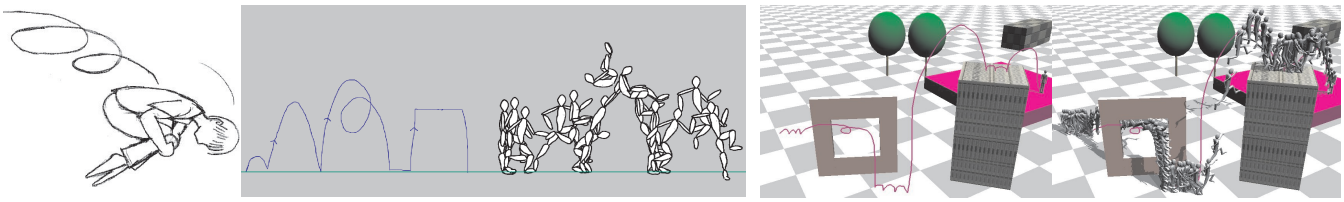
Motion Doodles: An Interface for Sketching Character Motion

Matthew Thorne David Burke Michiel van de Panne

University of British Columbia*



(a) (b) (c)

Figure 1: (a) Motion lines as used in a drawing (b) 2D motion sketch and the resulting animation (step, leap, front-flip, shuffle, hop) (c) 3D motion sketch and resulting animation (walk, flip through window, walk, leap onto building, walk, leap off building)

Abstract

In this paper we present a novel system for sketching the motion of a character. The process begins by sketching a character to be animated. An animated motion is then created for the character by drawing a continuous sequence of lines, arcs, and loops. These are parsed and mapped to a parameterized set of output motions that further reflect the location and timing of the input sketch. The current system supports a repertoire of 18 different types of motions in 2D and a subset of these in 3D. The system is unique in its use of a cursive motion specification, its ability to allow for fast experimentation, and its ease of use for non-experts.

Keywords: Animation, Sketching, Gestural Interfaces, Computer Puppetry

1 Introduction

Animation has existed as an artform for approximately 80 years and the technology used to create animations has evolved tremendously. Unfortunately, animation tools usable by non-experts remain in short supply. In this paper we develop a cursive motion notation that can be used to "draw" motions. We present an interactive animation system that interprets the notation as it is drawn. The system is simple enough to be usable by novice animators, including children. With an appropriate tailoring of the motion vocabulary, we expect that motion sketching systems may find applications in film storyboarding, theatre staging, choreography for dance and sports, and interactive games.

*email: mthorne,dburke,van@cs.ubc.ca

1.1 Overview

What does it mean to "sketch a motion" for a character? We propose one possible answer to this question, inspired in part by motion illustration techniques, such as the use of loops to indicate a tumbling motion as shown in Figure 1(a).

To begin using our system, the user draws the character they wish to animate. This is accomplished by drawing the body, head, arms, legs, and feet, from which a character armature is inferred. Animations can then be created by drawing motion sketches, which are interpreted on the fly to yield interactive animated motions. Figures 1(b) shows an example 2D motion sketch and the resulting animation. A subset of the motion sketch gestures can also be drawn on top of a 3D image to obtain a 3D character animation, as illustrated in Figure 1(c). Significantly, our gestures are highly visual in nature and thus serve both as a means of motion control and as a meaningful visual record (notation) of the motion.

A block diagram of the system is given in Figure 2. The path from sketching a motion to producing the motion itself is implemented as a pipeline in order to allow for animated motion to be produced while the motion sketch is still ongoing. The drawn sketch undergoes a multi-stage segmentation process to extract recognizable motion primitives from the input motion sketch. Having identified one or more motions, multiple parameters are extracted from the corresponding portion of sketch, including the start and end points, timing information, and possibly several other features. These parameters are passed on to an animation back-end, which in the current system is based on parameterized keyframe motions.

1.2 Contributions

The primary contributions of this paper are as follows:

- We present the design of a set of continuous (cursive) gestures for sketching a significant variety of motions, their locations, and their timing. These gestures are implemented in a sketch-based animation system and are demonstrated on a variety of display devices.

- We present a system that allows novices to sketch a 2D character and then draw a variety of animated motion for it, all within tens of seconds.

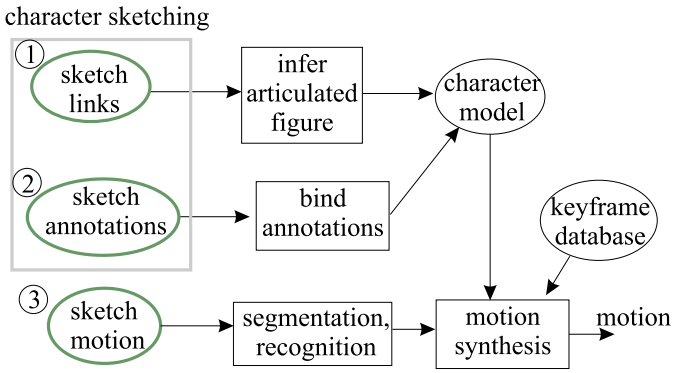


Figure 2: The sketching system.

2 Previous Work

The use of sketching in computer graphics dates back to the seminal SketchPad system[Sutherland 1963]. More recently, the sketch-based modeling systems SKETCH[Zeleznik et al. 1996] and Teddy[Igarashi et al. 1999] are inspired examples of how sketches or drawn gestures can provide a powerful interface for fast geometric modeling. The notion of "sketching a motion" is less well-defined than that of sketching an object. Nevertheless, a number of approaches have been explored. Early work explores an animation-by-example approach for a single point and fits splines to an input stream of 3D points in order to produce a smoothed version of the acted trajectory[Balaguer and Gobbetti 1995]. More recently, a considerably more sophisticated type of animation-by-example approach has been proposed[Popovic' et al. 2003], wherein the trajectory (position and orientation over time) of a rigid body can be specified by example using a 3D tracker and then "cleaned up" automatically to synthesize the physically-based motion that best fits the sketched motion. The system we propose is significantly different in that we focus on how 2D stylus input can be used to drive stylized 2D and 3D character motion.

