Applications of transformed curves

R. Tsow
B. Land
Cornell University
Ithaca, NY 14853
USA

May 23, 1994

Abstract

Ghosh and Jain [1] have developed a simple algebra of shapes with closed 2D curves. They use a Fourier series representation to represent the closed 2D curve and define binary composition operations to manipulate them. We explore this idea and propose some interesting ways of defining surfaces in 3D space.

1 Background

1.1 Parametric Representation of Closed Curves

For every curve in space, we can view it as a set of parametric equations of one variable. Consider a point p on a curve. The position of p at any instant t is given by $\vec{p} = (x(t), y(t), z(t))$ where $x(t), y(t)$ and $z(t)$ are functions of t.

Furthermore, for every closed curve in space, we can assume that p is traveling continuously along the curve. Therefore, $x(t), y(t), z(t)$ are periodic and

$$\exists T s.t. x(nT+t) = x(t), y(nT+t) = y(t), z(nT+t) = z(t), \forall n \in \aleph$$

.

1.2 Fourier Transform of Periodic Functions

From [1], for every periodic function $f(t)$ with period T, there is a Fourier series associates with it such that:

$$f(t) = a_0 + \sum_{n=1}^{\infty} (a_n \cos\frac{2\pi nt}{T} + b_n \sin\frac{2\pi nt}{T}) \tag{1}$$

with coefficients:

$$\begin{aligned} a_0 &= \frac{1}{T}\int_0^T f(t)\,dt, \\ a_n &= \frac{2}{T}\int_0^T f(t)\cos\frac{2\pi nt}{T}\,dt, \\ b_n &= \frac{2}{T}\int_0^T f(t)\sin\frac{2\pi nt}{T}\,dt. \end{aligned} \tag{2}$$

We can now write $\vec{p} = [x(t), y(t), z(t)]$ where:

$$\begin{aligned} x(t) &= a_0 + \sum_{n=1}^{\infty} (a_n \cos\frac{2\pi nt}{T} + b_n \sin\frac{2\pi nt}{T}), \\ y(t) &= c_0 + \sum_{n=1}^{\infty} (c_n \cos\frac{2\pi nt}{T} + d_n \sin\frac{2\pi nt}{T}), \\ z(t) &= e_0 + \sum_{n=1}^{\infty} (e_n \cos\frac{2\pi nt}{T} + f_n \sin\frac{2\pi nt}{T}). \end{aligned} \tag{3}$$

whose coefficients can be derived from Equation 2.

1.3 Fourier Transform of Colsed Polyline Curves

Assuming a polyline representation of the curve, following [1] but extending from 2 to 3 dimensions, we have:

$$\begin{aligned} a_0 &= \frac{1}{T}\sum_{j=1}^{K} \left[\begin{array}{c} \left(x_j - \frac{\Delta x_j}{\Delta t_j} t_{j-1}\right)(t_j - t_{j-1}) + \\ \frac{1}{2}\frac{\Delta x_j}{\Delta t_j}\left(t_j^2 - t_{j-1}^2\right) \end{array} \right] \\ a_n &= \frac{T}{2\pi^2 n^2}\sum_{j=1}^{K} \frac{\Delta x_j}{\Delta t_j}\left(\cos\frac{2\pi nt_j}{T} - \cos\frac{2\pi nt_{j-1}}{T}\right) \end{aligned}$$

$$b_n = \frac{T}{2\pi^2 n^2} \sum_{j=1}^{K} \frac{\Delta x_j}{\Delta t_j} \left(\sin \frac{2\pi n t_j}{T} - \sin \frac{2\pi n t_{j-1}}{T} \right) \tag{4}$$

where:

K	:	Total number of control points representing the curve
$\vec{p_i}$	:	The ith control point, $[x_i, y_i, z_i]$
Δl_i	:	Length of $\vec{p_{i+1}} - \vec{p_i}$
L	:	Total length of the polyline curve $= \sum_{i=1}^{K} \Delta l_i$
v	:	speed of $\vec{p}$
Δt_i	:	Time to traverse the ith line segment $= \frac{\Delta l_i}{v}$
t_i	:	Time to traverse the first i line segment $= \sum_{j=1}^{i} \Delta t_j$
T	:	period $= t_K = \frac{L}{v}$

