Curve Morphing by Weighted Mean of Strings

Xiaoyi Jiang¹, Horst Bunke², Karin Abegglen², Abraham Kandel³
¹Department of Electrical Engineering and Computer Science, Technical University of Berlin, Germany
²Department of Computer Science, University of Bern, Bern, Switzerland
³Department of Computer Science and Engineering, University of South Florida, Tampa, FL, USA

Abstract

In this work we propose a novel approach to curve morphing. We represent curves by strings, i.e. sequences of symbols. The curve morphing problem is formulated as that of computing a weighted mean of two strings, which is then solved by a corresponding algorithm recently reported in the literature. Although only 2D curves are used in our experiments, the approach is applicable to curves in any dimension. Curve morphing has interesting applications in computer graphics, industrial design, robotics, etc. Additionally, it provides an alternative multiscale representation of curves.

1 Introduction

Shape morphing, also known as metamorphosis, shape interpolation, shape blending, and shape evolving, is to compute a continuous transformation from one shape (the source) to another (the target). It has widespread applications in computer graphics and industrial design. Recently, it has been also found to be useful in robotics [12]. Morphing algorithms have been developed for 2D image space, 3D voxel space, curves, and polyhedra. In this work we are concerned with curves. Although only 2D curves will be used to conduct experiments, our approach is applicable to curves in any dimensions.

There are two classes of curve morphing algorithms; see [4] for a general discussion. The first class [3, 10, 11] is based on an explicit point correspondence process which establishes the links between the source and the target points. In a so-called point path process, the source points travel along some path (trajectory) towards their corresponding target points. The second class of algorithms [5] transform the problem of curve interpolation to that of functional interpolation in terms of a partial differential equation. Several mathematical tools including regularization can then be applied to find solutions.

From the methodology point of view our curve morphing approach is very different from those known from the literature. It is inspired by the weighted mean of two points which is a basic concept from the n-dimensional real space. In this paper the notion of weighted mean is adopted from the n-dimensional real space to the domain of strings, in which our curves are represented. Finally, the curve morphing problem is formulated as that of computing a weighted mean of two strings and solved by a corresponding algorithm recently reported in the literature.

2 Morphing in arbitrary space

Given a source point \( p_1 \) and a target point \( p_2 \) in the n-dimensional real space, we have an infinite number of possibilities to transform (morph) \( p_1 \) into \( p_2 \). A reasonable choice is to use linear interpolation

\[ p(t) = (1 - t)p_1 + tp_2, \quad t \in [0, 1]. \]

Each intermediate point \( p(t) \) is called a weighted mean of \( p_1 \) and \( p_2 \), and satisfies

\[ d_E(p(t), p_1) = t \cdot d_E(p_1, p_2) \]
\[ d_E(p(t), p_2) = (1 - t) \cdot d_E(p_1, p_2) \]

where \( d_E \) represents the Euclidean distance of two points. Clearly, \( p(t = 0.5) \) corresponds to the (normal) mean of \( p_1 \) and \( p_2 \).

Based on the concept of weighted mean we can define a morphing transformation for an arbitrary space \( U \) in a similar manner. Given some distance function \( d(x, y), x, y \in U \), the morphing process transforms a source object \( p_1 \) to a target object \( p_2 \) through a series of objects \( p(t), t \in [0, 1], \) such that the properties

\[ d(p(t), p_1) = t \cdot d(p_1, p_2) \]
\[ d(p(t), p_2) = (1 - t) \cdot d(p_1, p_2) \]

hold.

Under this universal framework of morphing, the fundamental problem in a particular morphing task is the computation of weighted mean in the corresponding object space.
This has been studied in contexts beyond the n-dimensional real space, including strings [2] and graphs [1]. In the current work we will make use of the weighted mean of strings to accomplish curve morphing. For this purpose we need a string representation of curves and a distance function for the string space representing curves.

