*Computer-Aided Design and Applications*
© 2008 CAD Solutions, LLC
http://www.cadanda.com

# Axial Deformation with Controllable Local Coordinate Frames

Chow Yuk Pui[1] and Hui K. C.[2]

[1]The Chinese University of Hong Kong, ypchow@mae.cuhk.edu.hk
[2]The Chinese University of Hong Kong, kchui@mae.cuhk.edu.hk

## ABSTRACT

An intuitive deformation tool is essential in design applications where freeform control of an existing object is required. In this paper, we present a technique for deforming 3D objects using an axial curve, which allows direct control on the local coordinate frames (LCF) defined on the axial curves. Users can move and twist the LCFs arbitrarily, while maintaining the continuity and the smoothness of the shape. This avoids undesirable twists as a result of the lack of intuitive control on the LCFs. The continuity and smoothness of the axial curve, and hence the associated geometric shape is maintained by adopting a suitable interpolation scheme to determine the LCF between user defined LCFs. Experiments show that the proposed method is capable of maintaining the shape's smoothness while eliminating undesirable twist and distortion.

**Keywords:** axial deformation, local coordinate frames, point inversion.
**DOI:** 10.3722/cadaps.2008.66-76

## 1. INTRODUCTION

Using axial curves to deform existing geometric shapes to create new products is a popular technique in geometric modeling. Although this technique is widely adopted in aesthetic design, however, the details (fairness and smoothness) of the geometric model may be undesirably distorted in the deformation. The Free-Form Deformation (FFD) [1] method introduced by Sederberg T. and Parry S. can be used to generate complex objects through manipulating a set of lattice control points. The Extended Free-form Deformation (EFFD) [2] technique provides a more intuitive user interface by allowing for the use of control lattices that are not parallelepiped-shaped. Bloomenthal and Lim [3] introduced an IK skeleton and a geometric skeleton technique for shape manipulation. Cao [4] proposed a different definition of axis for generalized cylinders for simple geometric shapes deformation. Wang et.al [5] introduced a feature-based technique for modifying freeform surfaces. Yoshizawa S. et.al [6] proposed to use control points to represent the surface of a geometric shape.

There has been considerable amount of works in the application of shape modeling techniques in animation, game developments [7], industrial and aesthetic design [8]. Their works mainly concentrate on the manipulation of special shape features on a surface, but not the deformation of an object.

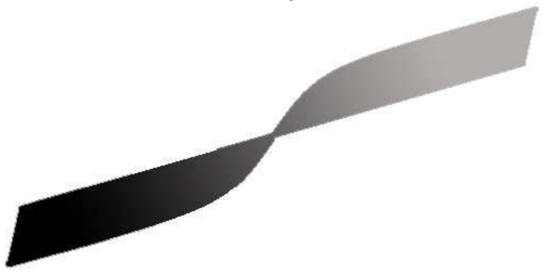
Fig. 1: An undesirable twist caused by the self-intersection of a curve-pair.

The axial deformation technique introduced by Lazarus et.al.[9] deforms an object by adjusting the shape of an axial curve. Although this method can be used for deforming complex objects, the lack of intuitive control on the axial curve may result in deformed shapes with undesirable distortions. Hui [10] proposed to use axial curve-pairs to perform freeform design. An axial curve-pair is composed of a primary curve and an orientation curve. The orientation curve is an approximate offset of the primary curve. The object shape can be modified by adjusting the curve-pair. In order to maintain the relationship between the two curves, a synchronized movement of the control points of the two curves is required. One limitation of this approach is that an undesirable twist of the object may be obtained when the primary curve and the orientation curve intersects each other. Figure 1 shows an example of this undesirable twist where distortions are found in the intersecting region.

In this paper, a different approach is employed. An axial curve and a set of user defined local coordinate frame constitute the basic elements of our framework. An object attached to the axial curve can be deformed by adjusting the curve. One important issue is the control of the twist of the curve. Instead of using an additional orientation curve [10], we make use of a set of local coordinate frame defined on the axial curve. The twist of the curve can be directly controlled by manipulating the local coordinate frames. By attaching an object to the axial curve, each vertex of the object is mapped to a local coordinate frame, and hence the object can be deformed according to the mapped local coordinate frame. To maintain the smoothness of the object, a special interpolation scheme is adopted to maintain a smooth change in the LCF between the user defined LCFs.

## 2. AXIAL DEFORMATION

Given a 3D object O, our goal is to provide a design tool to deform it in an intuitive way. While allowing the designer to deform, bend, twist or stretch the shape freely, the continuity and smoothness of the shape has to be maintained. Axial deformation proposed by Lazarus *et al.* [9] suggests that an object can be deformed by an axial curve **c**. Given a point **p** in the object O, **p** can be associated with the curve **c**, so that **p** is specified relative to a local coordinate frame defined on the curve **c**. By adjusting the curve **c** and the set of user defined local coordinate frames, the new position of **p** can be computed. The two major components of this axial deformation scheme are the axial curve and the set of local coordinate frames, and hence one important issue is the manipulation of the local coordinate frames. These concepts will be discussed in the following sections.