Walking motions can be easily created by drawing a desired path on the ground plane for the character to follow, with the drawing also possibly governing the walking or running speed. Given the path and path timing, the character motion can be implemented in many different ways[Arikan and Forsyth 2002; Girard 1987; Kovar et al. 2002; Park et al. 2002; van de Panne 1997]. Our system supports many types of motion other than walking and allows for control over additional motion parameters.

Video game interfaces are perhaps one of the most readily-accessible forms of animation; a joystick and buttons provide the ability to control the direction and speed of the character as well as other sets of context-dependent predefined motions. However, game interfaces cannot fully replicate the control offered by a cursive device such as a stylus – consider for example the difficult task of cursively writing one's name with a joystick. Our system exploits a user's skills at cursive drawing and offers a greater degree of control over motions than many game interfaces. In the case of walking, for example, we offer simultaneous interactive control over step length, step height, step style, and step time.

A number of character animation systems use some form of acting as their interface. The spectrum of techniques here includes full-body acting, as used in motion capture or performance animation; the use of body silhouettes[Lee et al. 2002], various forms of computer puppetry[Sturman 1998; Oore et al. 2002; Laszlo et al. 2000], and systems that can infer a desired mapping from an actor or animator to a character using a two-pass imitate-then-modify process[Dontcheva et al. 2003]. The 2D stylus-based input of our system differs significantly from the above systems in that it aims to exploit drawing skills and not acting skills. The drawn input of our system also serves as a static visual record of the motion, something that is not available for acting-based interfaces. Labanotation[Hutchinson and Balanchine 1987] is an example of a written motion notation system for dance choreography, which can be automatically translated into 3D human figure animations[Wilke et al. 2003]. We aim for a written notation system that is much easier to learn and use, foregoing much of the detailed control that a general motion notation system can provide.

More distantly-related previous work looks at creating animations from sequentially-drawn sketches of a character, somewhat like traditionally-drawn keyframes. With appropriate constraints, a 3D character pose can be inferred for each hand-drawn frame[Davis et al. 2003].

3 Character Sketching

The two core components of our animation system are a character sketching tool and a motion sketching tool, as shown in Figure 2. We first discuss the character sketching tool, which consists of sketching the skeletal links comprising the basic character shape, followed by the optional addition of drawn annotations.

3.1 Sketching the Skeleton

A character sketch begins with drawing the links that will represent the character's articulations and basic shape. The system assumes that this is sketched in a side view using a total of 7 links, one for each of the head, torso, upper arm, lower arm, upper leg, lower leg, and foot. Each link is drawn using one continuous stroke, and the links can be drawn in any order. Links may or may not intersect when they are drawn and they may or may not contain some surface detail, such as adding in a sketched thumb, pot-belly, or nose. Figure 3(a) shows an example sketch.

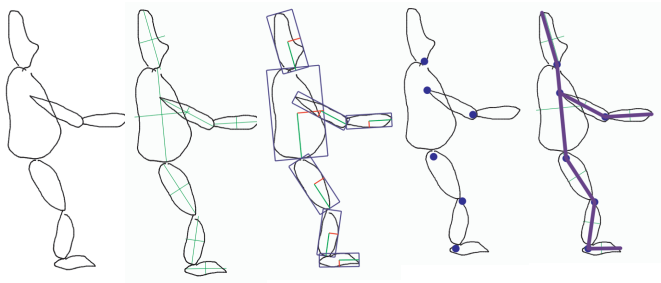


Figure 3: The process of inferring the skeleton from the sketch. (a) The seven sketched links. (b) Computed major and minor axes. (c) Oriented bounding boxes. (d) Computed joint locations. (e) Computed skeleton.

Once skeletal links have been drawn, the system automatically infers the locations of the joints, labels the links, and creates the second arm and leg. Recognizing the human form is addressed in numerous ways in the computer vision literature, but we are solving a simpler problem, one that benefits from additional constraints. Individual links do not need to be identified – each recorded stroke is already known to be a link. Also, the expected connectivity of the links is known in advance. Users can sketch the character in a wide range of initial configurations, as shown in Figure 5.

The pseudocode for inferring the skeletal structure from the sketched links is given in Figure 4. Once all seven links have

been sketched, the principal axes of each link are computed as shown in Figure 3(b). Each sketched link outline is treated as a series of n points P_i, and the principal axes are computed by fitting the points to an anisotropic Gaussian distribution. If M is the mean of the points, the major and minor axes of the box are chosen as the unit-length eigenvectors U_j of the covariance matrix $\Sigma = \frac{1}{n}\sum_i (P_i - M)(P_i - M)'$. Σ is tridiagonal in 2D, so the QL algorithm with implicit shifting can be used to solve the eigenproblem[Press et al. 1992]. Next, the points are projected onto the axes to find the intervals of projection $[a_j, b_j]$ along those axes, in other words $a_j = min_i |U_j \cdot (P_i - M)|$ and $b_j = max_i |U_j \cdot (P_i - M)|$. Finally, an oriented bounding box is computed from the intervals of projection, centered at $C_{box} = M + \sum_j \frac{a_j + b_j}{2} U_j$, where $\frac{a_j + b_j}{2}$ are the extents along each axis.

