Motion Doodles: A Sketching Interface for Character Animation

Matthew Thorne David Burke Michiel van de Panne

University of British Columbia*

April 17, 2003

Abstract

We present a novel system which allows a character to be sketched and animated within a minute and with a limited amount of training. The process involves two steps. First, a character is sketched by drawing the links representing the torso, head, arms, and legs. An articulated skeleton and mass distribution is then inferred from the sketched links. Further annotations can be added to the skeleton sketch and are automatically bound to the underlying links. Second, the desired animated motion is sketched using a series of arcs and loops that are interpreted as an appropriate sequence of steps, leaps, jumps, and flips. The motion synthesis process then ensures that the animated motion mirrors key features extracted from the input sketch. For example, the height, distance, and timing of an arc that represents a jump are all reflected in the resulting animated motion. The current prototype allows for walking, walking with a stomp, tip-toeing, leaps, jumps, in-place stomps, front flips, and back flips, including the backwards versions of all these motions.

1 Introduction

Computer animations can have a wide range of requirements. Sometimes the goal is to generate physically-realistic and natural motions; at other times the goal may be to produce physically-impossible exaggerated motions. For applications such as production animation, very fine control is demanded over the final motion. For others, such as games, motion has to be generated in real-time and thus the detail of control offered to the user is quite limited.

This paper presents an animation system that explores the "quick-and-dirty" end of the specturm. It allows for the modeling and animation of an articulated character in tens of seconds rather than tens of minutes. Changing the speed with which one can model and animate a character changes both how one animates and who can animate. It becomes much easier to experiment with multiple character designs and motions, something that is particularly useful in supporting the coarse-to-fine workflow most often used in creating an animation. Animation also becomes serious fun and somewhat addictive. There are also more subtle ways in which a fast tool changes how one works. As an anecdotal example of this, we often found ourselves redrawing and animating a character from scratch instead of the seemingly cumbersome effort of typing or selecting the filename of an existing sketched character model or animation. We further hope that our system will make animation accessible to a much wider audience, allowing even young children to easily create simple animations.

An overview of how the user interacts with our animation system is shown in Figure 1. A user sketches a desired skeleton, then sketches additional annotations and possibly an environment, and lastly sketches a series of continuous arcs and loops representative of the desired motion. Figure 2 shows how the system processes the user's input to ultimately synthesize an animation. We now review the character and motion sketching processes in more detail.

1.1 Character Sketching

The character sketching consists of several types of input, the first and most important one being the sketch of the skeleton. A skeleton is sketched by drawing the seven links comprising the head, torso, upper arm, lower arm, upper leg, lower leg, and foot of the character. The animation system then infers a set of joints that serve to define the articulated character and identifies each sketched link to be a particular body part, such as the head or torso.

Once the skeleton has been sketched and identified, a user may add a variety of annotations. Thus, we may desire that the character

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

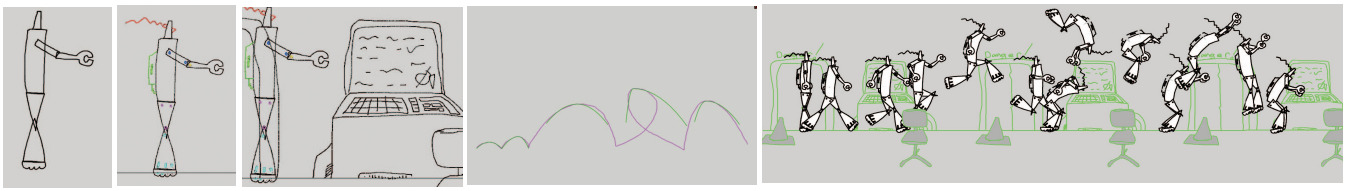


Figure 1: A sketched character animation. (a) Sketched skeleton links. (b) Sketched annotations. (c) Environment. (d) Sketched motion doodles. (e) Resulting animation.

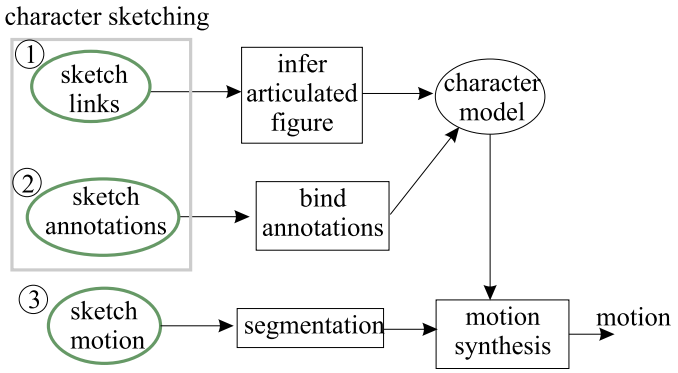


Figure 2: The sketching system.

have a hat on their head, carry an umbrella in their hand, wear glasses and shoes, or have hairy legs. These annotations can be sketched directly on top of the existing skeleton sketch and are automatically bound to the closest link of the underlying skeleton.

