Gestural Motion Editing using Mobile Devices

Noah Lockwood
University of Toronto
Industrial Light & Magic

Karan Singh
University of Toronto

Abstract

We present novel techniques for interactive editing the motion of an animated character by gesturing with a mobile device. Our approach is based on the notion that humans are generally able to convey motion using simple and abstract mappings from their own movement to that of an animated character. We first explore the feasibility of extracting robust sensor data with sufficiently rich features and low noise, such that the signal is predictably representative of a user's illustrative manipulation of the mobile device. In particular, we find that the linear velocity and device orientation computed from the motion sensor data are well-suited to the task of interactive character control. We show that these signals can be used for two different methods of interactively editing the locomotion of an animated human figure: discrete gestures for editing single motions, and continuous gestures for editing ongoing motions. We illustrate these techniques using various types of motion edits which affect jumps, strides and turning.

Keywords: character animation; motion editing; mobile devices

Concepts: •**Computing methodologies** → *Animation;*

1 Introduction

Animated characters are increasingly prevalent in films, games, and as avatars in interactive virtual experiences. Authoring or manipulating an animated 3D character with several complexly correlated degrees of freedom can be a daunting task for novices and meticulous work for experts.

One promising approach to interactive character animation are *performance interfaces*, where the user physically performs a version of a motion, and their performance is used to manipulate the digital character in space and/or time in an absolute or relative fashion. An appealing aspect of a performance interface is that intricate space-time relationships resulting from the motion of biomechanical joint chains are implicitly included in the physicality of the performance; manually authoring these relationships to appear plausible is extremely difficult. The challenge with all performance interfaces is defining mappings between the user's motion and the manipulated character that are both conceptually natural and easy to perform by the user.

Mobile devices such as tablets and smartphones provide an opportunity for performance interfaces, given their increasing computational power and familiarity among many users. While the touch surface of a mobile device could be utilized in a manner similar to prior work using a pen [Thorne et al. 2004] and multi-touch input [Lockwood and Singh 2012] for character control, we are to our knowledge the first to explore the use of simple motion sensor data from a commodity mobile device for interactive control of motion editing.

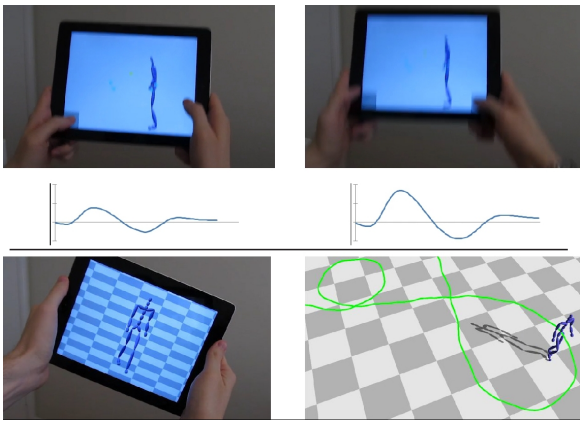


Figure 1: *Reference and performed motions (showing tablet vertical velocity) to edit a jump (top); Tablet orientation to control character heading (bottom).*

Overview: We found informally that users could utilize two different methods for communicating character motion by manipulating a handheld device: by either mimicking the overall body motion in an illustrative gesture, or by mimimicking interaction with a control device correlated with a parameter of a motion such as character heading. We analyzed the sensor data available in mobile devices, which many users are comfortable manipulating, to enable interactive control of motion editing with either *discrete gestures* (for motions with a defined beginning and end) or *continuous gestures* (for ongoing but parameterized motions). These gestures were then used to control the editing handles of a path-based motion editing algorithm [Lockwood and Singh 2011], which resulted in motion edits that matched user expectation (Figure 1).

Contribution: Our primary contribution is, to our knowledge, the first system to use mobile device motion sensors for interactive character animation. Within this context, we analyze example performance data to propose motion signals best suited to the task of controlling motion edits. We then develop two novel algorithms: first, the usage of discrete gestures in the form of separate reference and editing gestures which control the extent of a standalone motion; and second, the usage of a user's continuous gesture to edit ongoing motion interactively, while maintaining continuity across motion cycles.