1. Wait for seven links to be sketched
2. Fit oriented bounding boxes to all links
3. For each link i
4. For each major-axis end-point on link i, P_i^1 and P_i^2:
5. Search all links $j \neq i$, for the closest point, P_j
6. If links i and j are not aligned
7. create joint J_n at intersection of major axes of i and j
8. else
9. create joint J_n at midpoint of $P_i P_j$
10. Identify and remove all duplicate joints
11. Identify links based on connectivity
12. Create duplicate arm and leg segments.

Figure 4: Algorithm for inferring the skeleton from the sketch.

The next step is to determine the connectivity of the links and the locations of the resulting joints. For this, a closest link is defined for each major-axis endpoint, measured in terms of the minimal Euclidean distance from the major-axis endpoint to any point on another link. Once link j has been identified as being the closest link for a major-axis endpoint on link i, a joint is created at the geometric intersection of the extensions of the major axes of links i and j. However, this will not produce a sensible joint location if links i and j are nearly parallel. If the major axes are within 20° of being parallel, the mid-point of the line segment connecting the major axis end-points of i and j is used. This joint-creation process will result the creation of duplicate joints, such as a second 'ankle' joint being created when processing the major-axis endpoint at the toe of the foot. These duplicates are trivially removed.

Once the joints and their associated links are known, we resort to the expected topology of a human figure in order to label all the links as being the head, the torso, etc. The torso is identified as being the only link having 3 associated joints. The head link is identified as being attached to the torso and having no further joints. The arms and the legs are similarly identified by their unique connectivity. If the identification process fails, this is reported to the user. The bones for the underlying skeleton are finally constructed by connecting the appropriate joints with straight line segments. The sketched links are then redefined in the local coordinate frame of the resulting bones. The default reference pose used to start all animations is given by a standing posture that has all bones being vertical and the feet being horizontal. Figure 5 is illustrative of the variety of skeleton sketches that the system can recognize.

There are two additional joints internal to the torso that are not shown in the figures. They represent bending at the waist and the upper back, and are added to facilitate tucking during the forward and backwards somersault motions. The joints are located at fixed fractions along the torso bone. A joint is also automatically added at the ball of the foot in a similar fashion.

The current algorithm used for inferring the skeleton will fail if

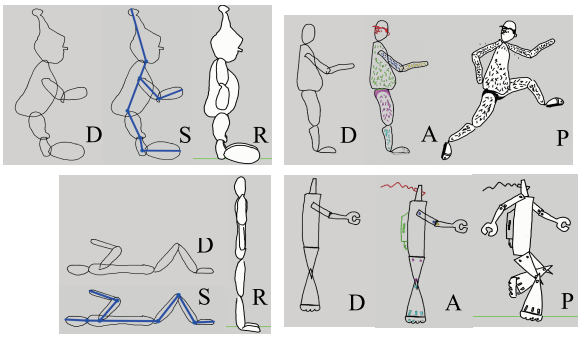


Figure 5: A variety of skeleton sketches and their inferred skeleton. *D*: the original drawing; *S*: inferred joints and the fitted skeleton; *R*: character in the reference pose; *A*: drawn annotations; *P*: an animated pose.

the arms are sketched in a downwards pose parallel to the torso, or if the character is sketched in a pose such that the hands are located close to the head, knees, or feet. These types of malformed sketches or erroneous link assignments could be addressed with some additional sophistication in the skeleton recognition algorithm. In practise the algorithm is quite robust in its current form. The system does not currently allow for the user to refine the skeleton after it is constructed by the system.

3.2 Adding Annotations

The system allows the user to add annotations which serve to decorate the links of the character. Thus, one can sketch additional features such as eyes, ears, hands, hair, a hat, a nose, and shoes. All annotations automatically become associated with the closest link. In our current version, this will result in annotations that break, such as for a sleeve that crosses multiple links. There are a number of known skinning techniques for addressing this problem, although these have not yet been implemented in the current system.

4 Motion Sketching

Sketching a motion for a character requires a degree of abstraction not present when sketching geometry. The motion sketch needs to convey a significant amount of information: (1) the type of motion; (2) the spatial location and extent of the motion; and (3) the timing of the motion. In this section we describe the design of the gestures for the 2D system, how the gestures are segmented and recognized, how the output animation is generated, and, lastly, how the 3D system works.

4.1 A Cursive Alphabet for Character Motions

Our gesture vocabulary was designed with the following principles in mind: (1) The motion gestures should be cursive, thus allowing the specification of one motion to smoothly flow into the specification of the next motion; (2) Given the limited number of very-easy-to-draw gestures that are available, the effort to draw the gesture should reflect the effort required to produce the corresponding motion. Thus, a regular walk should be easier to draw than a stiff-legged walk, which itself is easier to draw than a one-legged hop. (3) The gesture should be reminiscent of the corresponding motion, to the extent this is possible; (4) Gestures related to locomotion should allow for forwards and backwards motions; (5) Similar motions should have similar gestures; (6) Gestures should allow for the generation of stylistic variations where possible.

Guided by these principles, we developed the gesture vocabulary shown in Figure 6. These 18 motions gestures (31 when allowing for backwards-traveling variants) all allow for control over the timing of the motion by having the animated motions directly reflect the time taken to draw the gestures. They also allow for control over the start and end points of the motions, and all but two of the motions provide control over a height parameter, either the height of the swing foot during a step or the height of the body center of mass during a jump.