A foreground and background environment may also be sketched during the annotation process, as shown in Figure 1. A keypress is used to enable the foreground or background sketch modes, thereby ensuring that environment annotations are not accidentally bound to the links of the character. While the environment sketch generally consists of a passive background or foreground, props can be added which support secondary animation effects. Our currently implemented example of this involves stepping through water, which automatically invokes a particle system to generate appropriate splashing behavior. Other active props are possible, although not currently implemented.

It is worthwhile noting that the sketch is recorded with a stylus or mouse. The stroke information obtained this way allows the system to know when one stroke ends and another one starts, as well as the time it took to draw a given stroke. This same information cannot be extracted from a pencil-and-paper sketch which is then scanned in.

1.2 Motion Sketching

Once the character has been constructed, the time required to make an animation is given by the amount of time it takes to sketch the motion. The motion sketch makes use of what we call "motion doodles" — these are interactively sketched arcs and loops that are then interpreted to synthesize motions such as walking steps, jumps, and flips. Figure 1(d) and (e) show an example motion doodle and the resulting synthesized motion.

Motion doodles are reminiscent of several motion illustration techniques: arcs tracing the motion of an airborne object, arcs tracing the motion of a foot during walking, and the use of loops to indicate the tumbling motion during back-flips and front-flips. Motion doodles are in part abstract gestures, such as a front loop indicating a front flip. They are also in part kinematic, such as an arc designating a step and thereby specifying the path and target location of the step. The doodles additionally contain implicit timing and style information that are extracted and used in the motion synthesis process. For example, an asymmetric arc that represents a walking step will be interpreted as a tip-toe step if the peak of the arc is significantly to the left of the midpoint, and as a stomping step if the peak is significantly to the right of the midpoint.

The system allows for cartoon-style motions that are physically implausible, such as a character leaping over a building. Motions can be further stylized by altering the sets of keyframes that are behind the various motions.

2 Previous Work

Many media have tools suited for various levels of detail. A musical composition might begin with the creation of the melody for the refrain before being embellished in many ways. Automobile design often begins with a conceptual sketch. The workflow commonly used in designing animations also offers tools for working at various levels of details, often beginning with the use of storyboards and animatics[1], and then proceeding on to the use of keyframes for the detailed design of the motions. Bridging the gap between animatics and a well-designed keyframe motion remains a significant task, however. In many situations, it would be useful to have an animation system that allows for the most relevant features of a desired motion to be specified as input, with the remaining details being automatically filled in.

Many approaches have been proposed for this challenging problem. One class of methods focusses on animation as an optimization problem, typically employing some mix of hard and soft constraints, including constraints imposed by the physics of motion. Another class of methods leverages pre-existing motion capture data by modifying, interpolating, or stitching together motion sequences. Our animation system has a set of keyframes that play an important role in the motion synthesis and so it partly fits into this second class of techniques. However, the elements of a motion that are pre-existing and the elements that come directly from the user are very thoroughly blended in our case. For the remainder of this section, we provide a more detailed look at the techniques in graphics and animation that are most closely related to our system.

Geometric modeling systems that incorporate some notion of sketching have a long history that begins with the SketchPad system[Sut63]. More recently, the sketch-based modeling systems, SKETCH[ZHH96] and TEDDY[IMT99], are very inspiring examples of how sketches or drawn gestures can provide a powerful interface allowing for quick-and-dirty design. In a different vien, an algorithm for estimating a skeleton for closed polygonal models is presented in [TT98].

The use of demonstration to create animations has a long history that begins in 1969[Bae69]. More recent work includes the sketching of 3D animation paths[BG95], and many animation systems that allow the sketching of a walking path for a character to follow. With respect to locomotion, several algorithms allow for the specification of both foot locations and their timing in order to create animations[Gir87, vdP97]. Other techniques can be used to provide further sophistication in control over features such as step length[BC89], although this is done with a slider-based interface to set the parameters.

Similar goals of being able to "sketch" a desired motion were presented in [Pop01], wherein the trajectory (position and orientation over time) of a single rigid-body could be specified by example, and then "cleaned-up" automatically to synthesize the physically-based motion that best fits the sketched motion. Another system based on trajectory optimization allows for the motion of an articulated character to be defined in terms of a number of keyframes, some previous example motions, and some heuristics that help shape the optimization problem[LP02].

The notion of sketching an animation can also be thought of as a type of performance animation or puppetry, wherein an animator's gestures, drawn or otherwise, are transformed into the actions of the final character[Stu98]. These transformations can range from being direct, as in the joint-by-joint mirroring used in the most basic form of performance animation, to sophisticated abstractions such as using a dataglove to drive a phoneme synthesizer[FH95]. The imitation interfaces proposed in [BH00] and [LCR+02] use full-body motion or some projection thereof to serve as an example-based interface for generating the synthetic motions that best match the input.

