Journal of Modern Science and Technology
Vol. 1. No. 1. May 2013 Issue. Pp.84-95

Motion Control along Different Smooth Curves Using Arc Length Parameterization

Md. Baharul Islam

The easiest curve implementing is a linear curve which can be defined with the help of two end points. While this motion is the easiest, it is also practically one of the least used kinds of motion because of how unreal it looks. Let us consider a situation where we have 4 points and each of the 4 points is connected by straight lines. Now, if we have an object moving along these curves, the motion 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 virtual world. To avoid these issues we should use a smooth curve developed from these points for motion path. There are different kinds of smooth curves such as the Bezier curve, Hermite curve, B-splines, Catmull-Rom spline etc. They all vary in their complication, continuity at control points, ease of controlling etc. The objective of this paper is to control the motion along smooth curve by using arc length parameterization of a curve. This paper explains how ease in ease out motion along a curve which is one of the core concepts in animation.

Field of Research: Animation, Smoothness, Motion analysis, Distance-time function Velocity-time function, Digital Media, Arc Length Parameterization

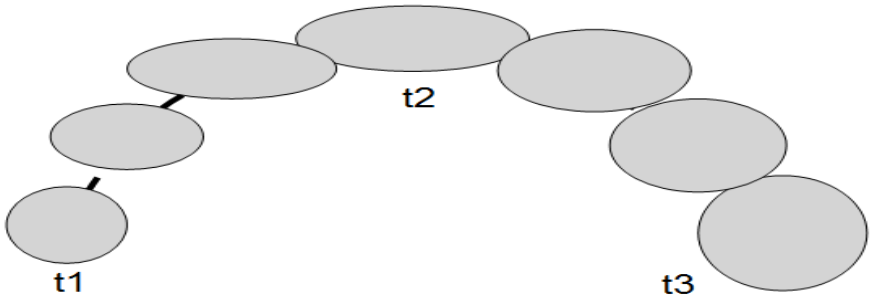
1. Introduction

'To animate' means to give life to an inanimate object, image or drawing. Proper modelling and rendering of still images or objects can give life. In order to animate something, the animator has to be able to specify directly or indirectly how the 'thing' has to move through time and space. Computer graphics parameters can be modified as functions of time. There are a lot of parameters that can be used in animation such as location, direction, shape, pose, texture coordinates, light parameters, camera parameters etc.

We can modify parameters using key-frame techniques. The term 'Key-framing' can be traced back to traditional hand animation technique. Key-framing requires so that the animator specify key positions for the objects and a well thought out plan of how the moving objects are going to behave over time. The computer then automatically fills in the missing frames by smoothly interpolating between those positions.

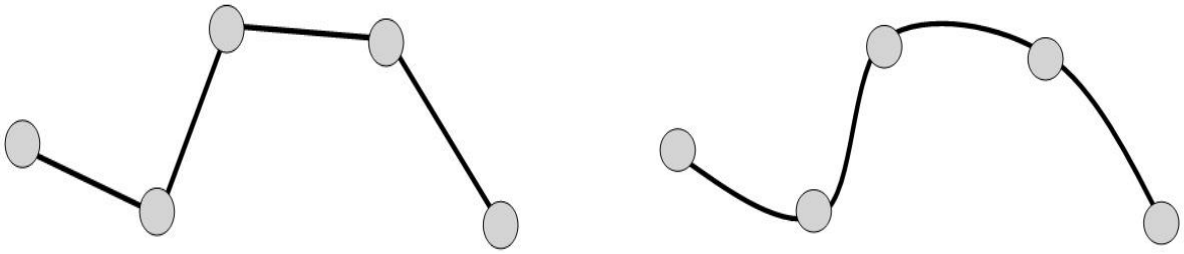
Senior Lecturer, Dept. of Multimedia Technology and Creative Arts, Daffodil International University, Dhaka-1207, Bangladesh Email: bahar_mag@yahoo.com or baharul@daffodilvarsity.edu.bd

Figure 1: Key-Frame for Animation



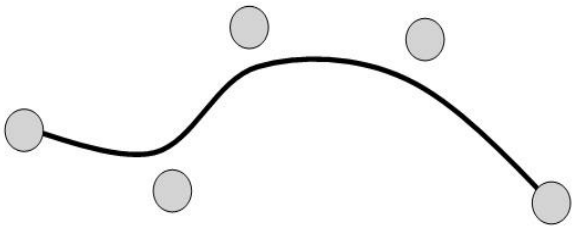
From the Figure 1, we took three key-frames at time t1, t2 and t3 for the object. Other key-frames are automatically created using interpolation method. Parameter of interpolation technique is distance that travelled by the object. We have to control interpolated values over time. We have to consider two things when interpolated objects such as location of objects and size of object. Smooth curve that means non-linear interpolation is better than straight line for realistic motion.