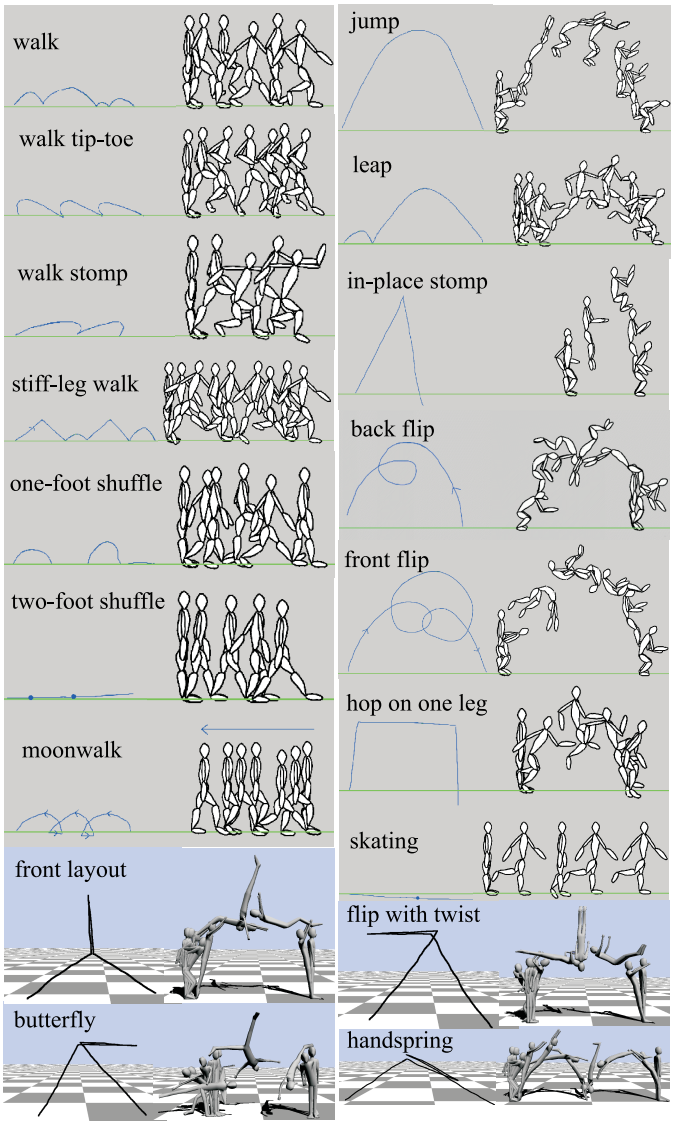


Figure 6: Gesture vocabulary.

The drawn gestures do not typically act as a direct representation of the motion of any particular part of the body. For example, the drawn arcs used for walking and its variations are evocative of the path taken by the swing foot, but are not an accurate representation of this motion. The path of the real swing foot begins from the previous foot-fall location, not the current one. Similarly, the arc drawn to represent a jump represents the location of the feet at the start and end of the arc, while the middle of the arc represents an approximate trajectory for the center of mass.

Because walks and jumps are both specified using arcs, they are distinguished by the height of the sketched arc. A "jump line" is overlaid onto the scene during motion sketching and represents the maximum arc height that is treated as a walking step. Arcs that pass over this line are treated as jumps or leaps. Additional details regarding the parsing and recognition of the input gestures are provided in the next section.

For motions such as a jump with a twist, it is difficult to find a 2D drawing gesture that is evocative of what is fundamentally a 3D motion. Our system recognizes a class of more abstract gestures in order to support such motions. The four motions appearing at the bottom of Figure 6 shows this class of gestures being employed to control various gymnastic motions.

4.2 Sketch Segmentation

The input sketch is processed in multiple stages. The first *tokenization* stage consists of taking a stream of input points from the stylus and producing a corresponding list of tokens. Figure 7(top) shows the six types of tokens that are output by this first stage and Figure 7(middle) shows an example input sketch which has been labelled using these tokens. Once the sketch input has been tokenized, a *parsing* stage is used to identify the set of admissable gestures, as shown in Figure 7(bottom). Lastly, the *motion identification* stage identifies the specific motions to be generated. Segmenting and recognizing gestures based on a regular expression grammar has a number of precedents, a good example being the framework set out in [Hammond and Davis 2003]. We now describe the segmentation steps in additional detail.

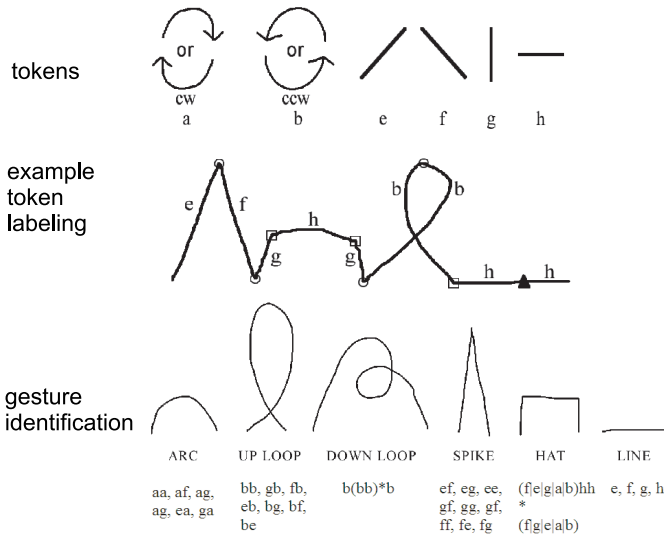


Figure 7: Segmenting the motion sketch input.

The tokenization stage processes a sequence of time-stamped input points in six steps, as shown in Figure 8. In step one, the input points are segmented based upon changes in the vertical direction of motion (thus discerning between rising and descending strokes). Step two applies a simple corner detection algorithm[Chetverikov and Szab 1999]. However, corners may be falsely identified for quickly drawn loops and arcs on slow input devices, resulting in curves being represented by only a few sparsely-spaced points. If the stylus velocity as measured by finite differences at a corner point exceeds 35% of the maximum stylus velocity, then the point is no longer regarded as a corner point. Step four classifies segments as being either straight or curved. A straight segment is defined as having $r < 1.2$, where r is the ratio of the arc length to the geometric distance between segment endpoints. Also in this step, colin-

ear neighboring straight segments are merged. Step five adds segmentation points at locations where the stylus has paused, which are important for motions such as shuffles and skates. Lastly, step six assigns one of six tokens to each resulting segment based upon whether the segment is straight or curved, and, if straight, based upon the absolute angle of the line segments. Straight lines that make an angle of less than 30 degrees with the vertical or horizontal are assigned tokens g and h, respectively; otherwise they are assigned the token e or f, whichever is closer in terms of angle.

