Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

New Technique:

Sketch-based rotation editing*

Yue XIE[1], Wei-wei XU[2], Yi-zhou YU[3], Yan-lin WENG‡[1]

([1]*School of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China*)

([2]*Internet Graphics Group, Microsoft Research Asia, Beijing 100190, China*)

([3]*Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801, USA*)

E-mail: xie.full@gmail.com; wwxu@microsoft.com; yyz@illinois.edu; weng@cad.zju.edu.cn

Received Oct. 24, 2010; Revision accepted Mar. 4, 2011; Crosschecked Sept. 28, 2011

Abstract: We present a sketch-based rotation editing system for enriching rotational motion in keyframe animations. Given a set of keyframe orientations of a rigid object, the user first edits its angular velocity trajectory by sketching curves, and then the system computes the altered rotational motion by solving a variational curve fitting problem. The solved rotational motion not only satisfies the orientation constraints at the keyframes, but also fits well the user-specified angular velocity trajectory. Our system is simple and easy to use. We demonstrate its usefulness by adding interesting and realistic rotational details to several keyframe animations.

Key words: Quaternions, Keyframe interpolation, Animation editing
doi:10.1631/jzus.C1000373 **Document code:** A **CLC number:** TP391.4

1 Introduction

Real-world motion often has abundant rotational details, such as fluctuations in the instantaneous rotation axis of a spinning top on the ground. It is time-consuming for inexperienced artists to model this kind of rotation in existing keyframe-based animation systems.

In this paper, we introduce a sketch-based rotation editing system for enriching keyframe interpolated rotations. Our system modifies the rotation of a rigid object by editing its angular velocity trajectory. We choose to edit the angular velocity for two reasons. First, mathematically, given the initial orientation of the object and the trajectory of its angular velocity as a function of time, the object's entire rotational motion can be uniquely determined. This enables us to solve its altered rotational motion, satisfying a user's editing constraints by developing a variational algorithm for fitting an angular velocity trajectory. Second, angular velocity lies in a three-dimensional space and is easier to edit because of its direct connection with the object's visual rotation. An intuitive sketching interface is thus designed for users to conveniently edit angular velocity trajectories.

1.1 Related work

Our work is inspired by two lines of work: rotation editing and sketch-based modeling.

1. Rotation editing. Existing work on rotation editing focuses on smoothly interpolating rotations from a set of keyframe orientations of a rigid object. By generalizing spline curves in the Euclidean space to unit quaternions lying in the SO(3) space, Shoemake (1985) proposed to interpolate rotations using quaternion curves. The method, however, cannot guarantee C^2 continuity of the resulting spline curve. Kim *et al.* (1995) proposed an algebraic construction of spline curves for unit quaternions in SO(3) based on the exponential map and cumulative form of quaternions to resolve this problem. Different from the quaternion representation, the

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 61003145) and the Fundamental Research Funds for the Central Universities, China (No. 2009QNA5018)

©Zhejiang University and Springer-Verlag Berlin Heidelberg 2011

manifold spline technique in Hofer and Pottmann (2004) embeds the Euclidean motion of a rigid object in a 12-dimensional affine space and solves an optimal spline on a six-dimensional manifold to guarantee the orthogonality of rotation matrices. This method tends to produce shorter motion paths for points on the object. Another research direction is to impose physical constraints on a rotation animation. Barr *et al.* (1992) used angular velocity constraints at keyframes to constrain quaternion curves. This method is significantly accelerated through the use of quaternion splines (Ramamoorthi and Barr, 1997). Our paper goes one step further along this direction. Not only angular velocities at keyframes, but also angular velocities at any moment can be specified through a sketching interface. This makes it possible to add details to the smoothly interpolated rotational motion.