Figure 2: Linear and Non-Linear Interpolation System



For interpolation, keys are the sample points of the curve that represents actual locations where curve should pass through all given sample points. From the figure 2, both curves are interpolating all key-frames. First curve is linear whereas second curve is non-linear and more realistic movement. Another curve is approximating 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 that is shown in figure 3.

Figure 3: Approximation Curve Using Control Points



Smoothness of a curve is very important for a motion along curve. There is some continuity that can measure how curve will be smoothness. 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 (Liang, X Zhang, C Zhong, L & Liu, Y 2005). 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 (Kui, F Yue, S Juan, W & Quanyuan, W 2009).

We can express a curve using three formats like parametric, explicit and implicit that are given below respectively.

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

Parametric form of the equation are given below

$$P(u) = \mathbf{a}u^3 + \mathbf{b}u^2 + \mathbf{c}u + \mathbf{d}$$
$$P(u) = P_0 b_0(u) + P_1 b_1(u) + P_2 b_2(u) + P_3 b_3(u)$$
$$P(u) = \sum_{i=0}^{3} P_i b_i(u)$$

Where P_0, P_1, P_2, P_3 are four control points that mentioned 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

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

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

Bezier curve must have two properties like C^1 continuity and convex hull with local control. Bezier curves and Hermite curves are highly similar. By constructing G1 quadratic Bézier curves that are satisfying given positions and arbitrary unit tangent vectors conditions (Gu, 2009). 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.

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

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

Parametric spatial curve used to define the route of an object. Arc-length computation is necessary for motion control along a curve. Actually it is controlling the speed at which the curve is traced. The analytic approach to computing arc length

$$A(t) = \int_{t_0}^{t} \sqrt{(dx)^2 + (dy)^2 + (dz)^2}$$

Arc length is a geometric integration (Guenter, 1990). Numerical approaches for arc-length computation that divides the range *[t_0, t]* into intervals. There are two ways to compute arc length within each interval such as Gaussian quadrature and Simpson's rule. Arc length within range *[t_0, t]* computed as the sum of arc length within each interval. We can compute arc length within one interval by using adaptive method. For computing arc-length, we need build a table that divide the parameter range into intervals (Walter, 1996), compute arc length within each interval, and build a table of correspondence between parameter and arc length. It has another option to map parameter to arc length which map parameter to an interval, finding arc-length value before this interval from the table and compute the arc length within part of the interval. This paper organized following by literature review, methodological analysis of works, findings, discussion of challenges with conclusion.

2. Related Works

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). Several methods have been developed to approximate arc length parameterization, some of which are based on curve dependent tables of data, while others are not. It is difficult to obtain a meaningful comparison for methods which are not of the same class. In some applications such as graphical simulation and animation, where the animated object is to appear at approximately evenly spaced intervals for smooth appearance, it is the performance rather than the accuracy that is of importance (Madi, 2004). The simplest method may be called the basic parametric flow (BPF), since a number of N points are produced by uniform parameter spacing. Although the method is simple and fast, it is well known that it is not suitable for generating points along the arc length of a curve (Madi, 2004). The method described by Sharpe and Thorne can accurately produce arc-lengths (Sharpe and Thorne, 1982). However, it has a high computational cost associated with "extracting" the corresponding parametric value. Optimal Parameterization (OP) is mathematically a rather intricate process. The given polynomial curve Q(t) is first transformed into an equivalent rational form by transforming the parameter t.

3. Methodological Analysis

One of the most common kinds of motion that we see in animation and robotics is ease in ease out motion. We are considered only the motion of a round object. We concentrate on showing ease in ease out motion and demonstrating its effects on motion.