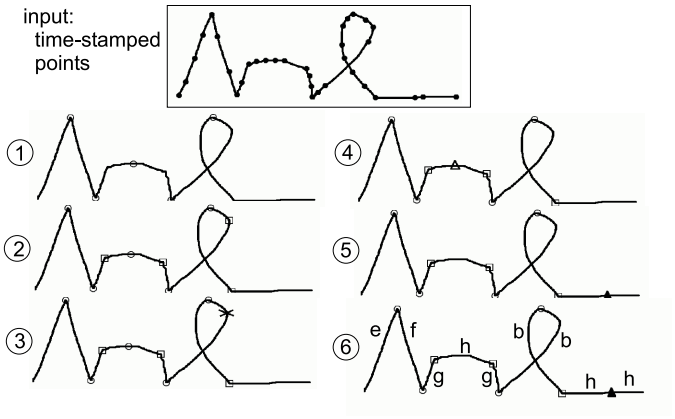


Figure 8: Tokenization consists of six steps: (1) Rough segmentation based on vertical direction of motion; (2) Corner detection; (3) Corner post-processing; (4) Merge colinear segments; (5) Identify pauses; (6) Token assignment.

Given an input sketch that has been labelled with tokens, the parsing stage groups tokens together into the set of admissable gestures shown in Figure 7(bottom). This is accomplished by matching the regular expressions corresponding to each gesture type. Lastly, the motion identification stage determines the specific motion to be executed. In some cases, the identified gestures will map directly to particular motions. For example, the gesture for a one-legged hop has a unique interpretation. In other cases, additional criteria are examined in order to disambiguate the desired motion.

In order to further distinguish between a walking step and a jump, the maximum height of the arc is used. An arc of height $h > h_{walk}$ is determined to be a jump or leap; all others are interpreted as some type of walking step. In our current implementation, a horizontal line is drawn at $y = h_{walk}$, thereby providing the animator with an easy point of reference while sketching.

An additional criteria is employed to avoid impossibly-large walking steps. Any arc of length greater than a maximum step length $d_{maxstep}$ is interpreted as a jump rather than a step. This avoids the ill-posed inverse-kinematics problems that would otherwise arise in performing such steps. Similarly, if an arc passes above $y = h_{walk}$ but does not allow sufficient ground clearance for the jump to be completed due to the character's geometry, a vertical offset is computed for the apex, allowing for a feasible jump.

Tip-toe and stomp walking steps are distinguished from regular walking steps by examining the relative position of the apex of the sketched arc with respect to its endpoints, as measured by $\alpha = (x_{apex} - x_{start})/(x_{end} - x_{start})$. A tip-toe step is identified for $\alpha < 0.35$ and stomp-step for $\alpha > 0.65$. At present, the system performs a discrete classification of each step as being a regular, tip-toe, or stomp step. A final ambiguity exists between shuffle steps and a slides, which are both represented by horizontal line segments. A shuffle step is assumed if the length of the step is less than $d_{maxstep}$ and otherwise becomes a slide.

If during a sketch the pen remains stationary for more than $0.5s$, the character is brought to a standing posture. A standing long jump occurs as a result of a sketched jump arc whenever both feet are together. If this is not the case, such as when a walking step is followed by a jump arc, the jump is classified as a leap.

Because the user's sketch is always ahead of the motion segmentation and synthesis, successfully segmented motion actions are stored in a queue for processing. It is possible that the motion queue is exhausted while the next gesture is still in the process of being drawn, in which case a pause is introduced into the output motion. This pause exists only as an artifact during the original sketching process and disappears during a motion replay.

Animators can also sketch motions directly in an environment with other animated objects. This allows for motions that need to be coordinated with existing animated objects, such as executing a leap over a falling character or jumping out of the way of a car. Because we use the time at which a motion sketch was drawn as the reference timing for the motion, coordination with existing animated motion is easily accomodated. The character animation produced during an initial 'live' sketch may not be properly synchronized with the action because of the necessary delay in recognizing gestures. However, a motion replay produces the correctly timed result.

4.3 Output Motion Synthesis

Once an input gesture has been appropriately identified and mapped to a particular motion, e.g., "single back flip", the key parameters for that motion segment are extracted, and the output motion synthesis can begin. Common parameters to all motions include the start and end positions, as well as the motion duration. Most other motions also extract a parameter relating to the location and timing of the apex of the sketched input gesture. For example, jumps and leaps are parameterized in terms of duration of ascent, duration of descent, the maximum height of the jump, and the start and end locations. Front and back flips have additional parameters describing the number and direction of rotations. The walk, walk-stomp, tip-toe, stiff-leg walk, hop, and one-foot shuffle all use the arc height parameter to control the height of the swing leg during stepping.

Once the type of motion and the related parameters are known, the desired motion could be synthesized in one of several ways. We choose to employ a parameterized keyframe-based motion synthesis technique. Motion-capture-retargeting techniques or space-time optimization techniques could also be considered, but the kind of physical realism obtained with these techniques is not one of our goals, nor are they necessarily appropriate for a system which encourages experimentation with cartoon-like super-human motions.

Each type of motion is implemented by breaking it into a fixed number of stages and then applying a number of tools: a keyframe database, a keyframe interpolator, an inverse-kinematics solver, and a means to position the center-of-mass at a specified point. As an example, the stages used to implement jumps are shown in Figure 9. A detailed description of the stages used for all motions may be found in [Thorne 2003].