2 Related Work

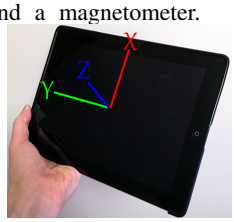
Our work is inspired by a rich body of research in puppet-like performance interfaces for animation, using input in a variety of forms including manipulation or movement through space of handheld input devices [Oore et al. 2002; Dontcheva et al. 2003; Shiratori and Hodgins 2008; Numaguchi et al. 2011; Jacobson et al. 2014],

MiG '16,, October 10-12, 2016, Burlingame, CA, USA
ISBN: 978-1-4503-4592-7/16/10
DOI: http://dx.doi.org/10.1145/2994258.2994276

body movement [Shin et al. 2001; Ha et al. 2011; Seol et al. 2013; Gupta et al. 2014; Rhodin et al. 2014], and touch-based or pen input [Thorne et al. 2004; Lockwood and Singh 2012]. Prior work has shown that a small number of points could be used to effectively manipulate a character's motion trajectory [Gleicher 2001], and even the trajectories of multiple interacting characters [Kim et al. 2009]. Our method builds upon the path-based motion editing algorithm of Lockwood and Singh [Lockwood and Singh 2011] which automatically modifies the motion and timing of a character according to biomechanical principles when the user edits its path by repositioning automatically-identified motion editing handles. In this work, we use motion sensor data to automatically control the positions and arrangement of these motion handles.

3 Motion Data from Mobile Devices

Smartphones and tablets today are commonly equipped with a 3-axis accelerometer, a gyroscope, and a magnetometer. The raw signal data from these sensors is typically processed to provide motion signals in the form of linear acceleration along each local axis of the device, rotation around a semi-arbitrary initial world coordinate frame, and a rate of rotation (in a predefined rotation order) around each local axis (shown inset for a tablet).



Motion sensor data has been analyzed for use in a wide variety of scenarios, ranging from a passive determination of user context/activity [Hinckley et al. 2005; Sprager and Zazula 2009; Kwapisz et al. 2011] to more active gestural control of the device [Jones et al. 2010; Zarek et al. 2012]. A performance interface to edit and control an animated character requires sensor signals that are smooth, so they can be mapped to character motion without introducing noise artifacts. The signal must also be sufficiently detailed, so that motion features like the motion extrema of the velocity signal shown in Figure 1(top) can be identified reliably in order to edit motion or compare the signals of different performances.

Sensor data needs to analyzed in a user-centric reference frame, where the data is more representative of a performance. We thus transform the device-local accelerometer data into user-space, using the gyroscope-based current rotation of the device. A pre-established vertical axis (opposite to gravity), and arbitrary but consistent horizontal axes are used, which are tracked by the device over time. Signals for velocity and position of the device are obtained by integration (using finite differences) of the user-space acceleration.

Figure 2 shows these values along the vertical axis for a "jump" gesture performed with the device, mimicking the motion of a full-body jump. There is substantial noise observable in the raw acceleration data, which can make the robust identification of motion extrema difficult; however, filtering the data could reduce the accuracy of the extrema by over-smoothing them. The twice-integrated position data, which should reflect the vertical up-and-down motion of the device during the jump, lacks noise and is dominated by drift introduced by the accelerometer data, resulting in a smooth but featureless signal. This drift is a result of bias in the accelerometers, which can also be observed as non-zero acceleration when the device was stationary at the beginning and end of the gesture. While this bias could potentially be measured and accounted for, it still presents a significant impediment to using integrated position data. Integrating acceleration once to provide a velocity signal is a good trade-off, providing smoothness while preserving important motion features. Accelerometer drift results in a linear offset that skews the velocity, but has a smaller impact on its motion features.