### 2.1 Axial Curve and Axial Space

In the proposed system, axial curves are specified by the user. An axial curve **c** is a curve that an object can be associated with. In this paper, B-spline is used for specifying the axial curve. A B-spline curve is a parametric curve defined with a set of control points:

$$\mathbf{c}(t) = \sum_{i=0}^{n} \mathbf{q}_i N_{i,k}(t)$$

where $\mathbf{q}_i$ are the control points, and $N_{i,k}(t)$ is the B-spline basis function of order *k* with $t_{start} \le t \le t_{end}$.

Given a parametric curve $\mathbf{c}(t)$ and a local coordinate system $\mathbf{l}(t) = \lfloor \mathbf{l}_x(t), \mathbf{l}_y(t), \mathbf{l}_z(t) \rfloor$, we can define the axial space as a 4-dimentional space which is a subset of $R^4$. A point $\mathbf{p}=(x, y, z)$, can be transformed to the axial space with a 4-tuple $\mathbf{p}=(t, u, v, w)$ where *t* specifies the coordinate frame the point **p** is belonging to, and $(u, v, w)$ are the local coordinates with respect to that coordinate frame.

A mapping function $f : R^4 \to R^3$ relates the axial space and the Euclidean space:

$$\mathbf{p} = f(t,u,v,w) = \mathbf{c}(t) + u\mathbf{l}_x(t) + v\mathbf{l}_y(t) + w\mathbf{l}_z(t)$$

Given the value of *t*, we can obtain a curve point $\mathbf{c}(t)$. Since an instance of a local coordinate system $\mathbf{l}(t)$ is defined on a curve point $\mathbf{c}(t)$, by subtracting $\mathbf{c}(t)$ from **p**, where $\mathbf{p}' = \mathbf{p} - \mathbf{c}(t)$, $\mathbf{p}'$ is then aligned with $\mathbf{l}(t)$, the values of *u*, *v* and *w* can then be computed by projecting $\mathbf{p}'$ to the three axis $\lfloor \mathbf{l}_x(t), \mathbf{l}_y(t), \mathbf{l}_z(t) \rfloor$.

$$u = \mathbf{p}' \cdot \mathbf{l}_x(t)$$
$$v = \mathbf{p}' \cdot \mathbf{l}_y(t)$$
$$w = \mathbf{p}' \cdot \mathbf{l}_z(t)$$

The problem remains is to obtain the value of *t*, which is usually defined as the value such that the curve point $\mathbf{c}(t)$ is closest to the object point **p**. This problem of inverse mapping an object point from the Euclidean space to the axial space will be discussed in the following section.

## 2.2 Local Coordinate Frame

One classic definition of the local coordinate frame on a curve is the Frenet Frame [11]. Given a curve $\mathbf{c}(t)$, the Frenet Frame is defined in terms of the first derivative $\mathbf{c}'(t)$ and the second derivative $\mathbf{c}''(t)$ of the curve:

$$\mathbf{T}(t) = \frac{\mathbf{c}'(t)}{\|\mathbf{c}'(t)\|}$$

$$\mathbf{B}(t) = \frac{\mathbf{c}'(t) \times \mathbf{c}''(t)}{\|\mathbf{c}'(t) \times \mathbf{c}''(t)\|}$$

$$\mathbf{N}(t) = \mathbf{B}(t) \times \mathbf{T}(t)$$

which are also referred to as the tangent **T**, normal **N**, and binormal **B** of $\mathbf{c}(t)$. This definition is straight forward and with the advantage that the frame can be computed analytically at any arbitrary point along the curve. But it suffers from two major limitations: Firstly, the frame is completely defined by the curve $\mathbf{c}(t)$, users can only adjust it via modifying the curve. No direct control on the orientation of the frame is allowed, and is not suitable for freeform design. Secondly, the second derivative may not be always available, $\mathbf{c}''(t)$ will vanish when the curve contains a straight segment, such that the binormal **B** is undefined, and hence the frame is undefined.

We proposed to use a method called Normal Plane Projection to create the normal vectors of the LCFs along the axial curve. Since the tangent at an arbitrary curve point is always available, a set of LCFs can be defined as:

$$\mathbf{l}_z(t) = \frac{\mathbf{c}'(t)}{|\mathbf{c}'(t)|}$$

$$\mathbf{l}_y(t) = \frac{\mathbf{N}(t)}{|\mathbf{N}(t)|} \qquad \text{where } \mathbf{N}(t) = \mathbf{Y} - (\mathbf{Y} \cdot \mathbf{l}_z(t))\mathbf{l}_z(t)$$

$$\mathbf{l}_x(t) = \mathbf{l}_y(t) \times \mathbf{l}_z(t)$$

However, the Normal Plane Projection method required an initial normal at the first knots point to get the normal at other knot points. Since the tangents are always available at the curve point, a tangent plane can be defined such that **t** is the normal to this tangent plane. And hence the normal vector of the LCF should be lying on this plane. One way to define the LCF is to project a vector onto this tangent plane. One possible choice is the world coordinate axis. Figure 2 shows a normal vector **n** which is obtained by projecting the y-axis $\mathbf{Y} = (0,1,0)$ onto the tangent plane.

