February 2013. Vol. 2, No.1 ISSN 2305-1493
International Journal of Scientific Knowledge
Computing and Information Technology

www.ijsk.org

MOTION ANALYSIS USING DISTANCE AND VELOCITY-TIME FUNCTION

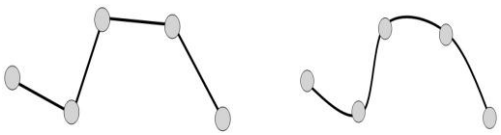
Md. Baharul Islam
Senior Lecturer, Department of Multimedia Technology and Creative Arts, Daffodil International University, Dhaka-1207, Bangladesh
E-mail: baharul@daffodilvarsity.edu.bd

Abstract- Motion is very easy between two points just like for straight line motion. Constant objects need acceleration for speed up at starting point and after reaching highest velocity, it needs to deceleration to stop at end point of the straight line. Suppose there are some turning points of a linear curve, then moving object along the curve will be very jerky at the turning points. This is very unlike how motion happens in the real world and this should be avoided in animations. The objective of this paper is to analysis motion for different kinds of curves that can be formed from same control points and analyze motion characteristics along a path for animation.

Keywords- Animation, motion, distance-time function, velocity-time function

1. INTRODUCTION

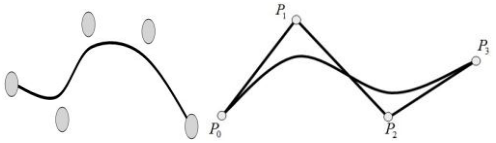
There are two types of curve linear and no-linear. Linear curves follow all of the sample points by straight line connection and non-linear interpolation follows all sample points but not as straight line. For interpolation, keys are the sample points of the curve that represents actual locations where curve should pass through all given sample points.



(a) (b)

Figure 1 (a) Linear and (b) Non-linear interpolation for five control points

From the figure 1, both curves are interpolating all key-frames. First curve is linear whereas second curve is non-linear and more realistic movement. We have another curve that is approximation where only two end points are interpolated. Other points are measuring the control the shape of the curve. Bezier and Hermite curve is one of the approximation curves.

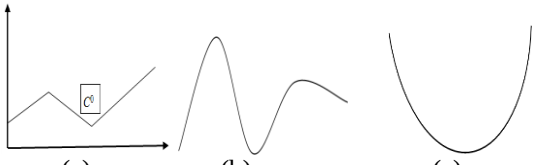


(a) *(b)*

Figure 2 Approximation curve using four control points

From figure 2, curves are not passed sample points exactly except two end points that is approximation curve. Smoothness of a curve is very important for a motion according to curve. How can we measure smoothness of a curve? Simply we can measure it by continuity of a curve. There is some continuity such as

- Positional continuity (C^0) is a small change in the value of the parameter always results in a small change in the value of the curve function.
- Tangential continuity (C^1) is a small change in the value of the parameter always results in a small change in the first derivative of the curve function.
- Curvature continuity (C^2) is a small change in the value of the parameter always results in a small change in the second derivative of the curve function.



(a) (b) (c)

Figure 3 (a) positional continuity (C^0) (b) tangential continuity (C^1) (c) Curvature continuity (C^2)

We can express a curve using three formats like

parametric, explicit and implicit that are given below respectively.

A conference paper was submitted on Global Engineering, Science and Technology 2012.

$$x = f(u) \qquad y = g(u), \qquad f(x,y) = 0, \quad y = f(x)$$

Parametric form of a equation are given below that can form a Bezier curve for four control points.

$$P(u) = \mathbf{a}u^3 + \mathbf{b}u^2 + \mathbf{c}u + \mathbf{d}$$

$$P(u) = P_0 b_0(u) + P_1 b_1(u) + P_2 b_2(u) + P_3 b_3(u)$$

$$P(u) = \sum_{i=0}^{3} P_i b_i(u)$$

Where P_0, P_1, P_2, P_3 are four control points that is shown in figure 2 and

$$b_0(u) = (1-u)^3$$
$$b_1(u) = 3u(1-u)^2$$
$$b_2(u) = 3u^2(1-u)$$
$$b_3(u) = u^3$$