3.1 Distance-time Function

Distance-time function is usually monotonic in time. This function will be C^1 continuous. 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.

$$\text{traveled\_distance}(t) = v(t) \cdot \Delta t$$

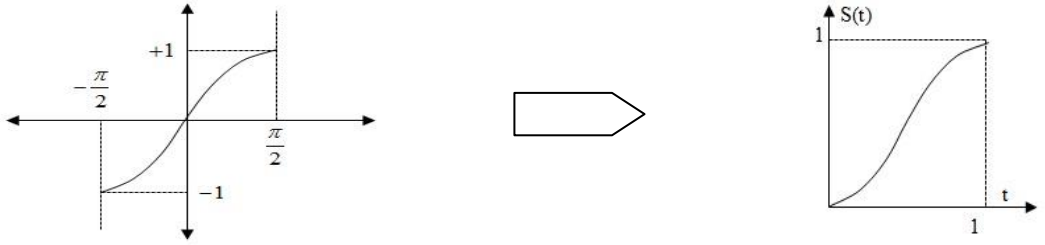
Where v(t) is velocity at time t and Δt is time step.

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.

3.1.1 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.

Figure 4: Sine Signal is working as Ease Function without Constant Speed



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

From the velocity function, we can see from figure 4 that using this ease function, we will never have a constant speed over any interval.

3.1.2 Ease Function with Constant Speed

It is combined a sine curve and a line segment. 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 is 1.0.

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

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

Section B: Constant Speed

Line that passes through the locations $\left(k_1,\frac{k_1}{\pi/2}\right)$ and $\left(k_2,\frac{k_1}{\pi/2}+k_2-k_1\right)$

Equation of a line segment $s(t)=mt+c$

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

Velocity function $v(t)=1.0$

Section C: Deceleration

Time $t = k_2$ to 1.0 into the domain Θ = 0.0 to π/2 and the range of the sine function 0.0 to 1.0 into the distance range of $\frac{k_1}{(\pi/2)} + k_2 - k_1$ to $\frac{k_1}{(\pi/2)} + k_2 - k_1 + \frac{(1.0 - k_2)}{(\pi/2)}$

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

$k_2 \le t \le 1.0$

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

Total distance traveled is not 1.0.

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

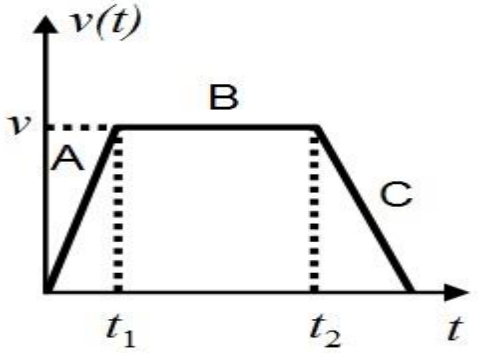
Normalized distance traveled ease (t),

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

3.2 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 (Steinbuch, 1998). This kind of motion will have a trapezoidal velocity time function as shown in figure 5.

Figure 5: 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:

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

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

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

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

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. But we still have one problem remaining. 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.

3.3 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 (Verscheure, 2006). 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 $t = t_{end}$

3.4 Arc Length Parameterization

Generally, the relationship between the changes in parameter u and the distance traveled is non-linear. It is very difficult for animator to predict and control the speed

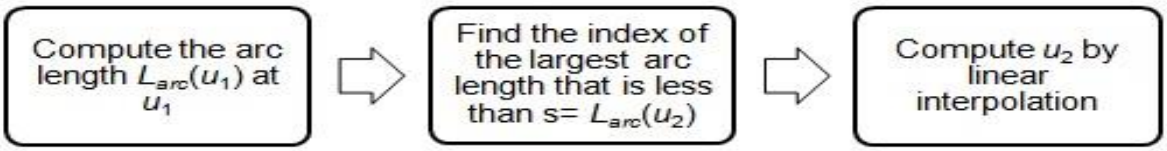
of an object by altering the parametric value of the curve and difficult to attain a constant speed. Parameterizing a curve depends on the arc length of the curve. There are two possible ways (analytic method, forward differencing) to find arc length and evaluate last parameter using first parameter.

- Given the parameters u_1 and u_2, determine the arc length between u_1 and u_2.
- Given an arc length s and parameter u_1, evaluate u_2 such that the arc length between u_1 and u_2 is s

$$S = \int_{u_1}^{u_2} \Delta S \qquad S = \int_{u_1}^{u_2} \left| \frac{dP}{du} \right| du$$

Where, Parametric curve $P(u)$, arc length from a point $P(u_1)$ to another point $P(u_2)$, dP/du tangent vector, integrate the length of the tangent vector times du. Consider the arc length ΔS between point $P(u)$ to another point $P(u+du)$ that approximately straight line. Approximate the arc length works as the length of the line. Calculating S requires evaluating the arc length integral. (Guenter, 1990) suggest using an adaptive application of Gaussian quadrature. The results presented here are found by a similar method, using integration. Instead of applying the integral to the entire curve, it is applied to each polynomial segment of the curve, and the results are summed. 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. The figure 6 represents the arc length parameterization procedures.