2. Sketch-based interfaces and modeling. Sketch-based interfaces have gained popularity in 3D modeling and editing applications. In the seminal work (Igarashi *et al.*, 1999), a sketch-based modeling system, Teddy, was introduced for creating 3D objects from 2D strokes. The system also allows users to edit the created 3D objects via extruding, cutting, and bending strokes. Sketching interfaces have also been successfully used in a variety of applications ranging from motion synthesis, to mesh deformation, hair modeling, tree modeling, pose designing, and mathematics visualization (Thorne *et al.*, 2004; Nealen *et al.*, 2005; Fu *et al.*, 2007; Chen *et al.*, 2008b; Bott and LaViola, 2010; Lin *et al.*, 2010), just to name a few. Our work extends the application scope of sketch-based interfaces to rotation editing. Note that rotation lies in the high dimensional SO(3) space, and one cannot directly sketch rotations. To circumvent this issue, we convert the rotation editing problem to the problem of fitting the angular velocity to a user-sketched curve.

1.2 Overview

The input to our system is a set of keyframe orientations of a rigid object. An initial temporal quaternion spline representing the object's rotation is constructed to interpolate the keyframe orientations. The user then specifies the target angular velocity trajectory using the sketching interface, and our algorithm optimizes the quaternion spline so that its angular velocity matches the target trajectory. Rotational variations are automatically added to the initial quaternion spline through the optimization. Note that the translational movement of the object's centroid is also part of the input to our system and remains unchanged during the optimization.

In our system, the user sketches the direction and magnitude of the angular velocity separately—the direction trajectory is specified on a unit sphere, and the magnitude trajectory is represented as a curve in a typical Cartesian coordinate with the horizontal axis representing the time and the vertical axis representing the magnitude. Fig. 1 illustrates the user interface of our system.

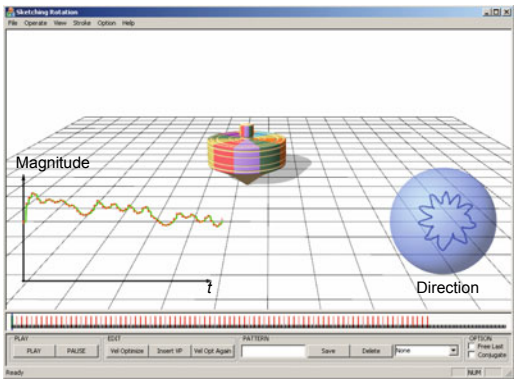

Fig. 1 Sketch-based rotation editing. Magnitude-time plane on the left: interface to sketch magnitude curves. Unit-sphere on the right: interface to sketch direction curves

Once the user sketches the target velocity trajectory, our system performs an optimization on the initial quaternion spline. Following the method described in Ramamoorthi and Barr (1997), a Hermite quaternion spline is adopted for the object's rotation. Quaternions and quaternion derivatives at its control points are optimized to make the angular velocity of this quaternion spline match the target angular velocity curve. We establish the relationship between the angular velocity ω and the unit quaternion q through a well-known formula $\omega = 2q'q^{-1}$, which means the user-sketched angular velocity is represented in the world coordinate system.

2 Fitting angular velocity

As mentioned, our input is a set of keyframe orientations of a rigid object, represented by unit quaternions. Let $q^i, i = 1, 2, ..., K$ denote the quater-

nion of keyframe i corresponding to time t_i. Suppose that the target angular velocity trajectory sketched by the user is $\omega^s(t)$. Our goal is to find a quaternion curve $q(t)$ that satisfies the keyframe quaternion constraints and whose angular velocity matches $\omega^s(t)$.