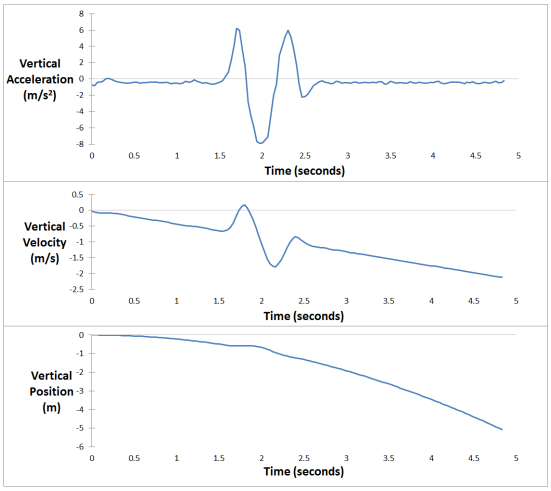


Figure 2: *Vertical acceleration, integrated velocity, and twice-integrated position during the performance of a "jump" gesture.*

However, mimicking the overall motion of a character is not always the most appropriate method of control. For ongoing control of motion heading, we utilize a more direct conceptual mapping between the device and the animated character: that of a steering wheel, which users may be familiar with from mobile or console video games. When the device is held level, users can reasonably expect the character to proceed straight ahead, and turn as appropriate in response to tilting the device. This "steering angle" of the device can be measured and applied instantaneously, as long as the motion editing algorithm ensures continuity of the character's motion over time. We are able to compute this instantaneous steering angle from the device's current rotation directly, as the device's rotation around an axis normal to its display, allowing the "steering column" to be oriented to the user's comfort (Figure 3). We compute the steering angle from world-space device rotation as the difference between the device's current rotation and a *level* version of that rotation, determined using the world up direction. This level rotation is the smallest rotation around an axis normal to the device's display which would make an edge of the device parallel with the ground plane. Steering can be disabled when the device is near parallel with the ground plane, which would otherwise result in a steering angle which is undefined.

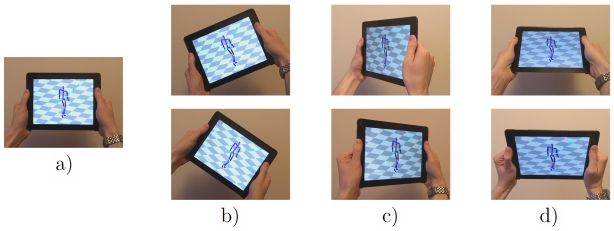


Figure 3: *Starting with a neutral "steering wheel" grip of the device (a), steering angle is measured as the rotation left or right around an axis normal to the display (b). Rotating the device around a vertical axis (c) or tilting it (d) should have no effect on the measured steering angle.*

4 Motion Editing with Discrete Gestures

We assume for the purpose of this section that the input motion is finite, like the jump in Figure 1(top), or has a representative cycle

like the walk in Figure 1(bottom). We also assume that each such motion has a pre-defined gestural representation, either computed automatically from the motion using any of a variety of pose extraction schemes [Assa et al. 2005], or defined manually in advance, such as a simple mimicking of the overall body motion. This gestural representation, which is correspondingly finite, is referred to as a *discrete gesture*. However, a single performance of this gesture is not sufficient for editing. Because different users might naturally perform the same in different ways, a baseline of some sort is required to determine the scale of the motion. Instead of requiring users to perform with a fixed scale, we assume that a single user's performances are consistent amongst each other, and utilize two gestures: a reference gesture, mimicking the current motion, and an editing gesture, mimicking the desired new motion.

Figure 4 shows that while the extents of the vertical velocity curves of various jump performances may differ, they velocity profiles are remarkably similar in shape. The velocity extrema may not coincide with spatially intuitive parts of the gesture; for example, the velocity maximum naturally occurs midway through the "upswing", before slowing and becoming negative prior to the apex of the device's trajectory. As long as we are able to match the velocity features with those of the reference motion, however, a meaningful difference between the motions can be computed to define a *scaling factor*: for example, a scaling factor of 2.0 for a "jump" motion would indicate that the user's editing gesture was twice as high as the reference gesture, and that the jumping motion should be edited accordingly.

