatten them between 0 and 1 for light motions, making the computed speed more uniform. For neutral animation, $L_w = 0$, $\overline{V_e^i} = 1$ and the initial timing is maintained.

Next, the scaling operator is applied to the overall animation, enhancing strong motions and restraining Light motions. In our implementation, we use $c = \frac{1}{-L_w+1}$ for light motions and $c = L_w + 1$ for Strong motion in Equation (8).

Lastly, we improve Weight transfer by making use of the Shape operator: an adequate rotation is applied to the spine of the character while contact with the ground is maintained. This is done by first classifying the bones of the character into spine bones, intermediate bones and end-effectors. We parse the input skeleton starting at the root and find a chain linking it to the chest and the head. All bones belonging to this chain are labeled as spine bones. Then unlabeled children of spine bones are labeled as intermediate bones. Finally, bones without children are labeled as end effectors. Using this labeling, the *Shape* operator is applied by shifting spine bone rotations along their local x axis by $\delta_\theta = 0.1 * L_w$, depending of the *Weight* value $\theta_{spine} = \theta_{spine} + \delta_\theta$, so that light motions use an uprising body posture while strong motions use down body-postures.

7 RESULTS AND DISCUSSION

We present results of objective user studies to respectively evaluate our action detector, our Laban extraction method and our method for transferring Laban features to input animations. We also present results of a subjective user study to evaluate how expressive the produced animations are and how well to correspond to the user's intents.

7.1 Action detection

We asked users to execute predefined action patterns 10 times, after showing them the recorded controller motion of a typical example. We computed the success rate per action category, measured as the percentage of executions that indeed matched the target regular expression. Results are summarized in Figure 7.

These results show that our system achieves good recognition results with respect to the user intent. Moreover, walking and jumping actions were always recognized as intended. More interestingly the Punch and Throw, which were the actions with the most recognition failures, still have more than 75% of recognition rate. Given that the regular expression for Throw is a sub sequence of the regular expression for Punch, this shows that our system is not so much confused by similarity.

In addition, we evaluated the efficiency of our regular expression learning algorithm by asking 5 users to train our system for 3 different actions (each user had a different action set to train). Users were then asked to test the system thrice with short tests (2 or 3 actions) doodles and several times with longer sequences (at least more than 5 actions). On simple doodles, our system achieved 94% of correct recognition rate. This number decrease to 90% on longer sequence which remains acceptable. The study revealed that motion involving rotations were more difficult to recognize, especially when users rotate the controllers away from their center of gravity, which induces unintended translations.

7.2 Laban weight and time recognition

As explained in Section 5.4, we trained two Bayesian classifiers for the two Laban dimensions Time and Weight, using 40 examples per

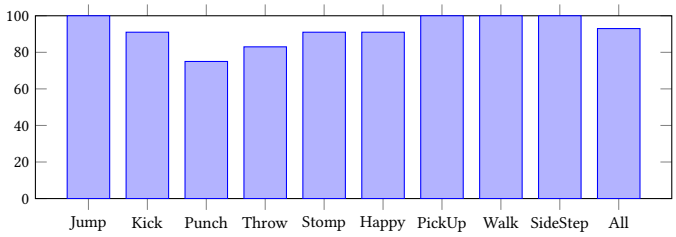


Figure 7: Percentage of actions correctly recognized during our user study.

class in each case and tested it on 20 labeled input curves not belonging to the training set. Figure 8 shows the classification accuracy for recognizing Laban qualities separately in the Time and Weight dimensions, and for recognizing combinations (states) of the two dimensions. We notice that feature selection significantly improves the recognition rates of individual features as well as combined feature (states), but the latter task remains difficult. As described in Section 5.3, we selected 4 features for the Time dimension and 2 features for the Weight dimension.

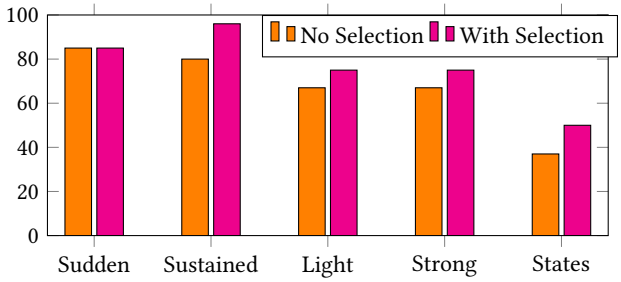


Figure 8: Laban effort classification accuracy in percentage before and after feature selection.