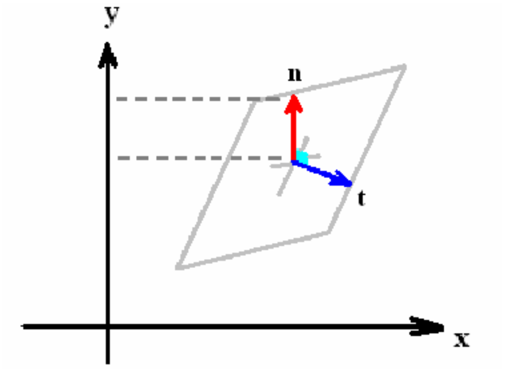

Fig. 2: Creating a normal vector by projecting the y-axis on a tangent plane.

With an initial normal $\mathbf{n_1}$ at the first knot point, $\mathbf{n_1}$ is projected to the tangent plane at the second knot point to give the normal $\mathbf{n_2}$ at the second knot point. The normal vector $\mathbf{n}_t$ at the other knot points are constructed in a similar manner, giving a set of normals and hence LCFs along the curve. Given a set of points on a curve, the normal $\mathbf{n}_t$ at a point is defined using the normal $\mathbf{n}_{t-1}$ at the previous location:

$$\mathbf{n}_t = \mathbf{n}_{t-1} - (\mathbf{n}_{t-1} \cdot \mathbf{t}_t)\mathbf{t}_t$$

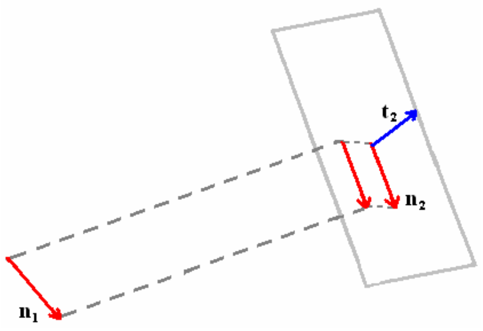

Fig. 3: Normal plane projection method.

One issue of this method is that, when the angle different between two consecutive normals is greater than 90 degrees, an undesirable flip may occur. This happens when the normal is projected in the opposite direction of the previous normal. One practical solution to this problem is to increase the number of user defined LCFs along the curve. This ensures a small angle different between two consecutive normals.

## 3. THE AXIAL DEFORMATION FRAMEWORK

There are three stages in our proposed axial deformation framework. They are the construction of the axial curve, the mapping between the object and the axial curve, and the interpolation of LCFs.

In the first stage, an axial curve is constructed. Usually the axial curve is constructed to approximate the skeleton of an imported object. Since in our method, the object deformation is performed using LCF, after constructing the axial curve, a set of LCFs is computed using the technique of normal plane projection [14] as discussed in the previous section. It requires a frame to be defined at the first knot position, in our system, it can be defined by using the Frenet Frame. Alternatively, when Frenet Frame is unavailable or as the user desire, the first frame can be assigned by the user by specifying the normal direction on the tangent plane at the first knot position. Figure 4a shows an example of the local coordinate frames produced along an axial curve using normal plane projection.

In the second stage, the system associates the imported object to the axial curve automatically. For each point on the object, the closest point on the axial curve and the local coordinates of the point with respect to the LCF at the closest curve point will be computed. This gives a 4-turple specifying a point in the axial space which is detailed in section 2.1. The inverse mapping from Euclidean space to the axial space will be discussed in section 3.1. Figure 4b shows an example of an object attached to an axial curve. Once the object points are mapped to the axial space, when the axial curve is adjusted or the LCFs on the curve are modified, the Euclidean coordinates of the object points can be updated by transforming the axial space coordinates back to the Euclidean space using the modified new local coordinate system.

In the final stage, we provide a method to maintain the continuity and the smoothness of the deformed object. After the object is mapped to the axial space defined by the axial curve and the local coordinate frames, the shape of the object is dictated by the shape of the curve and the coordinate frames. Maintaining the shape of the deformed object in other words is to maintain the shape of the axial curve and the coordinate frames. Since B-spline curve is used for the axial curve, which by definition is continuous and smooth, only the continuity of the frames have to be maintained with special technique.

Given a set of LCFs, the LCF between two given LCF is obtained by interpolating the pair of given LCFs. Various interpolation methods will be discussed in section 3.2. In this way, while enforcing the continuity and the smoothness, any unexpected twist can be avoided since the required LCF can always be obtained. Figure 4c shows a deformed object with a modified axial curve, and Figure 4d shows a deformed object with the LCFs twisted around the axial curve.

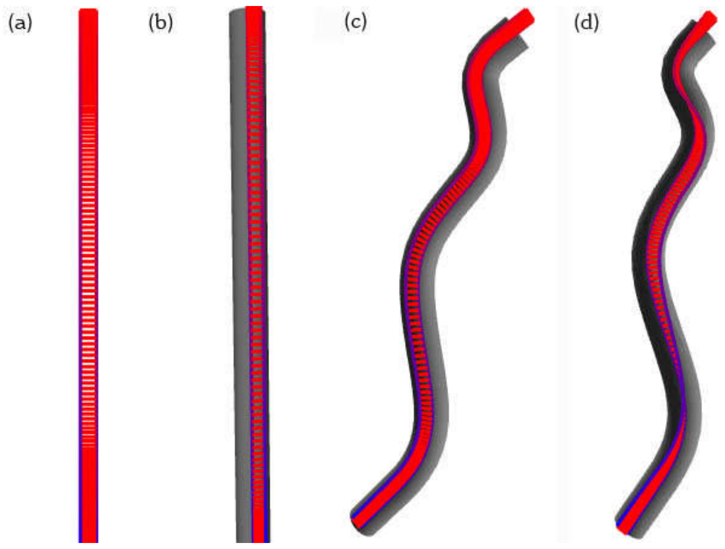

Fig. 4: A cylinder is deformed by an axial curve with LCFs.