The duration of each stage is determined as a fixed fraction of the input-sketch times associated with the motion. For example, a jump motion has both ascent and descent times, which are determined directly from the input sketch. The first three stages of a jump motion all determine their duration as a fixed fraction of the input-sketch ascent time. The durations of the remaining three stages correspond to fixed fractions of the input-sketch descent time.

All stages of any particular motion have an associated keyframe that defines target joint angles to be reached by the character at the end of the stage. A Catmull-Rom interpolant is used between successive keyframes. The global position of the character is controlled separately from the keyframes. For steps, the root of the character (located at the hip) is placed halfway between the known positions of the stance and swing feet. For the airborne phases of

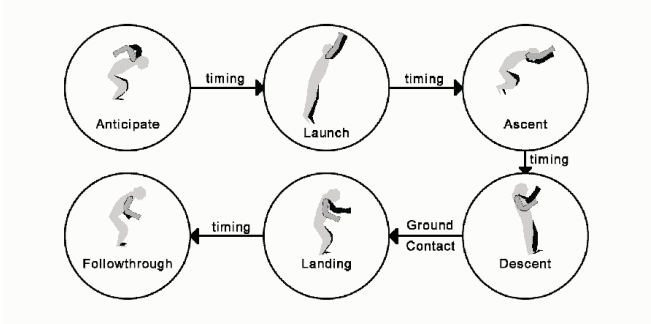


Figure 9: States and keyframes used for jumps.

jumps, flips, and leaps, the center-of-mass is directly placed at an appropriate position as determined by two parabolic center-of-mass curves, one for ascent and one for descent. These curves are fitted based upon the start, apex, and end locations of these motions.

Once the keyframes have been interpolated to obtain the pose and the skeleton has been positioned, some stage-specific modifications are implemented. These can perform a number of functions, one example being the modifications required to preserve continuity of the center-of-mass velocity upon jump landings, as required when the center-of-mass transitions from being controlled by the descending parabolic arc to being controlled by the landing-stage keyframes. Inverse kinematics is applied to the legs for all animation frames involving ground contact, such as landing and follow-through for the jump, or stance during walking.

A key issue in designing motions is making them robust to variations in the proportions of the character being animated. Thus, a character with a short torso and long legs will potentially move very differently than a character with a long torso and short legs. At present we deal with this issue primarily through the appropriate use of inverse kinematics during all ground contact phases. Also, the length of the largest possible step, $d_{maxstep}$, is dependent on the character's leg length. The remaining aspects of the motion in our current system are independent of the character proportions, being driven purely by joint angles. While generally robust, characters with extreme proportions will occasionally exhibit problems with body parts passing through the ground.

4.4 Sketching in 3D environments

The basic mechanisms used in the 2D system can be extended to work in a 3D setting, albeit with some caveats. The addition of an extra dimension introduces ambiguities into the interpretation of the 2D input sketch. A motion sketch for a 3D environment begins by positioning the camera such that it covers the desired workspace in its field of view. The sketch is then drawn directly overtop of the image produced by this fixed point of view, as shown in Figure 13. One can also sketch motion on top of a photograph by creating proxy geometry for the objects in the photo, as seen in Figure 14. The 3D character for our system is modelled in advance and not produced from a 2D sketch.

The sketching of walks, jumps, leaps, and flips can be mapped to a 3D environment in a relatively straightforward manner. The sketch is processed to find the start and end points of each gesture, as indicated by a change in the vertical direction of motion and the satisfaction of the corner metric. The identified 2D sketch segmentation points are then back-projected into the 3D scene in order to locate them in 3D. Given known 3D locations of the start and end points, the remaining 2D sketch points are back-projected onto the vertical plane V that embeds the 3D start and end points. The sketch points can now be processed in the 2D coordinate frame of V as with the 2D system. In this way, a proper apex for the motions can be extracted for arcs that are drawn "in perspective".

The above mapping process still results in a number of limitations. Motions moving directly towards the camera or away from the camera remain difficult to sketch. Additionally, some gestures become ambiguous when mapped to 3D, as shown in Figure 10. In-place stomps and stiff-legged walks become confused with the shuffle and skating motions, given that all these gestures are drawn with straight lines. There is a further ambiguity involving the direction the character is facing; the gestures for a forward-step and backwards-step cannot be distinguished in the 3D environment. Our system makes the particular assumptions shown in Figure 10 in order to resolve these ambiguities. Other assumptions, modes, use of context, or additional gestures could equivalently be used to help with disambiguation.

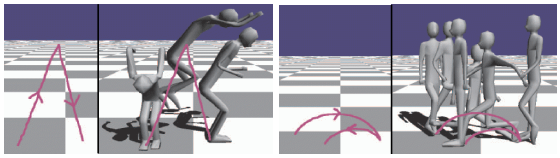


Figure 10: (a) A gesture which can be interpreted as two slides or an in-place stomp, resolved in favor of the stomp. (b) Ambiguity regarding the facing direction of the character is resolved by assuming that the character walks forwards.

Synthesizing motion in 3D occurs much as it does in the 2D case – the same features are extracted from the sketch as with the 2D system. A keyframe database serves as a back-end to fill in details of the motion that are not provided directly from the sketch or through inverse kinematics. Our prototype 3D system largely uses the same set of keyframes as in the 2D system.

There are two aspects in which 3D motion synthesis differs from its 2D counterpart: foot placement and direction smoothing. In the 2D system, the start and end point of each gesture marks where the foot (or feet) strikes the ground. In 3D, the feet are offset from the vertical plane embedding the sketch in order to allow for distinct left and right foot placements.