The matrix form of the above parametric equation is

$$P(u) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}$$

$$P(u) = \mathbf{U}^T \mathbf{M}_B \mathbf{P}$$

The above matrix form equation is for Bezier curve. Bezier curve must have two properties like C^1 continuity and convex hull with local control. By constructing G1 quadratic Bézier curves that are satisfying given positions and arbitrary unit tangent vectors conditions (Gu, 2009). Bezier curves and Hermite curves are highly similar. The key difference between the two curves lies in what the user needs to specify to get the curve. In a Bezier curve, the control points specify the shape of the curve by defining a convex hull for the curve. The above equation P(u) and figure 2 shows how the parametric equation of the curve can be obtained if we know the 4 control points. (Ferdous, 2008) presents novel contributions to Bezier curve theory, with the introduction of quasi-Bezier curves, which seamlessly integrate localised control point information into the inherent global Bezier framework, with no increase in either the number of control point or order of computational complexity. A quadratic trigonometric Bézier curve with single shape parameter which is analogous to quadratic Bezier curve is introduced (Bashira, 2012) that is adjusted the shape of the curve as desired, by simply altering the values of shape parameter, without changing the control polygon. In Hermite curve, the control points specify the end points and the tangents to the end points. So the curve is defined as the simplest curve that passes through the end points and is tangential to these end points.

$$P(u) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_0 \\ P_1 \\ \mathbf{t_0} \\ \mathbf{t_1} \end{pmatrix}$$

$$P(u) = \mathbf{U}^T \mathbf{M}_H \mathbf{P}$$

2. LITERATURE REVIEW

The demand of motion is significantly increasing in animated movie industry. There are two threads for generating motion such as using examples and controllers. A system generates multiple motions (Arikan, 2002) that satisfied a given set of constraints. A knowledge-based methods approach that incorporate dynamics constraints and uses dynamics simulations to generate motion (Multon, 1999). An overview of the automatic motion planning (Latombe, 1999) and human walking or running can be found in (Bruderlin, 1996). An acting-based animation system for creating and editing character animation at interactive speeds are introduced by (Dontcheva, 2003). Motion analysis and interpolation synthesis (Li, Wiley, 2002, 1997) for articulated figures are researched by (Lee, Lamoure, 2000, 1996). There are many correlations between joint actions in human and animal motion. These correlations are especially clear for a repetitive motion like walking. There are some advantages of these relationships to synthesize degrees of freedom (Pullen, Kovar, 2002). A technique for interpolating between motions derived from live motion capture or produced through traditional animation tools introduced in (Rose, 1998). A new scheme for planning natural-looking locomotion of a biped figure (Choi, 2002) are generated to facilitate rapid motion prototyping and task-level motion generation. They need to provide start and goal positions in a virtual environment, this scheme gives a sequence of motions to move from the start to the goal using a set of live-captured motion clips. A composite curve made by two quadratic Bezier curves to satisfy the endpoint constraints with both positions and directions of unit tangent vectors. The composite curve has the flexibility to obtain different shape (Gu, 2009).

3. METHODOLOGY

One of the most common kinds of motion that we see in animation and robotics is ease in ease out motion. We were demonstrating ease in ease out motion and analyzed characteristics of motion for using distance-time function and velocity-time

function.

1.1. Distance-Time Function

This function will be C^1 continuous and velocity will be changed smoothly over time. If suddenly change the velocity of moving objects, then this curve is not C^1 continuous. It is easy to control the movement of object based on the velocity.

Traveled distance (t) = v (t).Δ (1)

Where v (t) is velocity at time t and Δt is time step.

We have to normalize the time t that is defined in the range of 0.0 and 1.0 and distance is normalized in the range of 0.0 to 1.0 with C^1 continuity in our experiment.

1.1.1. Ease function without constant speed

Ease function is the motion control function. It is starting slowly, speed up, and then slow down. It is used sine curve which is mapping the time t = 0.0 to 1.0 into the domain and Θ = -π/2 to π/2. We are mapping the range of the sine function -1.0 to 1.0 into the distance range of 0.0 to 1.0 that is shown in figure 4.