Other points in the performance-animation/puppetry design space have been explored by using more abstract mappings between the user input and the resulting animation. The work of [OTH02] uses two 6-DOF tracking devices to provide for interactive bimanual control over the stepping and walking motions of a character. The method presented in [LvdPF00] provide a series of interfaces that allow the control of characters within the context of an ongoing dynamical simulation, thereby leaving issues such as balance to the user to solve. Other recent work uses performance animation to animate hand-drawn facial animation[BFJ+00].

While physics can serve as a useful constraint in the synthesis of motion, many animations require motions that are physically-impossible, but that should still behave in some predictable, plausible fashion. The work of [CPIS02] is an interesting recent example of simulating cartoon style animation for deformable objects. Other recent work looks at how to reuse motions from existing hand-drawn animations[BLCD02].

3 Character Sketching

The two core components of our animation system are a character sketching tool and a motion sketching tool, as shown in Fig-

[1] *Animatics* are still frames containing key poses that are used to help determine the camera positions and timing of an animation sequence

ure 2. We first discuss the character sketching tool, which consists of sketching the basic body shape or skeleton, followed by adding sketched 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 sagittal (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.

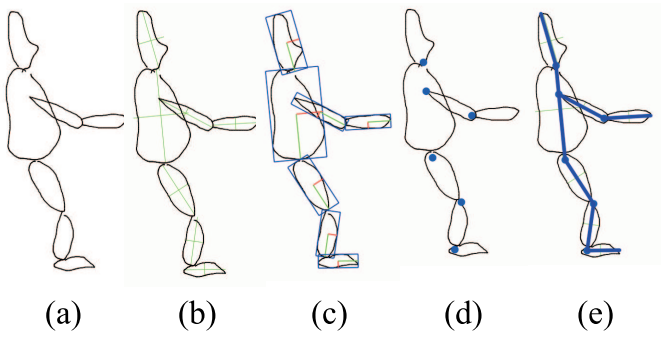


Figure 3: The process of inferring the skeleton from the sketch. (a) The seven sketched links. (b) Computed major and minor axes. (c) Oriented boxes. (d) Computed joint locations. (e) Computed skeleton.

Once skeletal links have been drawn, the system automatically infers the locations of the joints, labels the links, and creates the second arm and the second 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. The first is that individual links do not need to be identified – each recorded stroke is already known to be a link. Second, the expected connectivity of the links is known in advance. Lastly, characters are most often drawn in some kind of prototypical position. For example, it is natural to sketch an upright character rather than one that is upside-down. However, we do not take advantage of this last type of constraint and allow users to sketch the character in 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(c). 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 eigen-problem. 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 the torso segment
12. Create duplicate arm and leg segments.

Figure 4: Pseudocode for inferring the skeleton from the sketched links.

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, where this is 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 in a number of duplicate joints being created, 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.

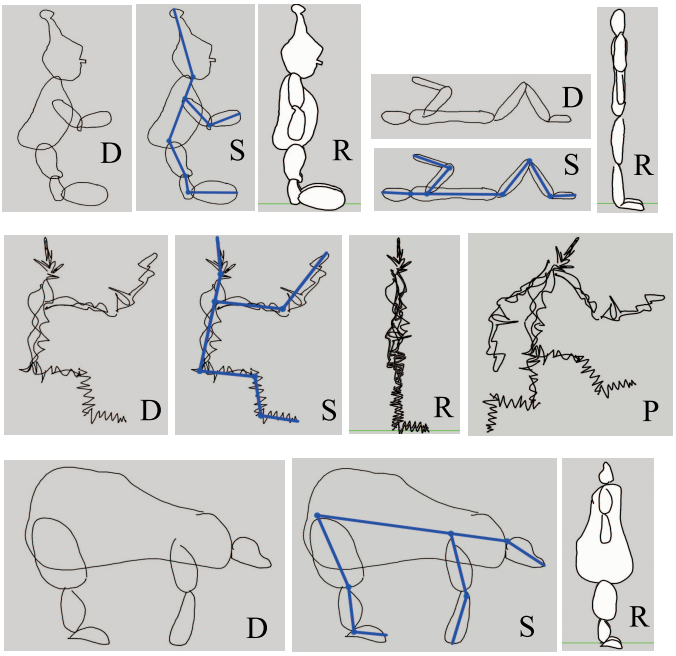


Figure 5: A variety of skeleton sketches and their inferred skeleton. *D*: the original drawing; *S*: inferred joints and the fitted skeleton; *R*: character in the reference pose; *P*: an animated pose.

There are two additional internal 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 algorithm used for inferring the skeleton will currently fail if 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 probably be addressed with some additional sophistication in the skeleton recognition algorithm. However, the algorithm is quite robust in its current form, as shown in Figure 5.