### 3.1 Point Inversion

Consider an object O, and a point $\mathbf{p}_a$ in O, where $\mathbf{p}_a = (x, y, z)$, is in 3D Euclidean space. In order to associate $\mathbf{p}_a$ to an axial curve, $\mathbf{p}_a$ has to be transformed into the 4D axial space in the form of $\mathbf{p}_a = u, v$ and $w$. As mentioned in section 2.1, once $t$ is found, $u, v$ and $w$ can be determined easily. The parameter $t$ is the parametric value at which the curve point $\mathbf{c}(t)$ is closest to the point $\mathbf{p}_a$ .

In some cases, more than one object points may be mapped to the same curve point $t$. However, these object points must be at different distance from the curve point. Therefore, they are different points $(t, u, v, w)$ in the 4D axial space of the curve. Different object points thus map to different points in the axial space of the curve.

One straight forward method to find $t$ is to discretize the parametric range into a set of values $t_i$. The required value of $t$ is the $t_i$ that minimizes the distance from $\mathbf{p}$ to $\mathbf{c}(t)$. Using a direct linear search, the time complexity of the process is O(N), where N is the number of $t_i$. This may be expensive when the set of $t_i$ is large. To reduce the time complexity, we adopt the technique of nearest neighbor search [15], which reduces the search time to O(logN). Given the parameter set $t_i$, a set of discrete curve points is obtained. The nearest neighbor search can then be performed on the set of curve points. Based on our experiments, with the use of the nearest neighbor search, real time response is attained even with large point set in the order of thousands.

However, one limitation of the above discrete method is that the selected value of $t$ may not be optimal if the optimal value is not in the set of $t_i$, and hence only an approximation can be obtained. Although the approximation is usually enough for freeform design, one may desire to have an optimal solution for the value of $t$. In this case, we propose to use a modified version of the method proposed by Wang *et al* [16]. This method combines a quadratic minimization problem with the Newton's method. Given three initial values of $t$, quadratic minimization converges slowly but is good at refining coarse estimates; this is used to provide a rough estimation for the Newton's method which converges quickly given a good initial value.

In this paper, we propose to replace the quadratic minimization with the above discrete approximation. The first reason is that the quadratic minimization suffers from the problem of local minima, and may converge to a wrong estimate for the Newton's method. Instead of a local minimum, the discrete approximation always provides a solution that approximates the global optimal solution. The second reason is that no initial guess is required for the discrete approximation while it is essential for the quadratic minimization, a bad initial guess may cause the algorithm to fail.

Our proposed two-step algorithm to find *t*:

1. Discrete approximation: Given a discrete set of curve points, and an object point $\mathbf{p}_a$ , the closest point $\mathbf{c}^*$ to $\mathbf{p}_a$ is obtained using the nearest neighbor search. The approximate optimal *t** is the value of *t* associated with $\mathbf{c}^*$.
2. Newton's Method: Given a point $\mathbf{p}_a$ , and a curve point $\mathbf{c}(t)$, the square of the distance between them is:

   $$D(t) = \left(c_x(t) - x\right)^2 + \left(c_y(t) - y\right)^2 + \left(c_z(t) - z\right)^2$$

   The Newton's method attempts to find the optimal value of $t^*$ by minimizing D(*t*). It leads to the following iterative equation:

   $$t^{*,n+1} = t^{*,n} + \frac{D'(t^{*,n})}{D''(t^{*,n})}, \qquad \text{n=0,1,2...}$$

   The initial value $t^{*,0}$ is obtained from step 1, which is usually close to the optimal value $t^*$ . With a good initial guess, this iterative process converges quickly - usually within 3 iterations. Besides, the local minimum problem which is usually found in quadratic minimization is also avoided by the good initial value.

## 3.2 Local Coordinate Frame Interpolation

After attaching an object to an axial curve, each point of the object is associated with the closest point on the axial curve, and the local coordinate frame at the corresponding curve point will be regarded as the orientation frame reference. When the LCF $\left(\mathbf{l}_x(t), \mathbf{l}_y(t), \mathbf{l}_z(t)\right)$ is transformed to $\left(\mathbf{l}^*{}_x(t), \mathbf{l}^*{}_y(t), \mathbf{l}^*{}_z(t)\right)$ with a rotation transformation **R**, the mapped object point $\mathbf{p}_a$ will move to a new position $\mathbf{p}_a{}^*$ with the same transformation relative to the LCF.

$$\mathbf{l}^*{}_x(t) = \mathbf{l}_x(t)\mathbf{R}$$
$$\mathbf{l}^*{}_y(t) = \mathbf{l}_y(t)\mathbf{R}$$
$$\mathbf{l}^*{}_z(t) = \mathbf{l}_z(t)\mathbf{R}$$
$$\mathbf{p}_a{}^* = \mathbf{p}_a\mathbf{R}\left(\theta(t)\right)$$