3 String representation of curves

A 2D curve is given by a sequence $C = (x_1, y_1) \ldots (x_n, y_n)$ of points in the $xy$-plane. For closed curves we assume that a starting point is specified in some way. (The issue of closed curves will be further discussed in Section 5.) To map such a curve into a string, we first sample the given data points such that the Euclidean distance between any pair of two consecutive points has a constant value $\Delta$. That is, $C$ is transformed into $\overline{C} = (x_1, y_1) \ldots (x_m, y_m)$, where $d_E((x_i, y_i), (x_{i+1}, y_{i+1})) = \Delta$ for $i = 1, \ldots, m-1$. Then, a string $z_1 \ldots z_{m-1}$ is generated from $\overline{C}$ where $z_i$ is the vector pointing from $(x_i, y_i)$ to $(x_{i+1}, y_{i+1})$.

Instead of a sequence of vectors as described above, a chain code representation could be used as well. But the representation actually used has a number of advantages over the chain code. First, it provides a higher angular resolution. Secondly, as all line segments used for the approximation are of the same length, the problem of shape distortion under rotation is avoided. Thirdly, because the length of the line segments is a parameter, the curve sampling resolution can be chosen by the user.

As the distance function $d(p_1, p_2)$ measuring the dissimilarity of two strings (curves) $p_1$ and $p_2$ we choose the well-known Levenshtein string distance [13] using edit operations. Three edit operations are allowed: deletion ($a \rightarrow \epsilon$), insertion ($\epsilon \rightarrow a$), and substitution ($a \rightarrow b$). Their costs are defined as: $c(a \rightarrow \epsilon) = c(\epsilon \rightarrow a) = |a| = \Delta$, $c(a \rightarrow b) = |a - b|$. Notice that the minimum cost of a substitution is equal to zero (if and only if $a = b$), while the maximum cost is $2\Delta$. The latter case occurs if $a$ and $b$ are parallel and have opposite direction. The distance $d(p_1, p_2)$ is the minimum cost taken over all sequences of edit operations that transform $p_1$ into $p_2$.

4 Weighted mean of strings

Given the string representation of curves we need an algorithm for actually computing the weighted mean of two strings $p_1$ and $p_2$ for some $t$ value to conduct curve morphing. Here we briefly describe such an algorithm; see [2] for full details. The distance of two strings can be computed by a dynamic programming technique [13]. As a by-product of the computation, we also obtain the optimal sequence of edit operations with associated cost $d(p_1, p_2)$ for transforming $p_1$ into $p_2$. Then, the following theorem provides the theoretical foundation of weighted mean string computation.

**Theorem 1** Let $p_1$ and $p_2$ be two strings, $S$ an optimal sequence of edit operations transforming $p_1$ into $p_2$. For any subsequence $S'$ of $S$ with the corresponding cost $\gamma = \sum_{e \in S'} c(e)$ we can construct a string $q$ by applying $S'$ to $p_1$. Then, we have $d(q, p_1) = \gamma$ and $d(q, p_2) = d(p_1, p_2) - \gamma$.

This theorem suggests a computational procedure as follows. To compute a weighted mean of two strings $p_1$ and $p_2$, we first compute their distance $d(p_1, p_2)$. From the optimal sequence $S'$ of edit operations we select a subsequence of edit operations, $S''$, of $S$ with cost $\gamma$ and apply $S''$ to $p_1$. The resulting string $q$ satisfies

$$d(q, p_1) = \gamma = t \cdot d(p_1, p_2)$$

$$d(q, p_2) = (1 - t) \cdot d(p_1, p_2)$$

with $t = \frac{\gamma}{d(p_1, p_2)}$. This is the basic procedure for computing the weighted mean of strings; further details can be found in [2].

For the purpose of curve morphing it suffices to compute the weighted mean of a source string (curve) $p_1$, and a target string (curve) $p_2$ for a series of values $t = \frac{1}{k}, \frac{2}{k}, \ldots, \frac{k}{k}$. This procedure will generate a total of $k$ intermediate curves.

5 Handling of closed curves

In dealing with closed curves we are faced with two problems. First, we have to choose a starting point on each curve. While an arbitrary selection on the first curve works well, the starting point on the second curve should be chosen with some care. For a fixed starting point on $p_1$ we may consider each point on $p_2$ in turn as a potential starting point and compute the distance $d(p_1, p_2)$. The optimal starting point on $p_2$ is then the point with the smallest distance value. This is the cyclic string matching problem for which efficient algorithms have been reported in the literature [7, 8]. For simplicity we will manually specify the starting points in our experiments described in Section 6.