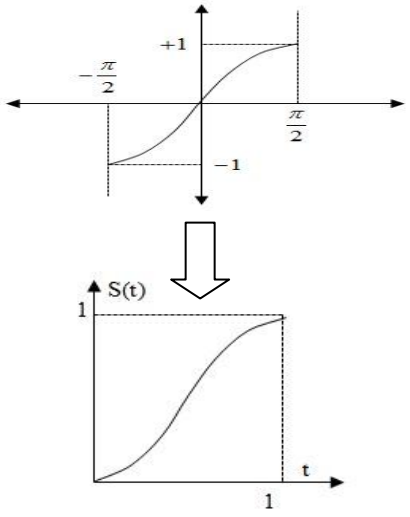


Figure 4 sine curves is working as ease function without constant speed

$$s(t) = \frac{\sin\left(t.\pi - \frac{\pi}{2}\right)+1}{2} \quad (2)$$

$$v(t) = \frac{\pi}{2}\cos\left(t.\pi - \frac{\pi}{2}\right) \quad (3)$$

From the velocity function, we will never have a constant speed over any interval using ease function from equation 3.

1.1.2. Ease function with constant speed

It is combined a sine curve and a line segment. From the figure 5, there are three sections. Section A works as acceleration of sine curve, section B works as constant velocity using line segment and section C is working as deceleration of sine curve. The slope of junction point marked as circle is 1.0.

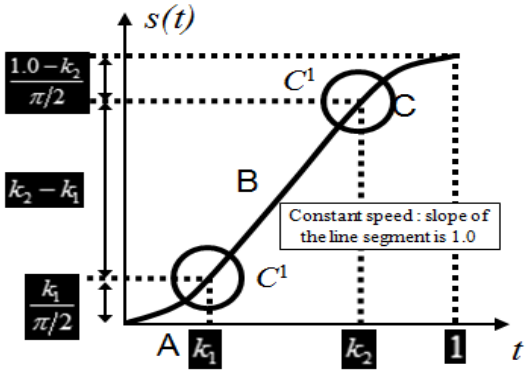


Figure 5 Ease function with constant speed

Section A: acceleration
Time t = 0.0 to k_1 into the domain Θ = -π/2 to 0.0
The range of the sine function -1.0 to 0.0 into the distance range of 0.0 to k_1/ (π/2)

Distance-time function:

$$s(t) = \frac{k_1}{(\pi/2)}\left(\sin\left(\frac{\pi}{2}\left(\frac{t}{k_1}-1\right)\right)+1\right), \quad 0 \le t \le k_1 \quad (4)$$

Velocity function:

$$v(t) = \cos\left(\frac{\pi}{2}\left(\frac{t}{k_1}-1\right)\right), \quad 0 \le t \le k_1 \quad (5)$$

Section B: constant speed
Line that passes through the locations

$$\left(k_1, \frac{k_1}{\pi/2}\right) \quad \left(k_2, \frac{k_1}{\pi/2}+k_2-k_1\right)$$

And the equation of a line segment

$$s(t) = mt + c \quad (6)$$

Distance-time function

$$s(t) = t + \frac{k_1}{(\pi/2)} - k_1, \quad k_1 \le t \le k_2 \quad (7)$$

Velocity function $v(t) = 1.0$ (8)

Section C: deceleration
Time t = k_2 to 1.0 into the domain Θ = 0.0 to

$$\frac{k_1}{(\pi/2)} + k_2 - k_1$$

π/2 and the range of the sine function 0.0 to 1.0 into the distance range of

$$\text{to } \frac{k_1}{(\pi/2)} + k_2 - k_1 + \frac{(1.0 - k_2)}{(\pi/2)}$$

Distance-time function

$$s(t) = \frac{1.0 - k_2}{(\pi/2)} \sin\left(\frac{\pi}{2}\left(\frac{t - k_2}{1.0 - k_2}\right)\right) + \frac{k_1}{(\pi/2)} + k_2 - k_1,$$

$$k_2 \le t \le 1.0 \qquad (9)$$

Velocity function

$$v(t) = \cos\left(\frac{\pi}{2}\left(\frac{t - k_2}{1.0 - k_2}\right)\right), \quad k_2 \le t \le 1.0 \qquad (10)$$