To maintain the smoothness of the object, when one particular local coordinate frame is modified, an interpolation scheme is adopted to propagate the changes to the other LCF. Assume two LCFs are rotated for an angle $\theta_0$ and $\theta_1$ respectively, and $\sigma$ is the ratio of the arc length $l(t)$ to the total arc length $l(n)$ of the curve. Since each point on the imported object is automatically mapped to a particular point of the axial curve, the LCFs on these mapped curve points may not be positioned evenly. Generally, the linear interpolation is an effective method to calculate the rotation angle of the remaining LCFs between two fixed LCFs.

$$\sigma = \frac{l(t)}{l(n)},$$
$$\theta_t = (1-\sigma)\theta_0 + \sigma\theta_1$$

However, the rate of change of $\theta_t$ may not be continuous at the LCF. Consider the first derivative of the above equation, the rate of change of $\theta_t$ depends only on the angle difference (or twist angle), i.e. $\theta_t' = \left(\theta_1 - \theta_0\right)\sigma'$ . Hence, when two consecutive segments are having large difference in their twist angles, a sharp turn will be obtained as shown in Figure 5.

An algorithm is presented to determine the rotation angle $\theta_t$ of the LCF at any *t* between the modified LCFs. The cosine interpolation [17] is adopted in this paper which is defined as follows:

$$\sigma_1 = \frac{l(t)}{l(n)},$$
$$\sigma_2 = \frac{1-\cos\left(\sigma_1\pi\right)}{2}$$
$$\theta_t = \left(1-\sigma_2\right)\theta_0 + \sigma_2\theta_1$$

where $0 \le \sigma_1 \le 1$ and $0 \le \sigma_2 \le 1$.

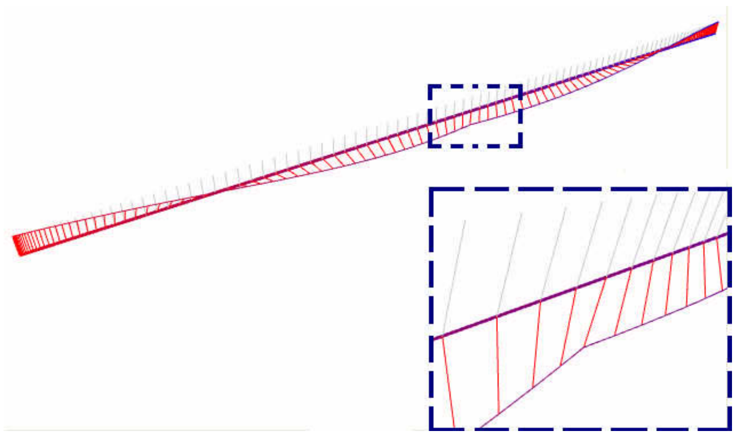
Fig. 5: The linear interpolation may cause a sharp corner when the middle LCF rotates 180 degrees.

One important advantage of using cosine interpolation is that the angle is guaranteed to change smoothly especially at the junction point of two segments. Consider the first derivative of $\theta_t$ :

$$\theta_t' = (\theta_1 - \theta_0)\sigma_2' = \frac{1}{2}(\theta_1 - \theta_0)\pi\sigma_1' \sin(\sigma_1\pi)$$

We can observe that at the two end points of each segment, where the values of $\sigma_1$ are 0 and 1, the rate of change is always zero. In other words, at the connection point of each segment, the torsion $\tau = \theta_t'$ at the junction point is always zero. The angle change is thus always smooth on the whole curve. Therefore, a suitable cosine function serves to provide a smooth transition between adjacent segments as shown in Figure 6.

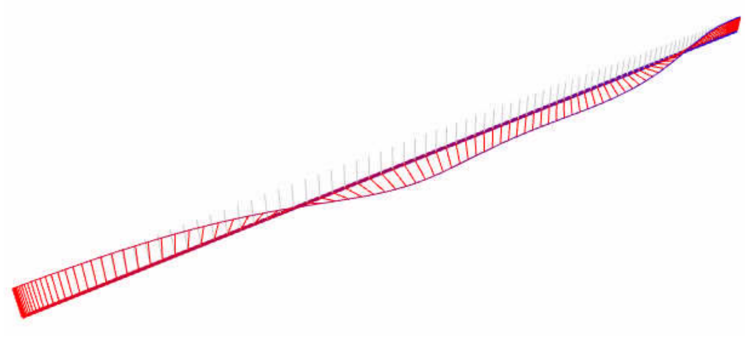
Fig. 6: The cosine interpolation.

## 4. RESULTS

An experimental system was implemented in a standard personal computer. The surface mesh of an object can be constructed using commercial CAD systems, such as Maya, 3D Studio Max etc. The mesh data file of the object is then exported from the CAD system and passed into our experimental system. The axial curve is constructed by specifying a number of data points. The system then constructs a set of local coordinate frames automatically by using Normal Plane Projection. Imported object can be attached to the axial curve which forms an axial representation of the object surfaces. Figure 7 shows a S- shape ribbon constructed with the experimental system. An axial curve is attached to the ribbon. The shape of the axial curve can be modified by adjusting the LCFs of the curve. The corresponding positions of the other frames are adjusted in accordance automatically.