We solve the above problem by minimizing the following energy function:

$$E = \alpha W_{\mathrm{T}} + \beta W_{\mathrm{A}}, \tag{1}$$

where

$$W_{\mathrm{T}} = \int_{t_1}^{t_K} \left(\|q''(t)\|^2 - \frac{(\sum_{k=0}^{3} q_k(t) q_k{}''(t))^2}{\|q(t)\|^2} \right) \mathrm{d}t,$$

$$W_{\mathrm{A}} = \int_{t_1}^{t_K} \|2q'(t)q^{-1}(t) - \omega^s(t)\|^2 \mathrm{d}t,$$

subject to: $q(t_i) = q^i, \ \ i \in D,$

$$U = \int_{t_1}^{t_K} \left(1 - \|q(t)\|^2\right)^2 \mathrm{d}t = 0.$$

There are two terms in the function, W_{T} and W_{A}. W_{T} is the same as the tangential acceleration formulation in Barr *et at.* (1992) and Ramamoorthi and Barr (1997), which is used to generate smooth rotation paths. The angular velocity term, W_{A}, is designed to measure the deviation of the angular velocity from the user-sketched target velocity. α, β are weighting coefficients for the two terms. Since W_{T} and W_{A} have different quantities and very different values, the values of the coefficients need to be carefully chosen to balance the effects of the two terms. In our experiments, we fixed $\beta = 1$, and adjusted the value of α, and found that $\alpha \in [0.001, 0.005]$ works well. Larger α values generate smoother rotation paths that may deviate from the user-specified target velocity, while smaller α values produce well-matched angular velocity that may not be smoothly varying. For all examples in this paper, we set $\alpha = 0.002$. Keyframe orientations and the unity of quaternions are posed as hard constraints. The trajectory of target angular velocity, $\omega^s(t)$, is derived from the user's sketches—the rotation axis of the angular velocity is defined by the direction curve on the unit sphere, and the rotation speed is defined by the 1D magnitude curve. Note that only a subset D of the keyframe constraints are enforced though D can include all keyframes. During rotation editing, the user can choose to free the constraints at certain keyframes.

Now we turn to the optimization algorithm. Following Ramamoorthi and Barr (1997), we construct the optimal quaternion curve $q(t)$ using Hermite interpolation to reduce the computational cost. The optimization algorithm takes the quaternions and their derivatives at the control points, which are required for Hermite spline construction, as the unknown variables, and finds their optimal values using sequential quadric programming (SQP), which is a popular algorithm for nonlinear optimization (Gockenbach, 2003). Algorithm 1 lists the steps of the optimization algorithm.

Algorithm 1 Fitting angular velocity

Input: The unit quaternions at key-frames, $Q_0 = \{q(t_1), q(t_2), \ldots, q(t_K)\}$
Output: The optimized Q_n and their derivatives $Q_n{}'$, variable frames V_n and their derivatives $V_n{}'$
//initialize the quaternion curve
InitCurve($Q_0, Q_0{}'$);
$i \leftarrow 0$;
while (1) **do**
  //find current optimal values
  OptimizeSQP($Q_i, Q_i{}', V_i, V_i{}', \omega^s(t)$);
  if MaxDeviation() $\leq \delta$ **then**
    return;
  end if
  VariableFrames($Q_{i+1}, Q_{i+1}{}', V_{i+1}, V_{i+1}{}', \omega^s(t)$);
  $i \leftarrow i + 1$;
end while

The algorithm starts with computing a smooth quaternion curve as the initial solution of the optimization. Given the unit quaternions $Q_0 = \{q(t_1), q(t_2), \cdots, q(t_K)\}$ at the keyframes, we compute the initial quaternion curve using Hermite interpolation in function InitCurve. The time derivative of the quaternion curve at keyframe i is estimated using the formula $q'(t_i) = [q(t_{i+1}) - q(t_{i-1})]/(t_{i+1} - t_{i-1})$, and we denote the set of derivatives at the keyframes as $Q_0{}' = \{q'(t_1), q'(t_2), \cdots, q'(t_K)\}$.

With the prepared Q_0 and $Q_0{}'$, the optimization algorithm iteratively runs SQP to solve the optimal quaternion curve. In each iteration, it searches for the optimal quaternions and their derivatives at the keyframes and variable frames (described shortly), checks the current fitting error using function MaxDeviation, and figures out possible new variable frames. The algorithm ends when the maximum fitting error falls below a user-specified thresh-

old δ. We use $\delta = 0.1$ for all examples in this work.