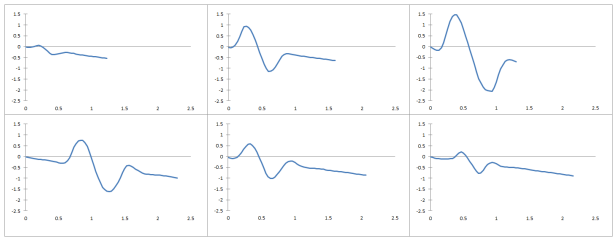


Figure 4: *Vertical velocity profiles of "jump" gestures of various heights.*

This scale factor is determined by the product of the relative scaling in each of the velocity and time dimensions. Due to the relative simplicity of the velocity profiles, our approach calculates this scaling from an approximate similarity transform from a small number of corresponding feature points (the local extrema) identified in the profiles. In the simple case of two identified features in each gesture, the scaling factor can be determined as follows. The features of the reference gesture, $\mathbf{r}_0$ and $\mathbf{r}_1$ correspond respectively to the features of the editing gesture, $\mathbf{e}_0$ and $\mathbf{e}_1$, with points in a two-dimensional space where the axes are time and velocity: $\mathbf{r}_i = (r_{it}, r_{iv}), \mathbf{e}_i = (e_{it}, e_{iv})$. The deltas between each gesture's pair of features, $\Delta\mathbf{r}$ and $\Delta\mathbf{e}$, are the vectors from the first to second features: $\Delta\mathbf{r} = \mathbf{r}_1 - \mathbf{r}_0 = (\Delta r_t, \Delta r_v), \Delta\mathbf{e} = \mathbf{e}_1 - \mathbf{e}_0 = (\Delta e_t, \Delta e_v)$. The final scaling factor, s, is the product of the ratios between each dimension of the delta vectors: $s = \frac{\Delta e_t}{\Delta r_t} \cdot \frac{\Delta e_v}{\Delta r_v}$.

This scaling factor represents the change in *spatial extent* between the reference and editing gestures. That same scaling factor can be applied to the current motion to obtain a new motion with a correspondingly edited extent.

We now describe how a motion can be parameterized to capture how the entire motion changes relative to an appropriate spatial extent. This parameterization specifies the position of a sequence of motion editing *handles* which are automatically detected by, and control the path-based motion editing method of Lockwood and Singh [Lockwood and Singh 2011]. We define the n editing handles for an input motion, with initial positions $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n$. Our parameterization is a set of functions $\hat{\mathbf{h}}_i(t)$, each of which determines the new position of a handle based on the scalar parameter t: $\hat{\mathbf{h}}(t) = \left\{\hat{\mathbf{h}}_1(t), \hat{\mathbf{h}}_2(t), \ldots, \hat{\mathbf{h}}_n(t)\right\}$. This motion parameterization can be generated procedurally, or manually authored by providing a number of example positions for the motion handles. We compute edited motion handles by simply computing the functions at a parameter $s \cdot \bar{t}$, where $\bar{t}$ is the parameter corresponding to the reference motion.

Figure 5 shows an example of this technique for editing jump height. The features within the vertical velocity profiles of the reference and editing gestures are determined, which results in the calculation of a scale factor of 2.98. This scale factor is applied to the initial jump height of 23cm, resulting in a new jump height of 68cm.

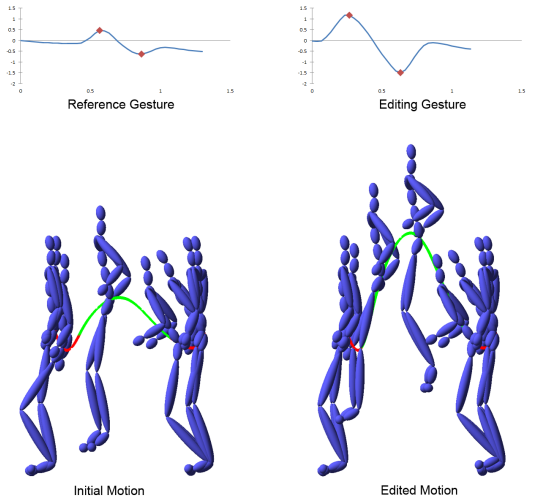


Figure 5: *Velocity profiles with identified features and the corresponding motions for the reference gesture (left) and the editing gesture (right) for editing jump height.*

The procedure for editing a jump's distance is similar to editing jump height. The reference and editing gestures are two "forward jump" gestures demonstrating the current and desired distance of the jump. While the vertical velocity of the handheld device is relevant in these gestures, the horizontal velocity is also important, as it demonstrates the horizontal distance of the gesture. However, since the device's world horizontal axes can be oriented arbitrarily, the gesture's motion may occur along a combination of those axes. We thus, use the magnitude of horizontal velocity $\|\mathbf{v}_{horiz}\| = \sqrt{\mathbf{v}_x \cdot \mathbf{v}_x + \mathbf{v}_z \cdot \mathbf{v}_z}$. Figure 6 shows the horizontal velocity profiles from four different "jump forward" gestures of increasing distance, along with the corresponding vertical velocities.