Total distance traveled is not 1.0.

$$\text{total distance} = \frac{k_1}{(\pi/2)} + k_2 - k_1 + \frac{1.0 - k_2}{(\pi/2)} \qquad (11)$$

Normalized distance traveled ease (*t*),

$$\text{ease}(t) = \frac{s(t)}{\frac{k_1}{(\pi/2)} + k_2 - k_1 + \frac{1.0 - k_2}{(\pi/2)}} \qquad (12)$$

1.2. Velocity-Time Function

Velocity is more intuitive than distance. It is easier to plan a motion by specifying speed up, slow down and constant velocity. Slope indicates acceleration and area under curve indicates distance traveled. We are used normalized time and distance. We had seen that how discontinuity in the path gave jerky and unrealistic motion. Similarly, if a moving body suddenly reaches its max velocity from an initial velocity of 0, this motion seems very unrealistic. A more realistic motion would be, if the body accelerated to that max velocity and instead of stopping suddenly after movement, it should slowly decelerate to a stop. This kind of motion will have a trapezoidal velocity time function as shown in figure 6.

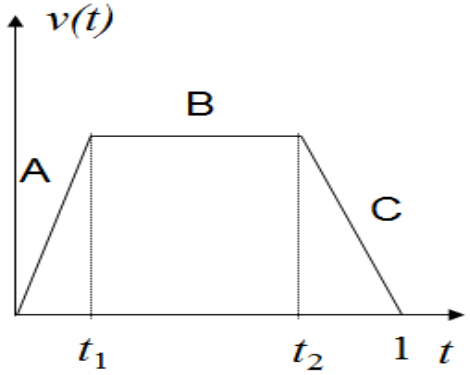


Figure 6 Ease using velocity-time function

As can be seen from the figure, there are also three parts to the motion A, B and C which refers to the period of acceleration, constant speed and deceleration respectively. The shape of the trapezoid is defined by three parameters, t_1, t_2 and v. The term 'v' is the maximum velocity; t_1 and t_2 are normalized times that define how long A, B and C last. Since the distance function of the object moving with this velocity is assumed to be normalized, the trapezoid ends at t= 1 and starts at t=0. The value of v (the maximum velocity) is between t_1 and t_2. All these values are interdependent and if two of them are specified the third can be obtained by using one of the following equations:

$$\text{area} = \frac{1}{2} v(t_1 + (1.0 - t_2)) + (t_2 - t_1)v$$

$$\text{area} = \frac{1}{2} v(1.0 + t_2 - t_1) \qquad (13)$$

With constraint

$$\frac{1}{2} v(1.0 + t_2 - t_1) = 1.0 \qquad (14)$$

$$t_1 = 1.0 + t_2 - \frac{2.0}{v} \qquad (15)$$

$$t_2 = \frac{2.0}{v} + t_1 - 1.0 \qquad (16)$$

$$v = \frac{2.0}{1.0 + t_2 - t_1} \qquad (17)$$

Once the values of all three parameters are obtained, we can find the value of the normalized distance using the following equations:

For $0 \le t \le t_1$: $\quad s(t) = v \cdot \frac{t^2}{2t_1}$

For $t_1 \le t \le t_2$: $\quad s(t) = v \cdot \frac{t_1}{2} + v \cdot (t - t_1)$

For $t_2 \le t \le 0$

$$s(t) = v \cdot \frac{t_1}{2} + v \cdot (t_2 - t_1) + \left(v - \left(\frac{v}{2} \cdot \frac{t - t_2}{1.0 - t_2}\right)\right) \cdot (t - t_2)$$

The smooth motion of a round object along the specified path can easily be traced using these three equations. It is possible to modify motion (Dontcheva, 2003) by changing the control points of curve. We obtained distance from the motion along the path equations are normalized distances. These need to be converted into the parameter which can be used in the equation so that we can find its position on the curve. In other words, we need to convert the value of s obtained into a value of u to be used in the curves. An efficient method for doing this would be to use an arc length table.