Since the user-sketched angular velocity curve might have a complex shape, the number of original keyframes is likely to be insufficient for producing a velocity trajectory that matches the target curve well, which is a well-known issue in spline fitting. A common scheme to cope with this issue is to insert additional control points at the positions with a large fitting error. We follow this scheme to add additional control points, called variable frames V, to reduce the overall fitting error.

To determine the positions of variable frames, we first divide the entire time interval $\{t_1, t_K\}$ into small fragments with the same size $\triangle T$. At each fragment f_n, the fitting error is computed using the following formula:

$$F_n = \int_{t_n}^{t_n+\triangle T} \|2q'(t)q^{-1}(t) - \omega^s(t)\|^2 \mathrm{d}t. \qquad (2)$$

If F_n is greater than the threshold δ, the midpoint of the fragment is selected as the position of a new variable frame.

3 Experimental results

We have implemented our rotation editing system on a machine with an Intel quad-core 3.0 GHz CPU and 4 GB RAM, and tested its capability on various examples. Readers are encouraged to check the accompanying video (http://www.zjucadcg.cn/weng/rotation.wmv) for a demonstration of the animation results.

Table 1 lists the animation datasets used in this work. Note that if the input animation is long, it is a burden for a user to sketch a complete angular velocity trajectory. Our system thus allows the user to edit any segment of the animation independently.

Table 1 Animation data used in this paper

	Number of frames	Number of keyframes
M	240	49
Top	420	85
Hat	240	49
Leaves	800	81
Pen	121	25

Fig. 2 shows a simple animation of a letter M. The blue curve on the unit sphere in Fig. 2b shows the user-sketched angular velocity trajectory, and the fitting result is shown in red. Even though the input sketch is of complex shape, our optimization algorithm still fits it well. The optimization converges quickly in about 20 iterations. In Fig. 2a, the cyan letters in the background indicate the original animation, and the 3D characters in the foreground indicate the editing result. In this example, we also edit the magnitude of the angular velocity to enrich the animation. The edited animation can be found in the accompanying video.

Note that the number of inserted variable frames depends on the complexity of the freehand sketch. In all examples in this study, at most 10 variable frames are used to produce sufficiently accurate fitting results. The optimization algorithm takes on average 2 s to search for the optimal quaternion curve for a rotation animation with 20 keyframes, which is fast enough to provide interactive responses to the user.

Fig. 3 illustrates the editing of a spinning top. The input is a simple circular motion of a top on the ground with a constant angular velocity. A curve pattern is applied to the user-sketched trajectory to enhance the variations in angular velocity (Fig. 3a). The pattern is also drawn by the user, and may be synthesized from an example curve style using curve analogies (Hertzmann *et al.*, 2002). By matching the velocity curve on the unit sphere, variations are automatically added to mimic the wobbling effect when a top is spinning on a rough ground (Fig. 3c).

Fig. 4 shows the editing of a leaf animation. Due to the influence of the surrounding air, the rotation of the leaf can be quite complicated. With our system, the artist needs only to design the desired orientation at a sparse set of keyframes, and additional rotational details can be conveniently generated and incorporated using our sketching system.

Note that a lot of effort is required to create the animations shown in this paper using previous methods that set the orientations of a rigid object at a set of keyframes and smoothly interpolate the rotation. It is thus very difficult to directly compare our method with previous methods. We tried to use Maya's keyframe interpolation to produce the animation of spinning top shown in Fig. 3. Even after the attempt for an hour, the created animation still does not have the realistic wobbling effect as observed in our result, which was easily created in less than one minute.