3.2 Adding Annotations

In many situations, the sketched skeletal links may be a sufficiently-detailed representation of the character for a quick-and-dirty animation. When this is not sufficient, our system allows the user to add further annotations which serve to decorate the links of the character. Thus, one can sketch additional features such as eyes, ears, hands, 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 that could be implemented to address this problem.

The foreground and background are sketched as annotations that are bound to the world coordinate frame, and keystrokes are used to invoke the foreground and background drawing modes. Figure 6 shows several skeletons before and after annotations have been added. As shown in the figure, the annotations automatically move with their associated links.

4 Motion Sketching

The notion of a motion sketch is not self evident. We wish to specify the movement of many degrees-of-freedom (DOF) over time using only a two DOF input device such as a stylus or mouse. We draw the inspiration for the motion-sketching component of system from the motion annotations sometimes employed in cartoon-style drawing, where a trailing arc will indicate the path of an object or character through the air, and loops in this trailing arc are indicative of a tumbling motion. The motion doodles used to sketch motions reflect these same semantics, and we additionally embed other context-sensitive motion indicators into the motion doodles in order to be able to produce a much wider range of motions.

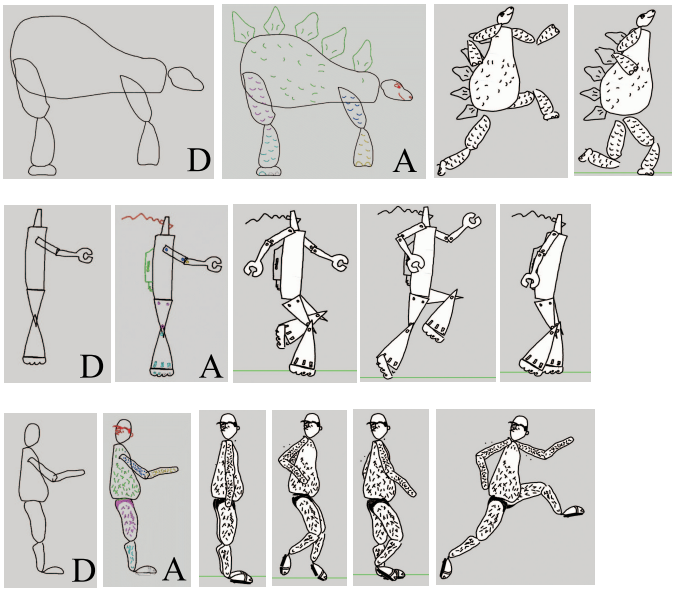


Figure 6: Sketching annotations to refine the character model. *D*: before annotation; *A*: with annotations.

4.1 Motion Sketching Principles

The motion sketching system is capable of a wide range of different types of motions, which are catalogued in Figures 7 and 8. The input motion sketch is representative of a number of things. In many cases the global shape and timing of the motion sketch reflects the motion of the center-of-mass, such as for the jumps, leaps, and flips. As such, a "slow motion" action can be created by simply slowly sketching the motion doodle for that action.

In other situations, the motion is representative of the desired location of the feet, such as for the walking motions, or the landing location for the jumps and flips. Although not shown directly in the figures, the system allows one to directly sketch the motion doodles on top of the environment, thus making it easy to jump over obstacles in the environment, for example.

Lastly, a number of other annotations can be embedded in the motion sketch. One example of this is the use of loops to indicate the number and type of revolutions to perform in a flip. Another is the use of asymmetry of the sketched walking arcs to specify either tip-toeing or stomping walks.

Many of the features contained in the input sketch are measured in an approximate fashion, either for simplicity, or to make allowances for annotations to be embedded into the sketch. For example, a sketched arc representative of a jump is defined by a limited number of parameters, namely the 2D positions and timings of the take-off, the peak-height, and the landing. These parameters are then used to fit a pair of ascending and descending parabolic arcs that will be followed by the center of mass. Other types of motion similarly extract a limited number of parameters from the input in order to provide the information for reconstructing the output motions.

4.2 Segmenting the Motion Sketch

The motion sketch needs to be broken into segments that correspond to the categories described in Figures 7 and 8. This segmentation is done online in our system, i.e., as the sketch is being drawn, but as with any gesture interpretation system, one often has to wait for the completion of an action or gesture before it can be appropriately labelled. This lag between the sketching and the execution of the motion is also a reflection of the anticipation required for the proper execution of most motions.

A 4-way relative-direction encoding of the input motion sketch is used in segmenting the sketch, as shown in Figure 9. For a forward step or a forward jump, one expects to see a relative motion up-and-right, followed by a relative motion down-and-right. The loop annotations designating front-flips and back-flips are recognized in a similar fashion, passing through one or more more cycles of the directional encoding, as illustrated in the example in Figure 9.