This allows us to independently recognize Laban qualities of Time and Weight with a precision above 80%, even with simple naive Bayes classifiers. Our results also confirm the findings of [Fdili Alaoui et al. 2017] that identifying Laban states combining multiple Laban dimensions is a more difficult task. Feature selection improves the recognition rate for that task by 13% to just above the 50% threshold.

7.3 Laban transfer

We conducted a series of experiments to evaluate the plausibility of our Laban Effort transfer method by testing if non-expert users were able to recognize the motion qualities of the resulting synthetic animations. To achieve this, we showed to 8 users 8 modified animations towards the 4 possible efforts (twice per effort) and asked them, for each animation, to label them among these four efforts. Results in terms of classification accuracy (percentage of correct recognition) are depicted in Figure 9. Those experiments demonstrate recognition rates well above chance level in all cases. We observed that for some cases users mistook strong animations as sudden and light animations as sustained which is also likely to be cause of State classification confusion in our recognition rates.

7.4 Freestyle animation

We tested the ease of use and expressiveness of our method by asking 5 users to freely create 3 complex animations in different

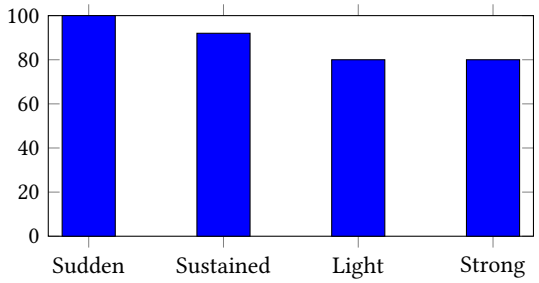


Figure 9: Accuracy of subjective recognition of transferred Laban effort qualities. Users were asked to recognize Laban effort qualities after viewing the transferred animations. Results are reported in percentage.

styles, and asking them to subjectively evaluate the resulting animations. All users first trained our system to use their own gestures during their recordings. The training sessions lasted five minutes on average. Users then created three free-style animations, for a total running time of three minutes each on average. The animations were then presented to them in a subjective view in the HTC Vive headset. Users recognized their intended actions in most of the cases and were generally satisfied with the transferred motion styles. They found the system easy to use. In some cases, false detections occurred while users were pausing or gesturing non intentionally. Transferred animation results created by these users (like the one shown in figure 10) are shown in the accompanying video.

Figure 10: Example of freestyle SMD and animation created by a user during evaluation.

7.5 Limitations

Our method comes with several limitations. Some gestures may be ambiguous, leading to multiple equivalent action sequences. We give priority to actions corresponding to the largest token sequences, which can lead to problems when the number of actions increases.

Another limitation of our system comes from the discomfort caused by the HTC Vive controlers. In future work, we would like to replace them with figurines equipped with HTC Vive trackers, which would make it easier to rest the figurines on the floor and change hands to perform more complex gestures and generating animation for scenes with more than two characters.

Future work is also needed for better recognition and transfer of combined motion qualities. While our system is able to recognize and transfer weight and time qualities separately, it performs poorly with subtle combinations such as light and sustained (as in gliding or floating) or strong and sudden (as in thrusting or slashing).

8 CONCLUSION

In this paper, we introduced a new way of creating expressive character animation sequences by playing with a tangible object. Our system parses the input spatial motion doodle into a sequence of actions, which are automatically translated into character animations with suitable motion qualities. Results show that our system can be used to create simple sequences of playful animations suitable for children and young adults.

This opens up several new directions of research. First of all, we would like to further extend two-handed manipulation for creating stories with coordinated motions of two characters, by learning to recognize two-handed gestures, rather than parallel singled-handed gestures. This would also us to quickly draft multi-character animations involving close interaction and contact. Secondly, dynamics simulation could be combined with our framework to improve the quality and expressiveness of the generated animations. Finally, our method is well suited for quickly drafting animations in immersive environments, and it would be interesting to integrate it with existing virtual reality painting systems.