A ribbon is modified to form a butterfly shape as shown in Figure 8. The deformation is obtained by orientating the local coordinate frames of the axial curve. Figure 9 shows a twisted ribbon. The local coordinate frame at the mid-point of the curve is rotated 180 degrees to obtain the twist, and the in-between frames are interpolated by Cosine interpolation. The result shows that no self intersection occurs on the ribbon.

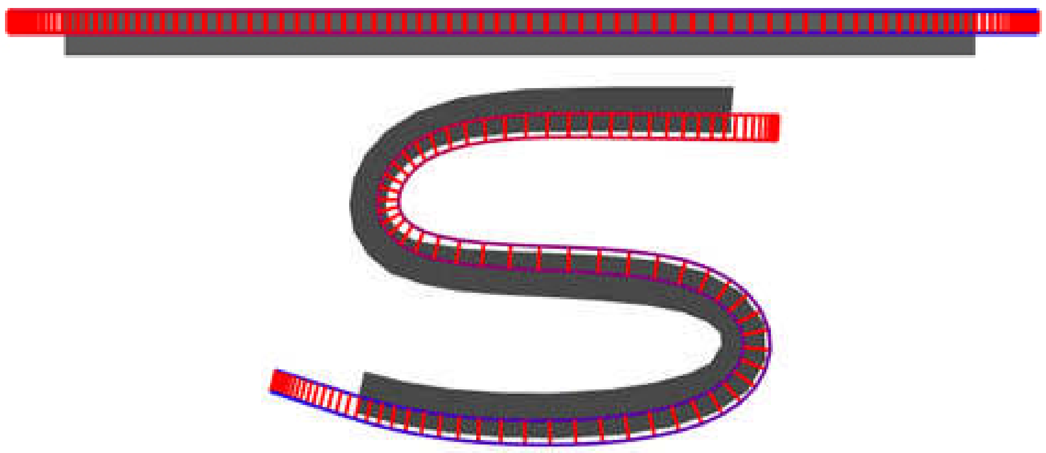

Fig. 7: A ribbon is bent to form a S-shape.

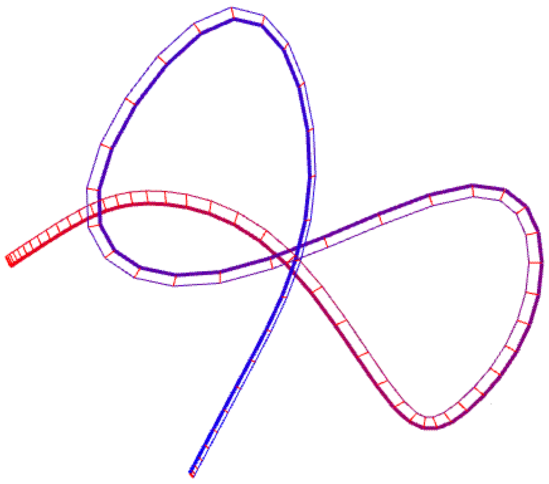

Fig. 8: Deforming a ribbon to a butterfly shape.

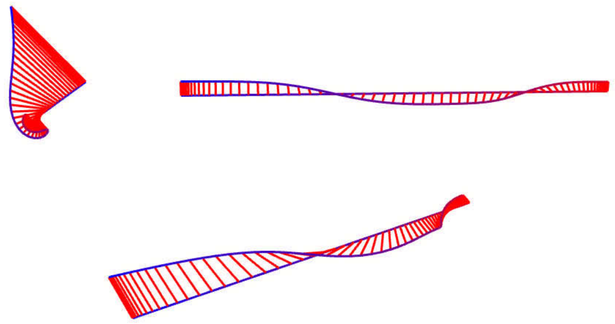

Fig. 9: A ribbon twisted 180 degrees, no intersection occurs.

Figure 10 shows the deformation of a set of leave. The initial structure of the leave is shown in Figure 10(a). The leave is attached to an axial curve (Figure 10(b)). The local coordinate frames are deformed to obtain a S-shape leave as shown in Figure 10(c). The axial curve is further twisted to obtain the shape as shown in Figure 10(d).

Figure 11 illustrates the design of the motions of a dolphin. The original posture of the dolphin is shown in Figure 11(a). An axial curve is defined through the body of dolphin as shown in Figure 11(b). In Figure 11(c), the axial curve is attached to the main body of the dolphin. The tail of the dolphin is then bent and moved upwards. Different postures can be obtained by orientating the axial curve. Figure 11(d) shows the tail of the dolphin being twisted 45 degrees about the axial curve.

Figure 12(a) shows the construction of an octopus structure. The head of the octopus is attached to the main axial curve. Individual arm is attached to its own axial curve as shown in Figure 12(b). The arms are deformed to obtain the shape in Figure 12(c).