1.3. Controlling Motion

Specifically, a distance-time function s(t) is controlling the movement along the curve path. s(t) determines how far the object should travel at a time t. we specified a distance-time function s(t) and velocity-time function v(t) to control the movement along the curve path. We can control a motion using s(t).

Step 1: Starting time t = 0
Step 2: At t, determine u such that $L_{arc}(u) = s(t)$
Step 3: Put the object at $P(u)$
Step 4: Increment the time step, $t = t + \Delta t$
Step 5: Repeat until $t = t_{end}$

A parameterized arc length table is used to link distance travelled s to the value of the parameter u. We can better maintain (Gleicher, 2001) the dynamics of the motion using parameterization. Sampling points at regular intervals of u are taken and the value of s for each of these parameter values is found. For example, if we decide on a resolution of 0.05, then values of s for u=0, u=0.05, u =0.10 are found and stored in the form of a table. So once we obtain the value of the total distance travelled, we can approximate the value of u for that distance by finding the closest value of u from the table and then approximating using ratios and proportions. It is intuitive from the fact that we use linear interpolation that the accuracy of this method is highly dependent on resolution used.

4. EXPERIMENTAL RESULTS

Our developed system has been made in such a way that user can intuitively use as application. Helpful tool, tips, texts and status messages have been incorporated to improve usability. The application window looks as Figure 8.

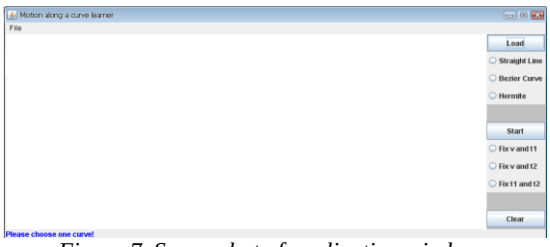


Figure 7 Screenshot of application window

There are four major panels in our system

- The menu bar on top which just has an option to see details of the system and to exit.
- A status bar to see what result your current action has done and to give advice on what is wrong.
- A button panel that helps the user chose options
- A drawing board where all the drawing and animation takes place.

By doing these steps the user can first take a look at the curve he constructed. He can then setup a motion. Once the motion is set up and started, the user can see the path followed by the ball and he can also see the rate at which the velocity is changing on a graph that is generated. Our developed system shows smooth motion along a created path for a round object in Figure 9.

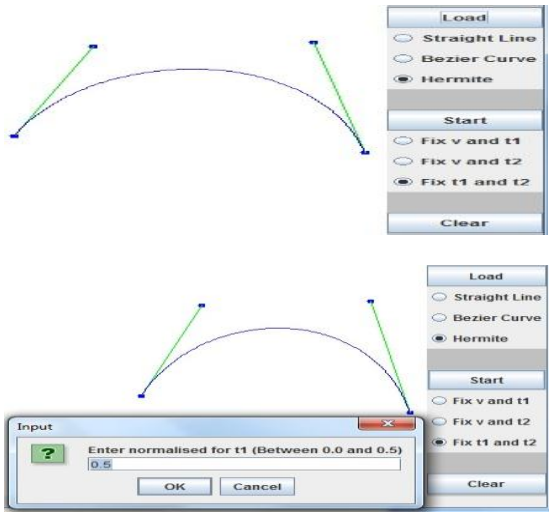


Figure 8 Different curve construction interfaces

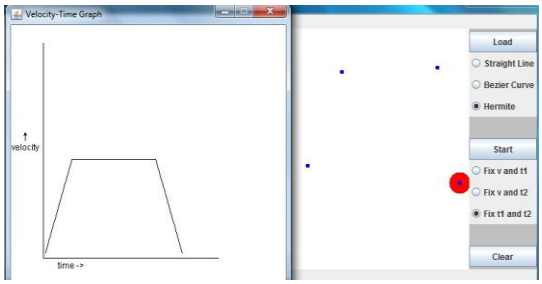


Figure 9 Velocity-time graphs for smooth motion along constructed curve

5. DISCUSSIONS

There were some interesting problems and challenges that we had to tackle while trying to develop a motion along a curve. We faced a challenge to design Hermite curve. The key difference of Hermite curve from a Bezier curve is in the way that it is specified. For creating a useful user interface, the best design have been for the user to first click on a starting and ending point and then interactively draw the tangents from the two points by dragging the mouse pointer. Besides being slightly more difficult to implement, we have to create entirely different