ACKNOWLEDGMENTS

We would like to thank Laurence BOISSIEUX for the animations she provided for testing, for the 3d model of the garden environment and for her advice and involvement in this project.

REFERENCES

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 (SCA '17)*.

Yunfei Bai, Danny M. Kaufman, C. Karen Liu, and Jovan Popović. 2016. Artist-directed Dynamics for 2D Animation. *ACM Trans. Graph.* 35, 4 (July 2016).

Leslie Bishko. 2014. Animation Principles and Laban Movement Analysis: Movement Frameworks For Creating Empathic Character Performances. ETC Press, Pittsburgh, PA, USA.

D. Bouchard and N. Badler. 2007. Semantic Segmentation of Motion Capture Using Laban Movement Analysis. In *Proceedings of the 7th International Conference on Intelligent Virtual Agents*, Vol. 4722.

Diane Chi, Monica Costa, Liwei Zhao, and Norman Badler. 2000. The EMOTE Model for Effort and Shape. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co.

Byungkuk Choi, Roger Blanco i Ribera, J. P. Lewis, Yeongho Seol, Seokpyo Hong, Haegwang Eom, Sunjin Jung, and Junyong Noh. 2016. SketchiMo: Sketch-based Motion Editing for Articulated Characters. *ACM Trans. Graph.* 35, 4 (July 2016).

M. G. Choi, K. Yang, T. Igarashi, J. Mitani, and J. Lee. 2012. Retrieval and Visualization of Human Motion Data via Stick Figures. *Comput. Graph. Forum* 31, 7 (Sept. 2012).

Loic Ciccone, Martin Guay, Maurizio Nitti, and Robert W. Sumner. 2017. Authoring Motion Cycles. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA '17)*.

Funda Durupinar, Mubbasir Kapadia, Susan Deutsch, Michael Neff, and Norman I. Badler. 2016. PERFORM: Perceptual Approach for Adding OCEAN Personality to Human Motion Using Laban Movement Analysis. *ACM Trans. Graph.* 36, 1 (Oct. 2016).

Sarah Fdili Alaoui, Jules Françoise, Thecla Schiphorst, Karen Studd, and Frederic Bevilacqua. 2017. Seeing, Sensing and Recognizing Laban Movement Qualities. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*.

Masayuki Furukawa, Yasuhiro Akagi, Yukiko Kawai, and Hiroshi Kawasaki. 2014. Interactive 3D Animation Creation and Viewing System Based on Motion Graph and Pose Estimation Method. In *Proceedings of the 22Nd ACM International Conference on Multimedia (MM '14)*.

Enrico Gobbetti and Jean-Francis Balaguer. 1995. An integrated environment to visually construct 3D animations. In *Proc. ACM SIGGRAPH*.

Martin Guay, Marie-Paule Cani, and Remi Ronfard. 2013. The Line of Action: An Intuitive Interface for Expressive Character Posing. *ACM Trans. Graph.* 32, 6 (Nov. 2013).

Martin Guay, Remi Ronfard, Michael Gleicher, and Marie-Paule Cani. 2015. Space-time Sketching of Character Animation. *ACM Trans. Graph.* 34, 4 (July 2015).

Saikat Gupta, Sujin Jang, and Karthik Ramani. 2014. PuppetX: A Framework for Gestural Interactions with User Constructed Playthings. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces (AVI '14)*.

Rachel Heck and Michael Gleicher. 2007. Parametric Motion Graphs. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games (I3D '07)*.

Kyunglyul Hyun, Kyungho Lee, and Jehee Lee. 2016. Motion Grammars for Character Animation. *Comput. Graph. Forum* 35, 2 (May 2016).

Alec Jacobson, Daniele Panozzo, Oliver Glauser, Cédric Pradalier, Otmar Hilliges, and Olga Sorkine-Hornung. 2014. Tangible and Modular Input Device for Character Articulation. *ACM Trans. Graph.* 33, 4 (July 2014).