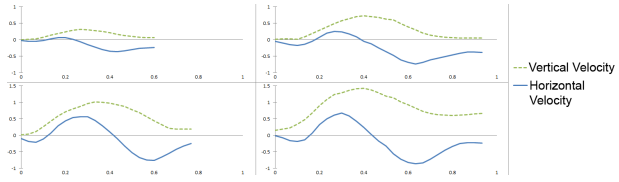


Figure 6: *Horizontal and vertical velocity profiles of four different "jump forward" gestures of varying distances.*

In these examples, the differing magnitude of the peak horizontal velocity offers a clear indication of the magnitude of the gesture, but the smoothness of the velocity over time makes determining exact correspondences between gestures difficult. Conversely, the vertical velocity has useful features, as utilized in the previous section to edit jump height, but does not capture the magnitude of the gesture's horizontal distance. We thus the vertical velocity features to determine timing, but the horizontal velocity values to determine the scale factor, as the product of the ratio of the duration between each pair of features and the ratio of the maximum horizontal velocity during the time between the corresponding features.

Figure 7 shows an example of this technique for editing jump distance. The features within the velocity profiles of the reference and editing gestures are determined, which results in the calculation of a scale factor of 1.77. This scale factor is applied to the current jump distance of 88cm, resulting in a new jump distance of 156cm.

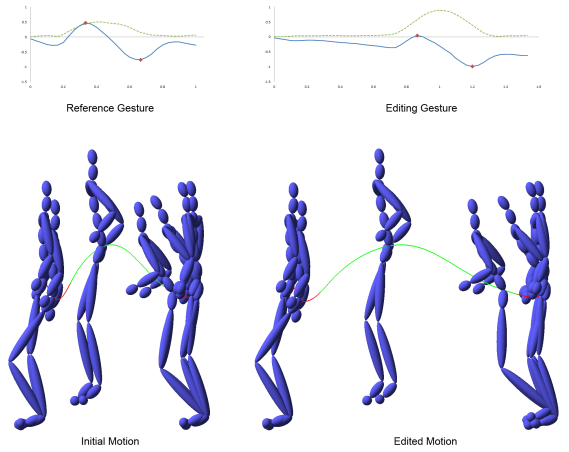


Figure 7: *Velocity profiles with identified features and the corresponding motions for the reference gesture (left) and the editing gesture (right) for editing jump distance.*

These methods can be applied to the editing of non-jumping motions as well. To edit the stride distance of a walking motion, a "forward walk" gesture similar to the forward jump is used, but the user can perform multiple "hops", representing steps. We cannot expect users to provide a precise correspondence between the reference and editing gestures; therefore, instead of identifying a single pair of features within each profile, the longest continuous sequence of such pairs is identified, where each pair signifies a performed step. The scaling factor is then calculated using the average of the calculations on all steps in each gesture, but instead of an average of ratios between reference and editing gestures, a ratio of averages is used, as the number of steps in both gestures may differ.

Figure 8 shows an example of this technique for editing stride distance. The features within the velocity profiles of the reference and editing gestures are determined, which results in the calculation of a scale factor of 1.75. This scale factor is applied to the current walking motion with a total distance of 3.36m, resulting in a new walk with a total distance of 5.89m.

5 Motion Editing with Continuous Gestures

Sometimes more direct spatial control over a motion is required than can be obtained by a velocity based gestural approach. A prime example of this is controlling the motion trajectory of a walking character. We address such scenarios with a technique utilizing ongoing or *continuous gestures*, demonstrated in Figure 9. As with the scenarios of discrete gestural editing, this is a case of a relatively simple motion with a single spatial degree of freedom.