Another problem arises when one directly applies the curve morphing algorithm described above. The computed intermediate curves are not necessarily closed. In the following we discuss two solutions for this problem. Let a computed intermediate string (curve) $C$ consist of $n$ vectors $(\Delta \cos \alpha_1, \Delta \sin \alpha_1) \ldots (\Delta \cos \alpha_n, \Delta \sin \alpha_n)$, where $\alpha_k, 1 \leq k \leq n$, represents the angle between the $k$-th vector and the $x$-axis. For notational simplicity we assume the starting point being the origin of the coordinate
system. Then, the $k$-th point of $C$ is $(x_k, y_k) = (\Delta \cdot \sum_{i=1}^{k} \cos \alpha_k, \Delta \cdot \sum_{i=1}^{k} \sin \alpha_k)$. The last point $(x_n, y_n)$ of $C$ is generally not $(0, 0)$. To make $C$ a closed curve we may add an additional small angle $\beta$ to each $\alpha_k$, resulting in a new position of the $k$-th point $(x'_k, y'_k) = (\Delta \cdot \sum_{i=1}^{k} \cos(\alpha_k + k\beta), \Delta \cdot \sum_{i=1}^{k} \sin(\alpha_k + k\beta))$. The closedness of $C$ is guaranteed by $(x'_n, y'_n) = (0, 0)$. Unfortunately, finding $\beta$ is a non-linear optimization problem, i.e. finding a solution of $(x'_n)^2 + (y'_n)^2 = 0$.

A second solution is based on the fact that only the amount $(-x_n, -y_n)$ is missing for the last point of $C$ to reach the origin. If we introduce a small correction $(-\Delta, -\Delta)$ to each vector, the $k$-th point of $C$ will take the new position $(x_k', y_k') = (x_k - k \cdot \Delta, y_k - k \cdot \Delta)$. In this case the last point $(x'_n, y'_n)$ corresponds to the origin. This second solution is very simple. However, the corrected closed curve usually has a non-uniform sampling in contrast to $C$ itself. Fortunately, this is not a real disadvantage since, if necessary, we can perform the sampling process again to achieve the uniform sampling density $\Delta$. Because of its simplicity we use this second correction procedure to generate closed intermediate curves.

6 Experimental results

Experiments have been carried out using a large database of fish shapes\footnote{Available at \url{ftp://ftp.ee.surrey.ac.uk/pub/vision/misc/fish\_contours.tar.Z}} from public domain and some other curves scanned by ourselves. In all experiments the sampling density was fixed to be $\Delta = 2.0$ (pixels). Figure 1 illustrates the morphing process of two fish shapes, where a total of ten intermediate curves are generated and shown in a top-down and left-right order. A second example, morphing of two hand shapes, is presented in Figure 2. Note that we apply a Gaussian filtering with a small kernel to smooth the curves. The fish shapes are all closed curves. To ensure the closedness of the intermediate curves we use the simple closing operation described in the last section.

It should be pointed out that our approach cannot guarantee intersection-free curve morphing, see Figure 3. But this is an inherent problem of almost all curve morphing algorithms reported in the literature, especially in the case of curves of complex shape. As a matter of fact, the work [6] seems to be the only one which is guaranteed to be intersection-free.

Traditional multiscale curve representations are based on Gaussian smoothing [9]. Curve morphing can be used for an alternative multiscale curve representation with the original curve as source and a circle as target [5]. Figure 4 shows such an example. In our experiments the radius of the circle is fixed to be $\sqrt{\frac{A}{\pi}}$ where $A$ represents the area of the target shape, forcing the target circle to have the same area as the source curve.

7 Conclusion

In this paper we have proposed a novel approach to curve morphing. We represent curves by strings. The curve morphing problem is formulated as that of weighted mean of two strings, which is then solved by a corresponding algorithm recently reported in the literature. Although only 2D
curves have been used to conduct experiments, it is obvious that our approach is applicable to curves in any dimensions. In addition to applications in computer graphics, industrial design, robotics, etc, curve morphing provides an alternative multiscale representation of curves.

Acknowledgments

The authors would like to thank the Center for Vision, Speech, and Signal Processing at the University of Surrey, UK, for making the database of fish shapes publically available.

References