Lucas Kovar, Michael Gleicher, and Frédéric Pighin. 2002. Motion Graphs. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '02)*. ACM.

Rudolf Laban. 1950. *The mastery of movement.* Macdonald & Evans, London.

Joseph J. LaViola and Daniel F. Keefe. 2011. 3D Spatial Interaction: Applications for Art, Design, and Science. In *ACM SIGGRAPH 2011 Courses (SIGGRAPH '11)*.

Chris Bregler Lorenzo Torresani, Peggy Hackney. 2007. Learning Motion Style Synthesis from Perceptual Observations. In *Proc. of Neural Information Processing Systems (NIPS)*.

Mentar Mahmudi, Pawan Harish, Benoit Le Callennec, and Ronan Boulic. 2016. Artist-Oriented 3D Character Posing from 2D Strokes. *Computers and Graphics* (2016).

Megumi Masuda, Shohei Kato, and Hidenori Itoh. 2010. A Laban-Based Approach to Emotional Motion Rendering for Human-Robot Interaction. In *Entertainment Computing - ICEC 2010 (Lecture Notes in Computer Science)*. Springer, Berlin, Heidelberg.

Jianyuan Min and Jinxiang Chai. 2012. Motion Graphs++: A Compact Generative Model for Semantic Motion Analysis and Synthesis. *ACM Trans. Graph.* 31, 6 (Nov. 2012).

Michael Neff and Eugene Fiume. 2005. AER: Aesthetic Exploration and Refinement for Expressive Character Animation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '05)*. ACM.

Frank E. Pollick, Uri Maoz, Amir A. Handzel, Peter J. Giblin, Guillermo Sapiro, and Tamar Flash. 2009. Three-dimensional arm movements at constant equi-affine speed. *Cortex* 45, 3 (2009). Special Issue on Cognitive Neuroscience of Drawing.

F. E. Pollick and G. Sapiro. 1997. Constant affine velocity predicts the 1/3 power law of planar motion perception and generation. *Vision Research* 37, 3 (Feb. 1997).

C. Rose, M. F. Cohen, and B. Bodenheimer. 1998. Verbs and adverbs: multidimensional motion interpolation. *IEEE Computer Graphics and Applications* 18, 5 (Sept. 1998).

Alla Safonova and Jessica K. Hodgins. 2007. Construction and Optimal Search of Interpolated Motion Graphs. *ACM Trans. Graphics, Proceedings of Siggraph* 26, 3 (July 2007).

Abraham. Savitzky and M. J. E. Golay. 1964. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry* 36, 8 (July 1964).

Klaus U. Schulz and Stoyan Mihov. 2002. Fast string correction with Levenshtein automata. *International Journal of Document Analysis and Recognition* 5, 1 (2002).

Hyun Joon Shin, Jehee Lee, Sung Yong Shin, and Michael Gleicher. 2001. Computer Puppetry: An Importance-based Approach. *ACM Trans. Graph.* 20, 2 (April 2001).

David J. Sturman. 1998. Computer Puppetry. *IEEE Comput. Graph. Appl.* 18, 1 (Jan. 1998).

S. C. L. Terra and R. A. Metoyer. 2004. Performance Timing for Keyframe Animation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '04)*.

Matthew Thorne, David Burke, and Michiel van de Panne. 2004. Motion Doodles: An Interface for Sketching Character Motion *(SIGGRAPH '04)*. ACM.

Shihong Xia, Congyi Wang, Jinxiang Chai, and Jessica Hodgins. 2015. Realtime Style Transfer for Unlabeled Heterogeneous Human Motion. *ACM Trans. Graph.* 34, 4 (July 2015).

M. Ersin Yumer and Niloy J. Mitra. 2016. Spectral Style Transfer for Human Motion Between Independent Actions. *ACM Trans. Graph.* 35, 4 (July 2016).

Hui Zhang, Yuhao Song, Zhuo Chen, Ji Cai, and Ke Lu. 2012. Chinese Shadow Puppetry with an Interactive Interface Using the Kinect Sensor. In *Proceedings of the 12th International Conference on Computer Vision - Volume Part I (ECCV'12)*. Springer-Verlag.