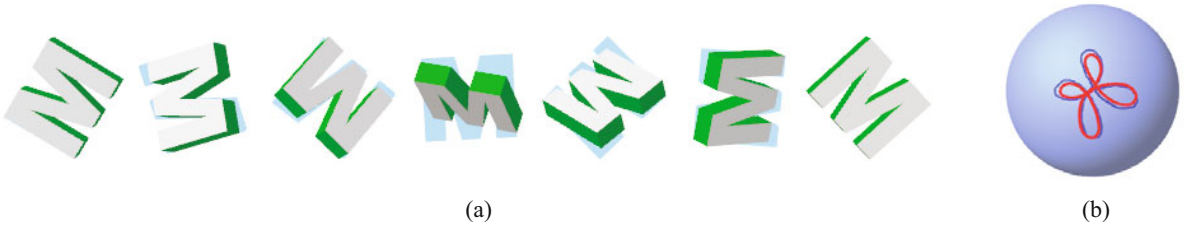

Fig. 2 Editing of the rotation part of an animation of a letter *M*. (a) The editing result. The original animation is shown in cyan. (b) The user's sketch (in blue) and the fitted angular velocity (in red)

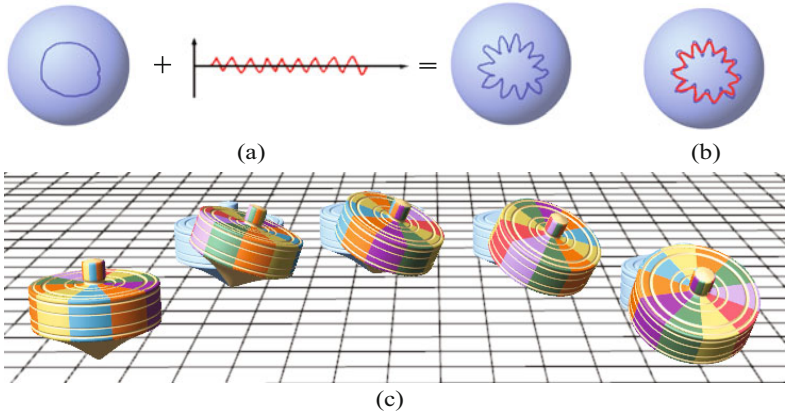

Fig. 3 Editing of a spinning top. In (a), a user-drawn pattern is applied to the user-sketched direction curve. In (b), the red curve indicates the fitting result. (c) is the editing result. The original animation is shown in cyan, and the editing result is shown in the foreground tops

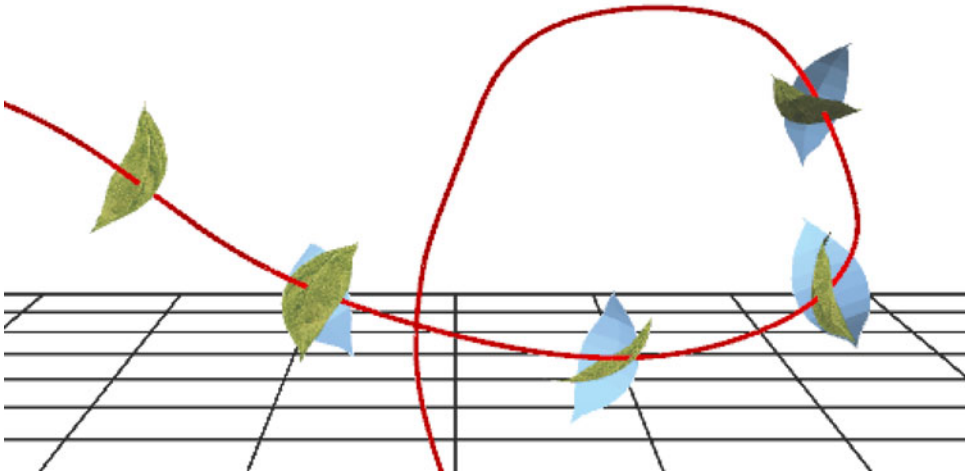

Fig. 4 Editing of a leaf animation. The curve in dark red shows the trajectory of the leaf's centroid. The original animation is shown in cyan