In order to further distinguish between a walking step and a jump, the maximum height of the arc is used. An arc of height greater than height h_{walk} is determined to be a jump, leap, or flip, while otherwise it is determined to be some type of walking step. In our current implementation, a horizontal line is drawn at $y = h_{walk}$ as shown in Figures 7 and 8, thereby providing the animator with a point of reference when sketching walking or jumping arcs. While we have found this criterion to be quite sufficient, another possibility is to allow an animator to directly impose a 'step' or 'jump' interpretation using a mode key. This would allow for both high steps and low jumps, which are mutually exclusive under the current scenario.

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 peak of the jump, allowing for a feasible jump.

If the user sketch remains stationary for more than $0.5s$, a standing posture is invoked. If the character was previously performing a stepping motion, the character brings the trailing leg forward to place it beside the other foot to achieve the standing posture. This also provides a mechanism for creating both a two-footed jump, and a one-footed leap. A normal jump occurs as a result of a sketched jump arc whenever both feet are together. If this is not the case, such as a walking step followed by a jump arc, it is identified as leap. Thus, transitioning from a walk to a two-footed jump is done by pausing for $0.5s$ to first allow a transition to a standing posture.

Once a segment of motion sketch input has been appropriately identified, the key parameters for that motion segment are extracted, and the output motion synthesis can begin. Because the sketch is always ahead of the motion synthesis process, successfully segmented motion actions are stored in a queue for processing.

4.3 Output Motion Synthesis

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. A summary of the various stages used to define each motion is given in Figure 10. The phases and keyframes for the walk cycles are inspired by those illustrated in [Wil01].

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, as recorded 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 are determined as a fixed fraction of the input-sketch descent time.

All stages have an associated keyframe that defines a target pose for 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 jumps, flips, and leaps, the center-of-mass is directly placed at the appropriate position as determined by the parabolic center-of-mass curve defined by the location of the peak of the input motion sketch.

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. Perhaps the most important modification is the inverse kinematics is applied to the legs for all stages 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 perform certain actions very differently than a character with a long torso and short legs. At present we deal with this issue at only two points. During the motion sketch, the length of the largest possible step, $d_{maxstep}$, is dependent on the character's leg length. Second, the inverse kinematics applied during any stage that involves ground contact makes use of skeletal dimension information. The remaining aspects of the motion in our current system are independent of the character proporitions, being driven purely by joint angles. While generally robust, characters with extreme proportions will occasionally exhibit problems with body parts passing through the ground.

A second motion design issue is dealing with variations in the input sketches. Thus, a jump of 10 metres will require a deeper crouch in anticipation of takeoff than a shorter jump of 1 metre. We do not address this issue in the current prototype and simply use the same anticipation keyframes. However, the landing phase of a jump is appropriately influenced by its height.

5 Results and Discussion

The animation system allows for animations to be produced very quickly. A new character can be drawn and animated in under 10 seconds, as demonstrated in Figure 11. The time taken to sketch the character and its annotations typically dominates the total animation time. The robot character shown in Figure 1 required approximately two minutes to draw and annotate, and another 4 minutes to draw the environment. Creating the subsequent animation required a matter of seconds.

While the speed with which an animation can be created is not the final measure of its success, it does greatly influence how an animation tool is used and who uses it. In particular, our sketching system has the potential to enable a wider audience to create their own character animations. The character sketching interface requires minimal knowledge to use – a user need only know what

links are expected to be drawn, that each link should be drawn with a single stroke, and that certain link configurations should be avoided during the original sketch, such as drawing the arms completely on top of the torso. As anecdotal evidence of this, a new user was able to use the system after only one minute of explanation (please refer to the video).

The motion sketching system allows for a variety of exaggerated cartoon-like motions, including leaping a building in a single bound, slow-motion jumps, and doing a flip with any number of rotations. The motion sketch can happen directly on top of a sketched environment, allowing the user to plan motions in the context of the environment.

The animation system can be used to produce a variety of motions for a large variety of characters. Figure 12 shows a typical example of how oddly-proportioned characters still animate in a well-behaved fashion. Lastly, Figure 13 shows a longer motion sketch that incorporates a variety of different types of motions, all in sequence.

Physical props, such as the edge of a book or ruler, can be used as a proxy for the ground when sketching with a tablet. The input stylus can bump into this edge when sketching jumps and walking steps, allowing for the stylus to be stopped by the 'ground' rather than being stopped by a gradual deceleration of the hand driving the stylus. This can allow for the locomotion to be controlled more precisely and in a way that mirrors the reality of a foot striking the ground. While not currently implemented, we foresee other uses for such proxies as well, such as being able to act out the sliding motion of a foot on the ground using the proxy.