The facing direction of a character does not change in the 2D system but it may do so in the 3D system. A vector from the start point to the end point of a gesture is used to define the character's direction of travel. In order to have this direction change smoothly, some form of smoothing is required. Our system interpolates the heading direction over specific stages of each motion. For walking motions, this gives the character the appearance of rotating on its heel. For brevity, we refer the reader to [Thorne 2003] for a detailed explanation of this process.

5 Results and Discussion

Numerous interactive demonstrations of the system are shown in the video that accompanies this paper. Sketching both a character and a motion of the type shown in Figure 1(b) is easily done in under thirty seconds. The system can also be used to draw and animate *mech-bots*, as shown in Figure 11. These are drawn using 5 links instead of 7 and are animated using the same motion data base, except for the reverse knee bend direction and the lack of arms.

More abstract gestures allow for the system to animate wider classes of motions. Figure 12 shows the sketch of a gymnastics tumbling sequence. As with all motion gestures, the motion type, height, distance, and timing of the individual motions are derived from the motion sketch.

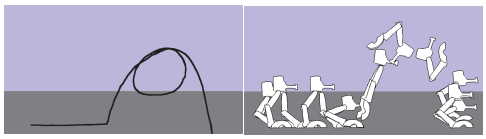


Figure 11: Animating a mech-walker: Shuffle walk and front flip.

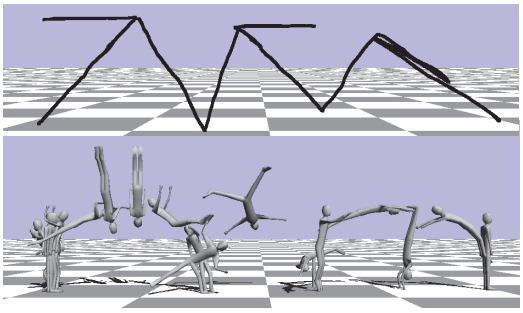


Figure 12: Sketching a gymnastics tumbling sequence: flip-with-twist, butterfly, and front-handspring.

The system is well suited for use with a variety of input devices. We have most commonly used it with a mouse during development, but other devices are more suited to producing fast and accurate gestures. Figure 15 shows the system in use on an electronic whiteboard and a Tablet PC. The SMARTboard allows the user to directly draw the desired motion sketch on top of the scene using a finger, and similarly for the Tablet PC with a stylus. Children found the system significantly easier to use with these direct input devices.

5.1 Uses of the system

The animation system presented here is not a substitute for the array of professional animation tools and techniques that are commonly used in film and games. Instead, the motion sketching system presents an alternate and highly-accesible means for users to create a certain class of character animations. The target class of motions should be tailored with the application in mind. The motions and gestures implemented in our prototype have been chosen to illustrate the motions that one might want for a storyboarding (variety of locomotion and jumping), for gymnastics choreography, and for motions that are illustrative of what can be done (moonwalk, flips).

There exist a number of commercial and research animation systems that are capable of synthesizing motions from a variety constraints. The goal of a cursive motion specification language is to make the specification of the constraints and timing a more transparent process – the users need not be fully aware of the specific parameters that drive the motion synthesis process. Unlike acting-based interfaces, the motion gestures provide a meaningful visual record of the motion, as well as allowing for "superhuman" unphysical and exaggerated motions.

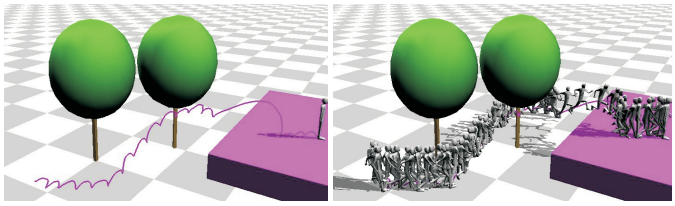


Figure 13: Walking and leaping around a set of trees.

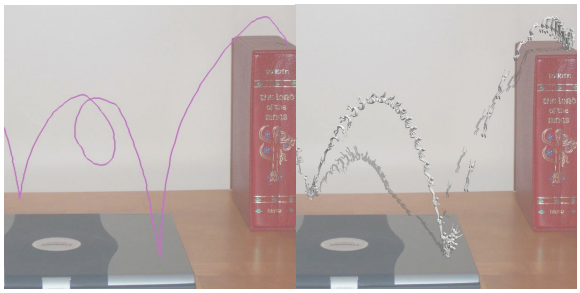


Figure 14: Stunts for a miniature character that have been sketched on top of an image with modelled 3D proxy geometry.

Figure 15: Sketching motions on a SMARTboard and a Tablet PC.

Over fifty people of all ages have used the system, including children as young as three. A number of anecdotal observations were made. Users rapidly learned the gesture vocabulary and enjoyed experimenting with the system in many ways. The gesture identification was occasionally problematic, with some gestures being interpreted in a fashion that did not reflect the user's intentions. We intend to further improve the robustness of the gesture recognition in order to address this issue.

Young children greatly enjoyed experimenting with the motion sketching, but found it difficult to understand the restrictions imposed on the character sketching, namely the use of seven links representing a side view of the human figure and the fact that the principal seven links had to be drawn before annotations could be added. Perhaps unsurprisingly, children would also put the system fully to the test by inevitably drawing motion gestures which had no meaningful interpretation. "Garbage in, garbage out" describes the behavior of the system in such circumstances.

Adults enjoyed the ability to sketch a character and then be immediately able to animate it. While we have conceived of the system as principally targeting novice users, an accomplished animator remarked that the system provided almost instantaneous satisfaction because of the immediacy of the animated results, something he felt was missing from present-day animation tools.