4 Conclusions

We have extended the application scope of sketch-based interfaces to rotation editing by developing a system for adding details to smoothly interpolated rotations. In our system, users edit rotations by sketching angular velocity curves. An optimization algorithm has been developed to solve the optimal rotational motion, which not only satisfies the keyframe orientation constraints, but also matches

the user-sketched angular velocity curve well. Our system is easy to implement, and is expected to be found useful in keyframe-based animation systems.

Acknowledgements

We would like to thank Prof. Kun ZHOU with Zhejiang University and the anonymous reviewers for their helpful comments.

References

Barr, A.H., Currin, B., Gabriel, S., Hughes, J.F., 1992. Smooth interpolation of orientations with angular velocity constraints using quaternions. *Comput. Graph.*, **26**(2):313-320. [doi:10.1145/142920.134086]

Bott, J., LaViola, J., 2010. A Pen-Based Tool for Visualizing Vector Mathematics. Eurographics Workshop on Sketch-Based Interfaces and Modeling, p.103-110.

Chen, X., Kang, S., Xu, Y., Dorsey, J., Shum, H.Y., 2008a. Sketching reality: realistic interpretation of architectural designs. *ACM Trans. Graph.*, **27**(2):11. [doi:10.1145/1356682.1356684]

Chen, X., Neubert, B., Xu, Y., Deussen, O., Kang, S., 2008b. Sketch-based tree modeling using Markov random field. *ACM Trans. Graph.*, **27**(5):109. [doi:10.1145/1409060.1409062]

Fu, H., Wei, Y., Tai, C.L., Quan, L., 2007. Sketching Hairstyles. Eurographics Workshop on Sketch-Based Interfaces and Modeling, p.31-36.

Gockenbach, M.S., 2003. Introduction to Sequential Quadratic Programming. Available from http://www.math.mtu.edu/∼msgocken [Accessed on Oct. 10, 2010].

Hertzmann, A., Oliver, N., Curless, B., Seitz, S.M., 2002. Curve Analogies. Proc. 13th Eurographics Workshop on Rendering, p.233-246.

Hofer, M., Pottmann, H., 2004. Energy-minimizing splines in manifolds. *ACM Trans. Graph.*, **23**(3):284-293. [doi:10.1145/1015706.1015716]

Igarashi, T., Matsuoka, S., Tanaka, H., 1999. Teddy: a sketching interface for 3D freeform design. *Comput. Graph.*, **33**(3):409-416. [doi:10.1145/311535.311602]

Kim, M.J., Kim, M.S., Shin, S.Y., 1995. A general construction scheme for unit quaternion curves with simple high order derivatives. *Comput. Graph.*, **29**(3):369-376. [doi:10.1145/218380.218486]

Lin, J., Igarashi, T., Mitani, J., Saul, G., 2010. A Sketching Interface for Sitting-Pose Design. Eurographics Workshop on Sketch-Based Interfaces and Modeling, p.1-8.

Nealen, A., Sorkine, O., Alexa, M., Cohen-Or, D., 2005. A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph.*, **24**(3):1142-1147. [doi:10.1145/1073204.1073324]

Ramamoorthi, R., Barr, A.H., 1997. Fast construction of accurate quaternion splines. *Comput. Graph.*, **31**(3):287-292. [doi:10.1145/258734.258870]

Shoemake, K., 1985. Animating rotation with quaternion curves. *Comput. Graph.*, **19**(3):245-254. [doi:10.1145/325165.325242]

Thorne, M., Burke, D., van de Panne, M., 2004. Motion doodles: an interface for sketching character motion. *ACM Trans. Graph.*, **23**(3):424-431. [doi:10.1145/1015706.1015740]