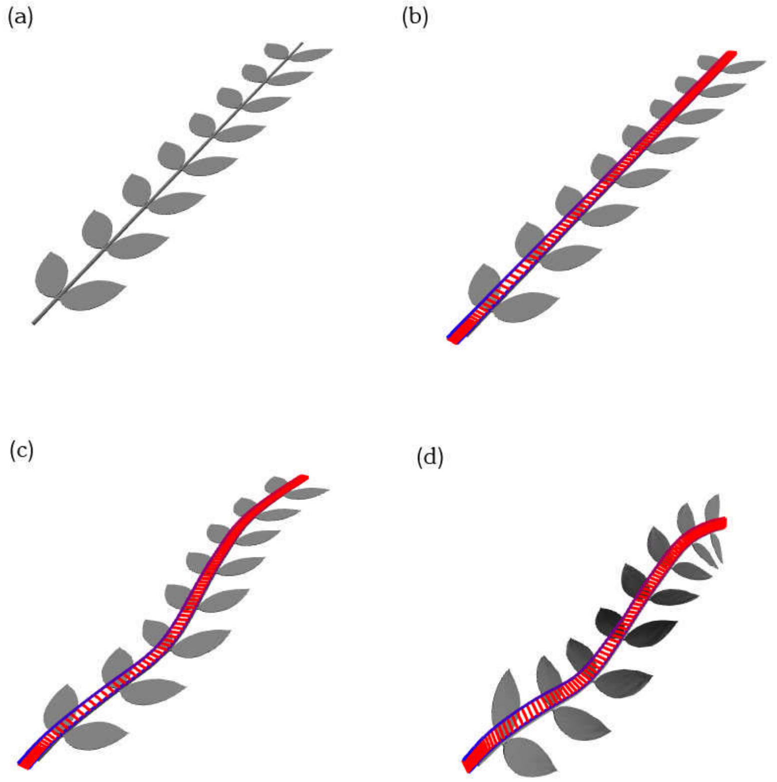

Fig. 10: Deformation of a leaf: (a) the original imported leaf; (b) the leaf stem defined with an axial curve; (c) result of deforming the leaf stems; (d) result of twisting the deformed stem to form a new shape.

## 5. CONCLUSION

Axial deformation with controllable local coordinate frames is a reliable technique for deforming freeform shape. An axial curve is constructed with a NURBS curve which is manipulated by the control points. However, deforming axial curve by control points may result in undesirable twist or distortion of an object. In the proposed algorithm for the axial deformation of an object, a set of local coordinate frames along an axial curve is obtained with the Normal Plane Projection method. When one particular local coordinate frame is modified, the modification is propagated to the whole set of local coordinate frames. An algorithm for interpolating the local coordinate frames is presented to maintain the smoothness and continuity of the coordinate frames along the axial curve. To associate object mesh points to points on the axial curve, each vertex on the object is mapped to the closest point on the axial curve. Users can directly control the LCFs to modify the shape of the axial curve. This modification is propagated to the attached mesh of the object. The new position of the mesh is updated automatically whenever the LCFs of the axial curve are modified. The smoothness and continuity of the axial curve can effectively reduce undesirable distortions of the shape. Experiments show that the technique can be used for blending and twisting regular or freeform objects and gives better results compared with the result of deforming object with curve pairs.

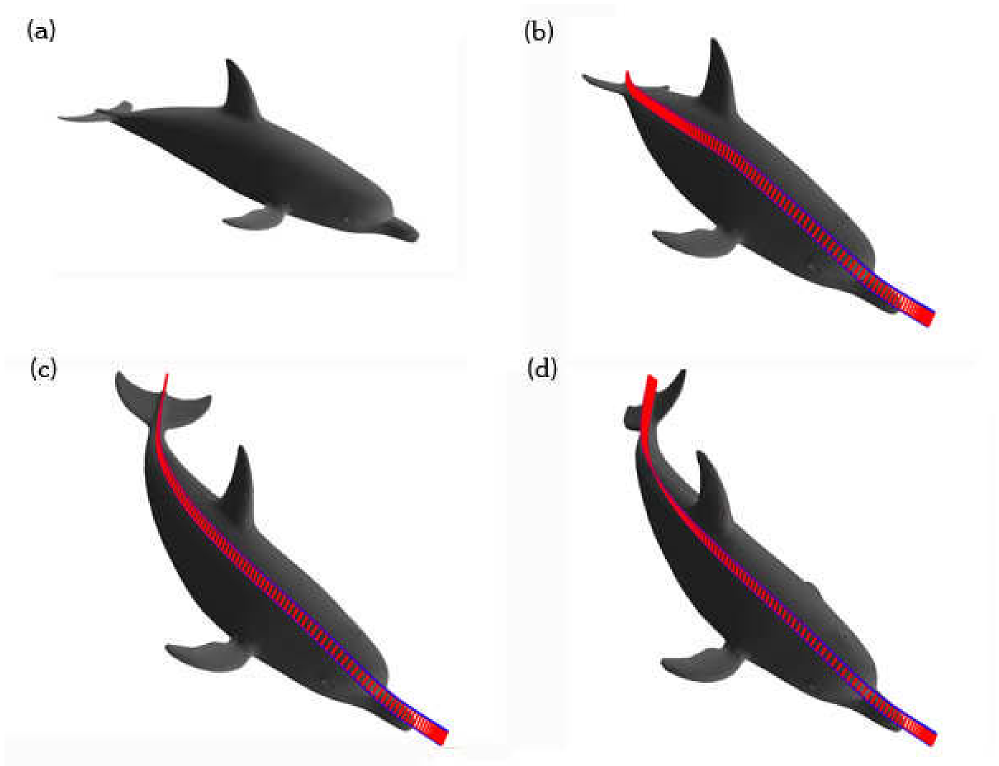

Fig. 11: Deformation of a dolphin: (a) the undeformed dolphin; (b) the dolphin modified by a single axial curve; (c) the tail of the dolphin is bent and moved upwards; (d) twisting the tail for 45 degrees.