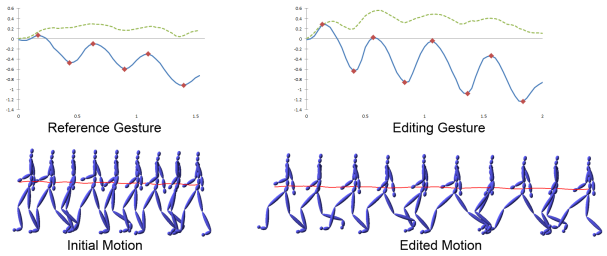


Figure 8: *Velocity profiles with identified features and the corresponding motions for the reference gesture (left) and the editing gesture (right) for editing stride distance.*

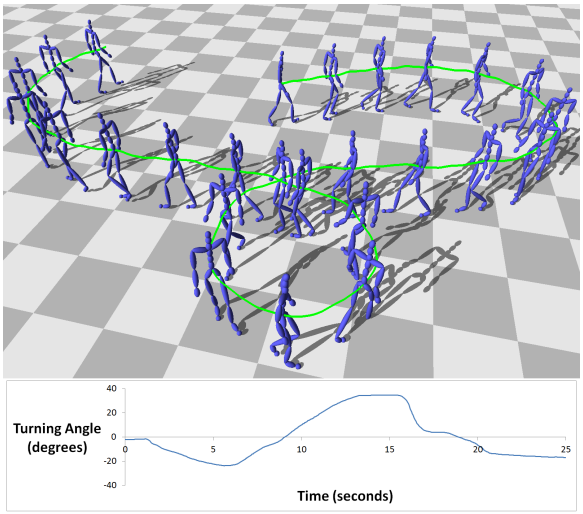


Figure 9: *A novel motion generated with continuous gestural control of the motion's turning angle in real-time.*

A parameterized motion, similar to those used for discrete gestural editing, forms the basis of the motion that will be generated with continuous control. The parameterization takes the scalar steering angle as input, and modifies a straight-ahead walking motion to have a constant turning rate by arranging the handles along an approximate circular arc. The entire parameterized motion is modified in real-time as the steering angle changes based on how the user manipulates the device. Equal successive rotations around a vertical axis are applied at each editing handle, so that the direction of each segment (formed between pairs of editing handles) differs from that of the previous segment by an angle equal to the steering angle multiplied by a constant factor. The distance between successive editing handles remains unchanged, and handles "bend" as if a rigid chain. Note that the bending this motion provides relative rather than absolute control of character orientation, thus retaining higher-frequency motion, such as hip sway, from the unedited motion. Figure 10 shows the results of this path bending for a variety of steering angles.

The method as described generates consistent motion directed by the continuous steering control of the user, but is ultimately limited by the length of the original motion. However, the cyclic repetition in a motion such as forward walking means that edits to a small rep-

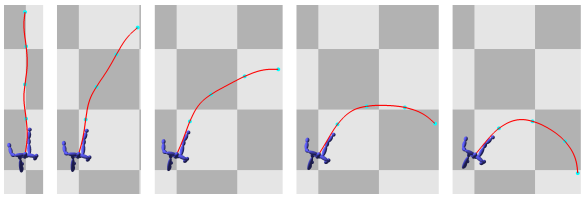


Figure 10: *A short walking motion with successive handles bent by interactive steering angles of 0, 10, 20, 30, and 40 degrees.*

resentative motion (a "sidecar" motion) can be applied repeatedly, to produce a streamed motion very similar to the results of editing a large motion composed of many cycles. In the case of forward walking, the cyclic portion of the motion is two forward steps, one left and one right. Whenever an evaluation of the sidecar motion would require a pose beyond the last frame, the method conceptually splits this timestep in two parts. First, the sidecar motion is stepped forward to exactly the last frame, and then "rewound" to the first frame. Then, the remaining timestep is taken and the procedure continues as before. For efficiency, it is desirable to have a sidecar motion which is as short as possible, but a motion consisting of the smallest repeatable part, two forward steps, will not generate consistent motion due to the path-based editing technique's behaviour at the endpoints of an edited motion. We thus use a walk with four steps, and the actual cycled portion of the motion is in the interior two steps, with the first and last steps providing continuity in the character's facing direction. The difference between using a two-step sidecar motion and the interior two steps of a four-step motion is shown in Figure 11.