5.2 Scalability

Adding a new motion to the system requires the creation of a new gesture that can be identified by a novel sequence of tokens, as well as the implementation of an appropriate sequence of states and keyframes that is capable of generating parameterized versions of the desired motion. In the future we wish to add gestures for a variety of falling motions as well as interaction with the environment. The system can potentially be made more scalable through the use of 3D input devices and the improved use of context in specifying motions. Such additional controllability would come at the expense of increased complexity of the interface.

5.3 Limitations

The current system has a number of limitations. The system does not support the complete gesture vocabulary in 3D due to the ambiguities introduced by the 3D mapping. These can be overcome in part by making some motions context-specific or by introducing user-specified modes to resolve such ambiguities. For example, in a figure-skating context, drawn loops are probably better reserved for spins around the vertical axis rather than the head-over-heels flips that the current system is capable of.

The system is not suitable for animation that requires unique or detailed motions. A partial solution would be to animate a character in a series of successive passes with a motion layering approach[Sturman 1998; Oore et al. 2002; Dontcheva et al. 2003].

A possible improvement on our sketch segmentation scheme would be to develop a system that can be "trained by example" to recognize specific desired sets of gestures[Rubine 1991]. However, the work presented in [Rubine 1991] is not directly applicable to our problem domain because of the cursive nature of our gestures; one stroke represents a compound sequence of parameterized (and therefore variable) gestures instead of a single gesture.

6 Summary

As kinematic and dynamic methods for synthesizing motions from constraints become increasingly mature, the specification of constraints becomes a bottleneck, particularly to novice animators. We have presented a cursive language for sketching 2D and 3D character motions. This is implemented in a system that allows novices to quickly learn to sketch-and-animate a human or "mech-bot" character of their own design in tens of seconds. The technique is well-suited to take advantage of Tablet PCs and electronic whiteboards. The system is simple enough for children to use, and has other potential applications in storyboarding and the choreography of athletic routines. It is our hope that this method and its future variations play a role in making animation a more accessible media.

Acknowledgements

We would like to thank the anonymous Siggraph reviewers for their comments on earlier versions of the paper. This research was supported by NSERC post-graduate scholarships and an NSERC Discovery Grant.

References

ARIKAN, O., AND FORSYTH, D. A. 2002. Interactive motion generation from examples. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (July), 483–490.

BALAGUER, J., AND GOBBETTI, E. 1995. Sketching 3D animations. *Computer Graphics Forum 14*, 3, 241–258.

CHETVERIKOV, D., AND SZAB, Z. 1999. A simple and efficient algorithm for detection of high curvature points in planar curves. *Proc. 23rd Workshop of the Austrian Pattern Recognition Group*, 175–184.

DAVIS, J., AGRAWALA, M., CHUANG, E., POPOVIC', Z., AND SALESIN, D. 2003. A sketching interface for articulated figure animation. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 320–328.

DONTCHEVA, M., YNGVE, G., AND POPOVIĆ, Z. 2003. Layered acting for character animation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) 22*, 3 (July), 409–416.

GIRARD, M. 1987. Interactive design of computer-animated legged animal motion. *IEEE Computer Graphics and Applications 7*, 6, 39–51.

HAMMOND, T., AND DAVIS, R. 2003. Ladder: A language to describe drawing, display, and editing in sketch recognition. *Proceedings of IJCAI 2003*.

HUTCHINSON, A., AND BALANCHINE, G. 1987. *Labanotation: The System of Analyzing and Recording Movement*. Theatre Arts Books.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: A sketching interface for 3d freeform design. In *Proceedings of SIGGRAPH 99*, 409–416.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (July), 473–482.

LASZLO, J., VAN DE PANNE, M., AND FIUME, E. L. 2000. Interactive control for physically-based animation. In *Proceedings of ACM SIGGRAPH 2000*, 201–208.

LEE, J., CHAI, J., REITSMA, P. S. A., HODGINS, J. K., AND POLLARD, N. S. 2002. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (July), 491–500.

OORE, S., TERZOPOULOS, D., AND HINTON, G. 2002. A Desktop Input Device and Interface for Interactive 3D Character Animation. In *Proc. Graphics Interface*, 133–140.

PARK, S. I., SHIN, H. J., AND SHIN, S. Y. 2002. On-line locomotion generation based on motion blending. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM Press, 105–111.

POPOVIC', J., SEITZ, S. M., AND ERDMANN, M. 2003. Motion sketching for control of rigid-body simulations. *ACM Trans. Graph. 22*, 4, 1034–1054.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 1992. *Numerical Recipes in C*. Cambridge University Press.

RUBINE, D. 1991. Specifying gestures by example. *Computer Graphics, Proceedings of SIGGRAPH '91 25*, 4, 329–337.

STURMAN, D. J. 1998. Computer puppetry. *IEEE Computer Graphics and Applications 18*, 1 (Jan-Feb), 38–45.

SUTHERLAND, I. E. 1963. Sketchpad: A man-machine graphical communication system. In *Proceedings AFIPS Spring Joint Computer Conference*, vol. 23, 329–346.

THORNE, M. 2003. *Motion Doodles: A Sketch-based Interface for Character Animation*. Master's thesis, University of British Columbia. http://www.cs.ubc.ca/grads/resources/thesis/Nov03/Matthew_Thorne.pdf.

VAN DE PANNE, M. 1997. From footprints to animation. *Computer Graphics Forum 16*, 4, 211–224. ISSN 1067-7055.

WILKE, L., CALVERT, T., RYMAN, R., AND FOX, I. 2003. Animating the dance archives. In *4th International Symposium on Virtual Reality, Archeology, and Intelligent Cultural Heritage*.

ZELEZNIK, R. C., HERNDON, K. P., AND HUGHES, J. F. 1996. Sketch: An interface for sketching 3d scenes. In *Proceedings of SIGGRAPH '96*, 163–170.