The animation system has a number of aspects which could easily be improved upon. One area requiring improvement is the directional encoding used to parse the motion arcs, which is brittle with respect to some types of input. A typical problem occurs when the motion sketch for a forward jump moves involves relative directions other than up-right and down-right, such as when the landing portion of the sketch overshoots the vertical. A problem with the current annotation framework is that the annotations are only made with the articulated character in its original sketched pose. This means that one cannot pressently add annotations to the part of the upper body that the sketched arm currently covers, as can be seen for the 'hairy man' in Figure 6.

6 Conclusions and Future Work

We have presented a system that allows a character to be sketched and animated very quickly and with virtually no animator training. This is achieved by making assumptions about the types of characters that can be drawn and about the type of control one might want in a quick-and-dirty character animation.

There are a large number of possible extensions to this sytem. For the character sketching component, we wish to exploit the use of more sophisticated skeleton recognition algorithms to allow for an even broader range of valid character sketches, as well as inferring the shape of 3D characters from 2D sketches. Also, given a set of automatically-labelled links and knowledge of what annotations are likely to be sketched (clothes, hair, hats, etc.), we can reasonably expect to identify the specific annotations that are being sketched and therefore automatically animate them using appropriate techniques, such as physics-based secondary motion.

With respect to motion sketching component, we wish to allow for variable terrain, a broader range of motions, motion along 3D paths using a 3D input device, and motion doodles for controlling quadrupeds. It should also be possible to embed additional annotations into the input sketch. For example, the size of the loop which indicates a flip could directly represent the amount of tuck used in the flip.

References

[Bae69] R. Baecker. Picture-driven animation. In *Proceedings of the Spring Joint Computer Conf., AFIPS Press*, pages 273–288, 1969.

[BC89] Armin Bruderlin and Thomas W. Calvert. Goal-directed, dynamic animation of human walking. In *Computer Graphics (Proceedings of SIGGRAPH 89)*, volume 23, pages 233–242, Boston, Massachusetts, July 1989.

[BFJ+00] I. Buck, A. Finkelstein, C. Jacobs, A. Klein, D. Salesin, J. Seims, R. Szeliski, and K. Toyama. Performance-driven hand-drawn animation. In *Symposium on Non-Photorealistic Animation and Rendering (NPAR 2000)*, pages 101–108, 2000.

[BG95] Jean-Francis Balaguer and Enrico Gobbetti. Sketching 3D animations. *Computer Graphics Forum*, 14(3):241–258, 1995.

[BH00] Matthew Brand and Aaron Hertzmann. Style machines. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference

Series, pages 183–192. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, July 2000. ISBN 1-58113-208-5.

[BLCD02] Christoph Bregler, Lorie Loeb, Erika Chuang, and Hrishi Deshpande. Turning to the masters: Motion capturing cartoons. *ACM Transactions on Graphics*, 21(3):399–407, July 2002. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002).

[CPIS02] Stephen Chenney, Mark Pingel, Rob Iverson, and Marcin Szymanski. Simulating cartoon style animation. In *NPAR 2002: Second International Symposium on Non Photorealistic Rendering*, pages 133–138. ACM SIGGRAPH / Eurographics, June 2002. ISBN 1-58113-494-0.

[FH95] Sid Fels and Geoffrey Hinton. Glove-talk ii: An adaptive gesture-to-format interface. In *Proceedings of CHI'95: Human Factors in Computing Systems*, pages 456–463. ACM Press, 1995.

[Gir87] M. Girard. Interactive design of computer-animated legged animal motion. *IEEE Computer Graphics and Applications*, 7(6):39–51, 1987.

[IMT99] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A sketching interface for 3d freeform design. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 409–416, Los Angeles, California, August 1999. ACM SIGGRAPH / Addison Wesley Longman. ISBN 0-20148-560-5.

[LCR+02] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics*, 21(3):491–500, July 2002. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002).

[LP02] C. Karen Liu and Zoran Popović. Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics*, 21(3):408–416, July 2002. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002).

[LvdPF00] Joseph Laszlo, Michiel van de Panne, and Eugene L. Fiume. Interactive control for physically-based animation. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 201–208. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, July 2000. ISBN 1-58113-208-5.

[OTH02] Sageev Oore, Demetri Terzopoulos, and Geoffrey Hinton. A Desktop Input Device and Interface for Interactive 3D Character Animation. In *Proc. Graphics Interface*, pages 133–140, May 2002.

[Pop01] Jovan Popovic. *Interactive Design of Rigid-Body Simulations for Computer Animation*. PhD thesis, Carnegie Mellon University, 2001. CMU-CS-01-140.

[Stu98] David J. Sturman. Computer puppetry. *IEEE Computer Graphics and Applications*, 18(1):38–45, Jan-Feb 1998.

[Sut63] I. E. Sutherland. Sketchpad: A man-machine graphical communication system. In *Proceedings AFIPS Spring Joint Computer Conference*, volume 23, pages 329–346, 1963.

[TT98] M. Teichmann and S. Teller. Assisted articulation of closed polygonal models. In *9th Eurographics Workshop on Animation and Simulation*, 1998.