interfaces for a Hermite curve and a Bezier curve. Then the system would be easier to access for a first time user. There was another problem that the moving object would not reach the last control point (the end point). Instead it was stopping slightly behind this point. When a tweak was added to make it just end up at that point, the movement became jerky and unnatural. We realized that the reason for the incomplete and jerky motion was that the arc length resolution was too large. When we changed to a smaller value of resolution the movement became very smooth and natural.

6. CONCLUSIONS

For developing the motion along smooth curve, we used Java in Netbeans IDE and could be run on a computer where installed java any version. This paper will be helpful for researchers who want to work on realistic motion for animation.

References

1. Arikan O., Forsythe D. 2002 ‘Interactive motion generation from examples’, *In Proceedings of ACM SIGGRAPH*, Annual Conference Series.
2. Bashira U., Abasa M., Awangb M.N.H., Alia J.M. 2012 'The Quadratic Trigonometric Bézier Curve with Single Shape Parameter', *Journal of Basic and Applied Scientific Research*, Vol. 2, No. 3, pp. 2541-2546.
3. Bruderlin A., Calvert T. 1996 ‘Knowledge-driven, interactive animation of human running’, *In Graphics Interface*, Canadian Human-Computer Communications Society
4. Choi M.G. 2002 'Planning Biped Locomotion using Motion Capture Data and Probabilistic Roadmaps', *ACM Transactions on Graphics*, pp. 1-25.
5. Ferdous A.S. 2008 'Quasi-Bezier Curves Integrating Localised Information’, *Pattern Recognition*, Vol. 41, No. 2, pp. 531-542
6. Dontcheva M., Yngve G., Popovic Z. 2003 'Layered Acting for Character Animation', *ACM Transactions on Graphics*, pp. 1-8
7. Gleicher, M. 2001 ‘Motion path editing’, *In Proceedings 2001 ACM Symposium on Interactive 3D Graphics*
8. Gu H.J. 2009 'Constructing G1 Quadratic Bézier Curves with Arbitrary Endpoint Tangent Vectors', *International Journal of CAD/CAM*, Vol. 9, No. 1, pp. 55-60.
9. Kovar L., Gleicher M., Schreiber J. 2002 ‘Foot skate cleanup for motion capture editing’, *Technical report*, University of Wisconsin, Madison.
10. Lamouret A.,Panne M. 1996 ‘Motion synthesis by example’, *Computer animation and Simulation*, pp.199-212.
11. Latombe J.P. 1999 ‘Motion planning: A journey of robots, molecules, digital actors, and other artifacts’, In *International Journal of Robotics Research*, vol. 18, pp. 1119–1128.
12. Lee J. 2000 ‘A hierarchical approach to motion analysis and synthesis for articulated figures’, *PhD thesis*, Department of Computer Science, Korea Advanced Institute of Science and Technology.
13. Li Y., Wang T.,Shum H.Y. 2002 ‘Motion texture: A two-level statistical model for character motion synthesis’, In *Proceedings of ACM SIGGRAPH 2002*, Annual Conference Series
14. Multon F., France L., Cani M.P., Debunne G. 1999 ‘Computer animation of human walking: a survey’, *The Journal of Visualization and Computer Animation*, vol. 10, pp. 39-54.
15. Pullen K., Bregler C. 2002 ‘Motion capture assisted animation: Texturing and synthesis’, *In Proceedings of ACM SIGGRAPH 2002*, Annual Conference Series.
16. Rose C., Cohen M., Bodenheimer B. 1998 ‘Verbs and adverbs: Multidimensional motion interpolation’, *IEEE Computer Graphics and Application*, vol. 18, no. 5, pp. 32-40.
17. Wiley D., Hahn J. 1997 ’Interpolation synthesis of articulated Figure motion’, *IEEE Computer Graphics and Application*, vol. 17, no. 6, pp. 39-45