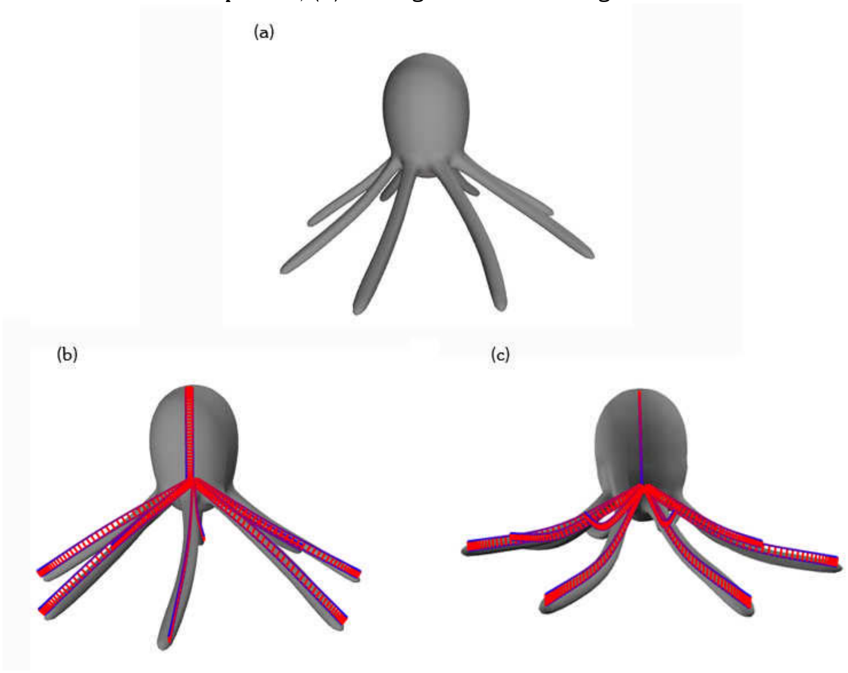

Fig. 12: Deformation of an octopus: (a) the undeformed octopus; (b) the octopus modified with multiple axial curves; (c) the octopus with bent arms.

## 6. ACKNOWLEDGEMENT
This work is supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region. (Project no. 412106.)

## 7. REFERENCES

[1] Sederberg, T.; Parry, S.: Free-form deformations of solid geometric models, Computer Graphics, 20, 1986, 151-160.

[2] Coquillart, S.: Extended free-form deformations: A sculpting tool for 3D geometric modeling, Computer Graphics, 24(4), 1990, 187-196.

[3] Bloomenthal, J.; Lim, C.: Skeletal Methods of Shape Manipulation, Proc. Shape Modeling International '99, 1999, 44-47.

[4] Cao, Y.: Axial Representations of 3D Shapes, Proc. IEEE Workshop on Statistical Signal Processing 2003, 311-314.

[5] Wang, C. C. L.; Wang, Y.; Yuen, M. M. F.: Design automation for customized apparel products, Computer-Aided Design, 37(7), 2005, 675-691.

[6] Yoshizawa, S.; Belyaev, A. G.; Seidel, H.-P.: A simple approach to interactive free-form shape deformations, Pacific Graphics 2002, October 9-11, 2002, 471-474.

[7] Yoshizawa, S.; Belyaev, A. G.; Seidel, H.-P.: Free-form skeleton-driven mesh deformations, ACM SIGGRAPH 2003, 247-253.

[8] Funkhouser, T.; Kazhdan, M.; Shilane, P.; Min, P.: Modeling by Example, ACM Transactions on Graphics, 23(3), 2004, 652-663.

[9] Lazarus, F. et al: Axial deformations: an intuitive deformation technique, Computer-Aided Design, 26(8), 1994, 607-613.

[10] Hui, K. C.: Free-form design using axial curve pairs, Computer-Aided Design, 34(8), 2002, 583-595.

[11] Faux, I. D.; Pratt, M. J.: Computation geometry for design and manufacturing, Wiley, 1979.

[12] Lossing, D. L.; Eshleman, A. L.: Planning a common data base for engineering and manufacturing, SHARE XLIII, Chicago, Aug, 1974.

[13] Woodward, C.: Skinning techniques for interactive B-spline interpolation, Computer-Aided Des., 20(8), 1988, 441-451.

[14] Bloomenthal, J.: Calculation of Reference Frames along a Space curve, Graphics Gems.

[15] Tao, Y.; Papadias, D. et al.: Continuous Nearest Neighbor Search, Proc. of 28th VLDB Conference, 2002.

[16] Wang, H.: An analytical solution for free-form roads in driving simulation, Technical Report 01-04, Department of Computer Science, the University of Iowa, 2001.

[17] Waters, K.; Levergood, T. M.: DECface: An Automatic Lip Synchronization Algorithm for Synthetic Faces, Tech. Report, CRL 93/4, DEC Cambridge Research Lab., 1993.