Figure 6: Steps on Arc Length Parameterization



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. 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 7 and velocity-time graph in figure 8.

Figure 7: Different Curve Construction Interfaces

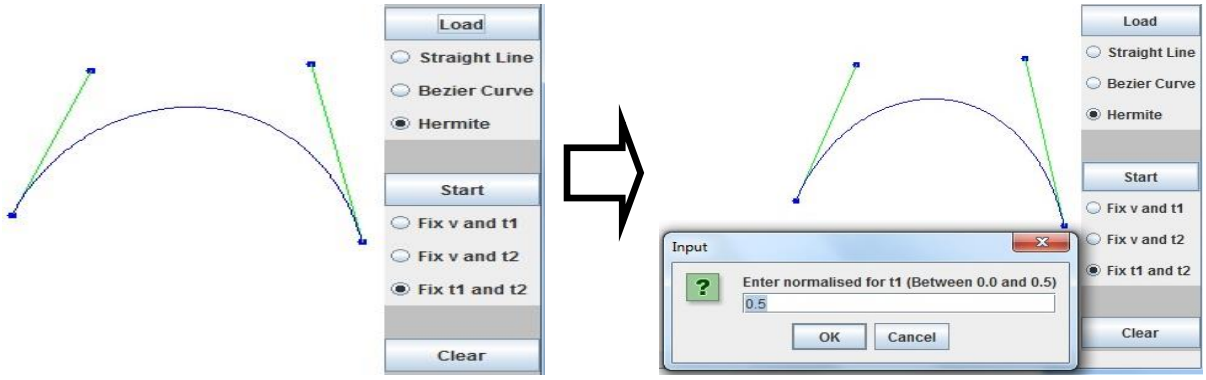
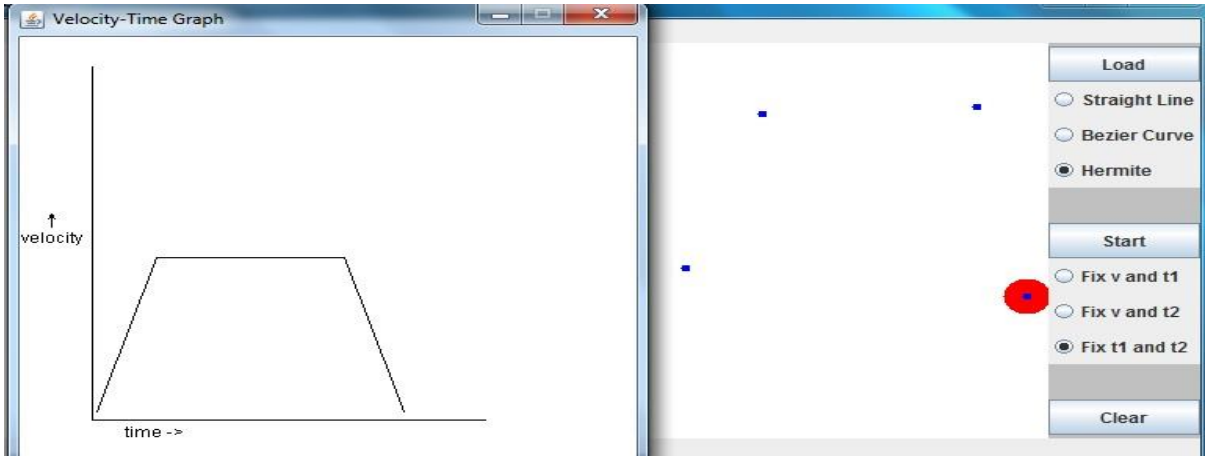


Figure 8 Velocity-Time Graphs for Smooth Motion along Constructed Curve



5. Conclusion

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. For developing this system, 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 analysis. We analysed only two types of smooth curves. In future we will work with shape motion along different types of curves.

References

Arikan, O., and Forsythe, D. (2002), 'Interactive motion generation from examples', *In Proceedings of ACM SIGGRAPH*, Annual Conference Series.,

Aung, C.H., Lwin, K.T., and Myint, Y.M. (2008), 'Modeling Motion Control System for Motorized Robot Arm using MATLAB', *World Academy of Science, Engineering and Technology*.

Bashira, U., Abasa, M., Awangb, M.N.H., and Alia, J.M. (2012), 'The Quadratic Trigonometric Bézier Curve with Single Shape Parameter', *Journal of Basic and Applied Scientific Research*, 2 (3), 2541-2546.