[vdP97] M. van de Panne. From footprints to animation. *Computer Graphics Forum*, 16(4):211–224, 1997. ISSN 1067-7055.

[Wil01] R. Williams. *The Animator's Survival Kit*. Faber and Faber, 2001.

[ZHH96] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. Sketch: An interface for sketching 3d scenes. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 163–170, New Orleans, Louisiana, August 1996. ACM SIGGRAPH / Addison Wesley. ISBN 0-201-94800-1.

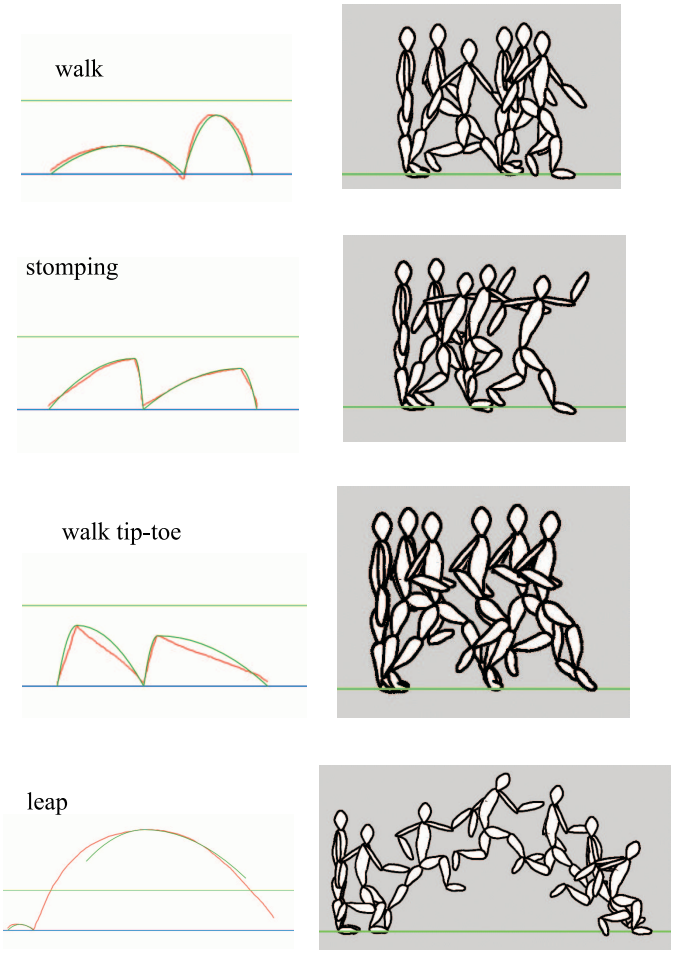


Figure 7: Walking motions and their variations. The horizontal green line in the motion sketches is a jump threshold, above which a curve is interpreted as a leap. The red curve is the input motion, while the green curve is the fit of two half parabolae to the input.

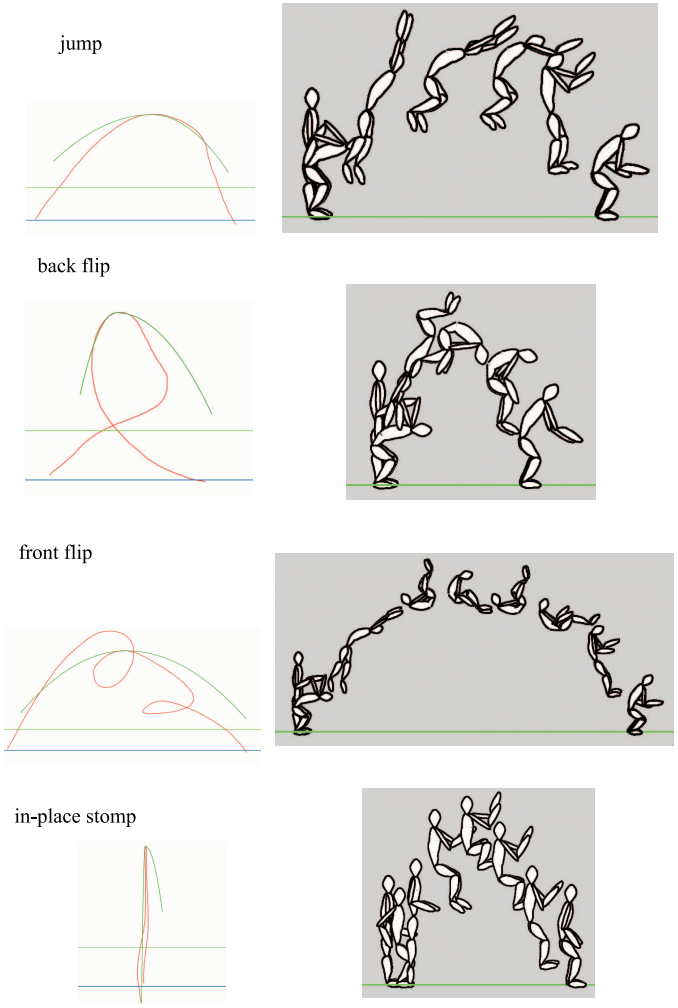


Figure 8: Jumping motions and their variations. The direction of travel in all figures is from left to right. The frames for the in-place stomp animation have been artificially translated to better depict the motion.