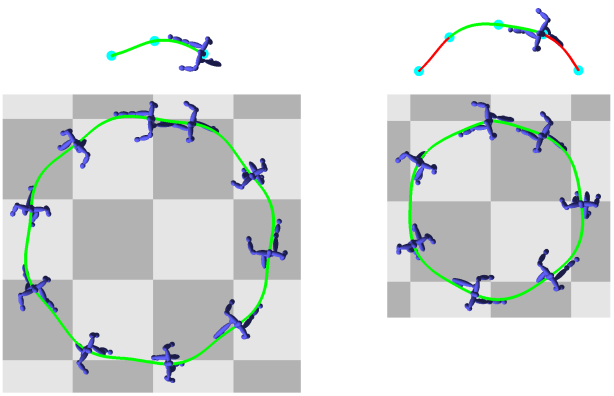


Figure 11: *A two-step sidecar motion (top left) results in a path shape that does not appear smooth when repeated (bottom left). Using the interior two steps of a four-step motion (top right) results in a tighter, smoother path when repeated (bottom right).*

6 Animator Feedback

One application of our work is to potentially make the creation of animation more accessible to users who have no prior experience. In addition, we were also curious how our techniques could be applied by professional animators in their efforts to produce extremely high-quality animation. To that end, we presented our techniques running interactively on a tablet to five visual effects character animators, with experience ranging from four to twenty-two years, and discussed their potential applications.

The animators all agreed that our techniques could have a significant impact on their work during the early stages of crafting an animated performance, where a character's movement through the scene and environment - the "blocking" - are established. Creating the blocking of a single performance through traditional keyframing can be quite time-consuming, a difficulty which is compounded by the often many iterations and versions produced as different possible performances are explored, or the goals change after feedback from a director. All of the animators felt that directly authoring initial character movements by gestures would be far more efficient, with some animators pointing out the similarities to the reference material that they perform themselves in video or motion capture. While the highly nuanced performances in feature-quality animation require significant animator time and effort to manually craft, our techniques could allow the broad strokes of a performance to be converged upon much more quickly, allowing even more time to refine the final performance.

7 Conclusion

We have presented a novel approach to interactive motion editing using gestural manipulations of mobile devices. A technique for discrete editing of motions identifies features in the velocity profile of the device during both an initial reference gesture and subsequent editing gesture. The difference between these gestures is applied to a variety of motions which have been specifically parameterized using a path-based editing algorithm. A second method for continuous editing of motions uses steering wheel-style manipulation of the device to control the path bending of a cyclical walking animation. Successive poses from this cyclical animation are arranged in space to ensure continuity as the motion continues indefinitely, with continuity during cycling enforced by careful automatic arrangement of the path editing handles.

There are limitations to our work. There can be noticeable noise as well as a lack of precision in handheld gestures, which can be caused by either the device's sensors and/or instability in the user's gestures; different methods or hardware could potentially track the position and rotation of the device with more precision, though even with perfect data the illustrative and imprecise nature of user performances of motion [Lockwood and Singh 2012] must be accounted for. We also require that the input motion be parameterized a-priori; while this is simple to specify, an automatic approach could be more flexible, such as determining which aspect of the motion to edit based on its similarity to the reference gesture [Dontcheva et al. 2003]. In addition, a system which unifies the use of discrete and continuous gestural control by seamlessly switching between those methods could be useful for generating full-body motions with more variety, such as causing the directional control of walking character to transition to directional control of a running character. Finally, a user study could provide quantitative data on how effective manipulating a mobile device is, compared to other input methods which could still drive our motion editing techniques. Despite these limitations, our work has shown the potential of motion sensor information to control animated characters independently, or in conjunction with touch-based techniques, on mobile devices.

Acknowledgements

We thank Isabelle Ortega for her invaluable and tireless support, Andrew German for lending his iOS programming expertise, and John Hancock of the Dynamics Graphic Project lab for technical assistance. We are also grateful to the anonymous reviewers for their comments, and to the animators for their useful feedback and perspective. All motion capture clips were from the Carnegie Mellon University motion capture database.

References

ASSA, J., CASPI, Y., AND COHEN-OR, D. 2005. Action synopsis: Pose selection and illustration. ACM Press.

BARNES, C., JACOBS, D. E., SANDERS, J., GOLDMAN, D. B., RUSINKIEWICZ, S., FINKELSTEIN, A., AND AGRAWALA, M. 2008. Video puppetry: a performative interface for cutout animation. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*.