Borenstein, J., and Koren, Y. (1986), 'Motion control analysis of mobile robot', *Transactions of ASME, Journal of Dynamics, Measurement and Control*, 109 (2), 73-79.

Bruderlin, A., and Calvert, T. (1996), 'Knowledge-driven, interactive animation of human running', *In Graphics Interface*, Canadian Human-Computer Communications Society.

Choi, M.G. (2002), 'Planning Biped Locomotion using Motion Capture Data and Probabilistic Roadmaps', *ACM Transactions on Graphics,* pp. 1-25. Ferdous, A. S. (2008), 'Quasi-Bezier Curves Integrating Localised Information', *Pattern Recognition*, 41 (2), 531-542.

Dontcheva, M., Yngve, G., and Popovic, Z. (2003), 'Layered Acting for Character Animation', *ACM Transactions on Graphics*, pp. 1-8.

Gleicher, M. (2001), 'Motion path editing', *In Proceedings 2001 ACM Symposium on Interactive 3D Graphics.*

Gu, H.J. (2009), 'Constructing G1 Quadratic Bézier Curves with Arbitrary Endpoint Tangent Vectors', *International Journal of CAD/CAM*, 9 (1), 55-60.

Guenter, B., and Parent, R. (1990), 'Computing the Arc Length of Parametric Curves, *IEEE Computer Graphics and Applications.*

Kovar, L., Gleicher, M., and Schreiber, J. (2002), 'Foot skate cleanup for motion capture editing', *Technical report*, University of Wisconsin, Madison. Kui, F., Yue, S., Juan, W., and Quanyuan, W. (2009), 'C^2 Nearly Arc-length Parameterizations', *Proceedings of the International Symposium on Information Processing,* Huangshan, China, August 21-23, pp. 437-440.

Lamouret, A., and Panne, M. (1996), 'Motion synthesis by example', *Computer animation and Simulation*, pp.199-212. Latombe, J.P. (1999), 'Motion

planning: A journey of robots, molecules, digital actors, and other artifacts', In *International Journal of Robotics Research*, 18, 1119–1128.

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.

Li, Y., Wang, T., and Shum, H.Y. (2002), 'Motion texture: A two-level statistical model for character motion synthesis', In *Proceedings of ACM SIGGRAPH*, Annual Conference Series.

Liang, X., Zhang, C., Zhong, L., and Liu, Y. (2005), 'C^1 Continuous Rational Re-parameterization Using Monotonic Parametric Speed Partition', *9th International Conference on Computer Aided Design and Computer Graphics*, pp. 2473-2477.

Madi, M. (2004), 'Closed-form expressions for the approximation of arc-length parameterization for the Bezier curves', *International Journal of Applied Mathmatics and Computer Science*, vol. 14 (1), 33–41.

Multon, F., France, L., Cani, M.P., and Debunne, G. (1999), 'Computer animation of human walking: a survey', *The Journal of Visualization and Computer Animation,* 10, 39-54.

Munoz, V.F., Gomez-De-Gabriel, J.M., Garcia-Morales, I., Fernandez-Lozano, J., and Morales, J. (2005), 'Pivoting motion control for a laparoscopic assistant robot and human clinical trials', *Journal of Advanced Robotics*, 19 (6), 695–713, <www.vsppub.com>

Pullen, K., and Bregler, C. (2002), 'Motion capture assisted animation: Texturing and synthesis', *In Proceedings of ACM SIGGRAPH*, Annual Conference Series.

Rose, C., Cohen, M., and Bodenheimer, B. (1998), 'Verbs and adverbs: Multidimensional motion interpolation', *IEEE Computer Graphics and Application,* 18 (5), 32-40.

Steinbuch, M., and Norg, M.L. (1998), 'Advanced Motion Control: An Industrial Perspective, *European Journal of Control*, 4, 278-293.

Verscheure, D., Paijmans, B., Brussel, H.V., and Swevers, J. (2006), 'Vibration and motion control design and trade-off for high-performance mechatronic systems' *Proceedings of the 2006 IEEE International Conference on Control Applications* Munich, Germany, pp. 1115-1120. Walter, M., and Fournier, A. (1996), 'Approximate Arc Length Parameterization' *Department of Computer Science*, The University of British Columbia, pp. 1-9.

Wiley, D., and Hahn, J. (1997), 'Interpolation synthesis of articulated Figure motion', *IEEE Computer Graphics and Application,* 17 (6), 39-45.