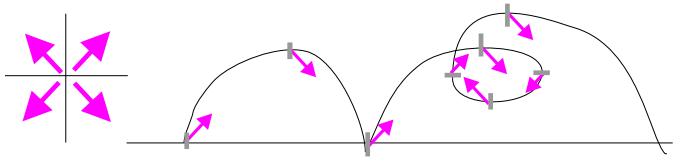
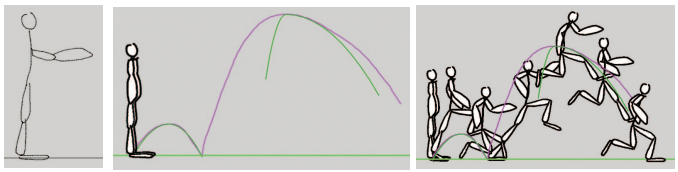


Figure 9: Segmenting the motion sketch input.

jump, in-place stomp,leap	walk, tip-toe, and stomping	front flip, back flip
anticipate launch ascent descent landing followthrough	contact down passing up	crouch launch ascent curl spin uncurl landing followthrough

Figure 10: Stages used for motion synthesis.



(a) (b) (c)

Figure 11: A typical fast character and motion sketch, done in under 10 seconds. (a) Character sketch. (b) Motion sketch. (c) Resulting motion.

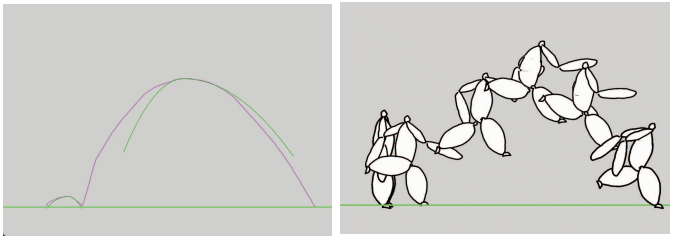


Figure 12: A character with odd proportions. (a) Motion sketch. (b) Resulting motion.

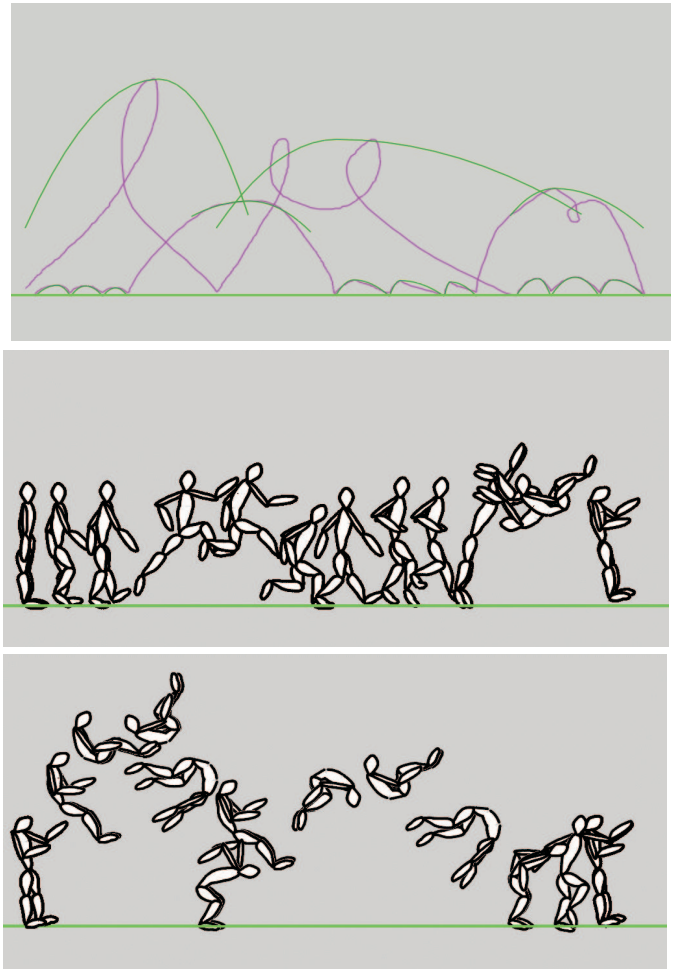


Figure 13: A longer sketched animation. The character takes 3 forward steps, a leap, 3 forward steps, a leap, 3 backwards steps, performs a backwards-traveling double front-flip, followed by one more backwards-traveling front flip.