DONTCHEVA, M., YNGVE, G., AND POPOVIĆ, Z. 2003. Layered acting for character animation. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*.

GLEICHER, M. 2001. Motion path editing. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*.

GUPTA, A., AGRAWALA, M., CURLESS, B., AND COHEN, M. 2014. Motionmontage: A system to annotate and combine motion takes for 3d animations. In *CHI '14: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

HA, S., BAI, Y., AND LIU, C. K. 2011. Human motion reconstruction from force sensors. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

HINCKLEY, K., PIERCE, J., HORVITZ, E., AND SINCLAIR, M. 2005. Foreground and background interaction with sensor-enhanced mobile devices. *ACM Trans. Comput.-Hum. Interact. 12*, 1 (Mar.), 31–52.

JACOBSON, A., PANOZZO, D., GLAUSER, O., PRADALIER, C., HILLIGES, O., AND SORKINE-HORNUNG, O. 2014. Tangible and modular input device for character articulation. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH) 33*, 4, 82:1–82:12.

JONES, E., ALEXANDER, J., ANDREOU, A., IRANI, P., AND SUBRAMANIAN, S. 2010. Gestext: Accelerometer-based gestural text-entry systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI '10, 2173–2182.

KIM, M., HYUN, K., KIM, J., AND LEE, J. 2009. Synchronized multi-character motion editing. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*.

KWAPISZ, J. R., WEISS, G. M., AND MOORE, S. A. 2011. Activity recognition using cell phone accelerometers. *SIGKDD Explor. Newsl. 12*, 2 (Mar.), 74–82.

LOCKWOOD, N., AND SINGH, K. 2011. Biomechanically-inspired motion path editing. In *SCA '11: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

LOCKWOOD, N., AND SINGH, K. 2012. Finger walking: Motion editing with contact-based hand performance. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '12, 43–52.

NUMAGUCHI, N., NAKAZAWA, A., SHIRATORI, T., AND HODGINS, J. K. 2011. A puppet interface for retrieval of motion capture data. In *SCA '11: Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

OORE, S., TERZOPOULOS, D., AND HINTON, G. 2002. A desktop input device and interface for interactive 3D character animation. In *Proceedings of the Graphics Interface Conference*.

RHODIN, H., TOMPKIN, J., KIM, K. I., KIRAN, V., SEIDEL, H.-P., AND THEOBALT, C. 2014. Interactive motion mapping for real-time character control. *Computer Graphics Forum (Proceedings Eurographics) 33*, 2.

SEOL, Y., O'SULLIVAN, C., AND LEE, J. 2013. Creature features: online motion puppetry for non-human characters. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

SHIN, H. J., LEE, J., SHIN, S. Y., AND GLEICHER, M. 2001. Computer puppetry: An importance-based approach. *ACM Trans. Graph. 20*, 2, 67–94.

SHIRATORI, T., AND HODGINS, J. K. 2008. Accelerometer-based user interfaces for the control of a physically simulated character. In *ACM SIGGRAPH Asia 2008 Papers*.

SPRAGER, S., AND ZAZULA, D. 2009. Gait identification using cumulants of accelerometer data. In *Proceedings of the 2Nd WSEAS International Conference on Sensors, and Signals and Visualization, Imaging and Simulation and Materials Science*, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, SENSIG'09/VIS'09/MATERIALS'09, 94–99.

THORNE, M., BURKE, D., AND VAN DE PANNE, M. 2004. Motion doodles: an interface for sketching character motion. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*.

YIN, K., AND PAI, D. K. 2003. Footsee: an interactive animation system. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*.

YOSHIZAKI, W., SUGIURA, Y., CHIOU, A. C., HASHIMOTO, S., INAMI, M., IGARASHI, T., AKAZAWA, Y., KAWACHI, K., KAGAMI, S., AND MOCHIMARU, M. 2011. An actuated physical puppet as an input device for controlling a digital manikin. In *CHI '11: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

ZAREK, A., WIGDOR, D., AND SINGH, K. 2012. Snout: One-handed use of capacitive touch devices. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, ACM, New York, NY, USA, AVI '12, 140–147.