Furthermore, if we substitute the time t by l/v into Equations 3 and 4, we have:

$$x(l) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos \frac{2\pi n l}{L} + b_n \sin \frac{2\pi n l}{L} \right) \tag{5}$$

and

$$\begin{aligned} a_0 &= \frac{1}{L} \sum_{j=1}^{K} \left[\begin{array}{c} \left(x_j - \frac{\Delta x_j}{\Delta l_j} l_{j-1} \right) (l_j - l_{j-1}) + \\ \frac{1}{2} \frac{\Delta x_j}{\Delta l_j} \left(l_j^2 - l_{j-1}^2 \right) \end{array} \right] \\ a_n &= \frac{L}{2\pi^2 n^2} \sum_{j=1}^{K} \frac{\Delta x_j}{\Delta l_j} \left(\cos \frac{2\pi n t_j}{L} - \cos \frac{2\pi n l_{j-1}}{L} \right) \\ b_n &= \frac{L}{2\pi^2 n^2} \sum_{j=1}^{K} \frac{\Delta x_j}{\Delta l_j} \left(\sin \frac{2\pi n t_j}{L} - \sin \frac{2\pi n l_{j-1}}{L} \right) \end{aligned} \tag{6}$$

therefore, the parametric equations are independent of time but only the curve length.

[1] noted 3 useful propoerties of this representation:

- **1:** Every closed curve has the same period T.
- **2:** If two periodic functions $f(t)$, $g(t)$ have the same period T, then the sum function

  $$h(t) = r_1 f(t) + r_2 g(t), r_1, r_2 \in \Re$$

  also has period T.
- **3:** All period functions of period T form a linear/vector space over the field of real numbers $\Re$.

Now consider two closed curves $C_1 = (x_1(t), y_1(t), z_1(t))$ and $C_2 = (x_2(t), y_2(t), z_2(t))$ as functions of period T. Then we can obtain the $x(t)$, $y(t)$, $z(t)$ from Equations 5 and 6. Therefore, they will all be periodic functions with period T.

1.4 Properties of Fourier Transformed Curves

As in [1], define an addition operator $\oplus$ between two curves as a component-wise sum:

$$\begin{aligned} C_1(t) \oplus C_2(t) &= (x_1(t), y_1(t), z_1(t)) \oplus (x_2(t), y_2(t), z_2(t)) \\ &= (x_1(t) + x_2(t), y_1(t) + y_2(t), z_1(t) + z_2(t)) \end{aligned} \tag{7}$$

and scalar product as

$$rC(t) = (rx(t), ry(t), rz(t)) \tag{8}$$

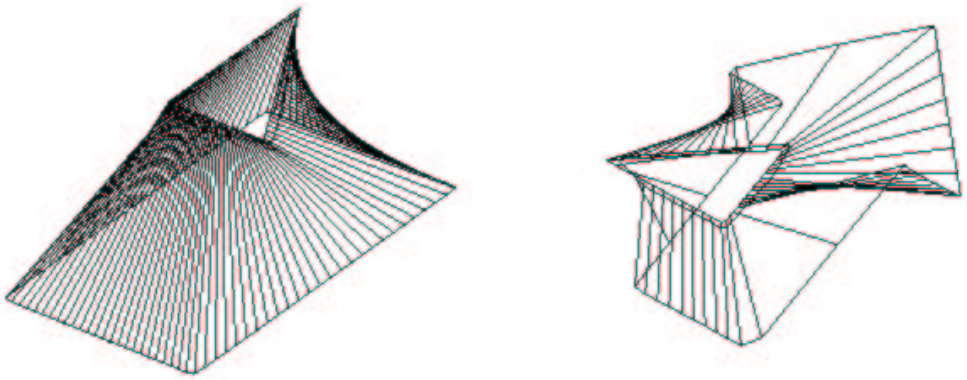
Thus, by the above lemmas and Equations 7 and 8, we have all closed curves form a linear/vector space over the field of real numbers $\Re$.

Notice that in our definition of Fourier Transformation of closed curves, the initial point and direction of traversal matters. Each curve is treated as a directed curve and has a starting point. Also observe that the additive inverse of a curve is the same curve with a different direction of traversal.

2 Applications

2.1 Connecting Arbitrary Curves

By transforming the curves, we can regenerate the curve by regenerating the control points. Moreover, we can regenerate any number of control points as desired, regardless of the number of control points we start with. This provides a nice way to connect arbitrary curves in 3-D, which is very important in 3-D reconstruction. For example, connecting hand digitized brain slices. Of course, one needs to have enough control points such that the shape of the curves is preserved.



connecting a triangle and a square connecting a triangle and an arrow

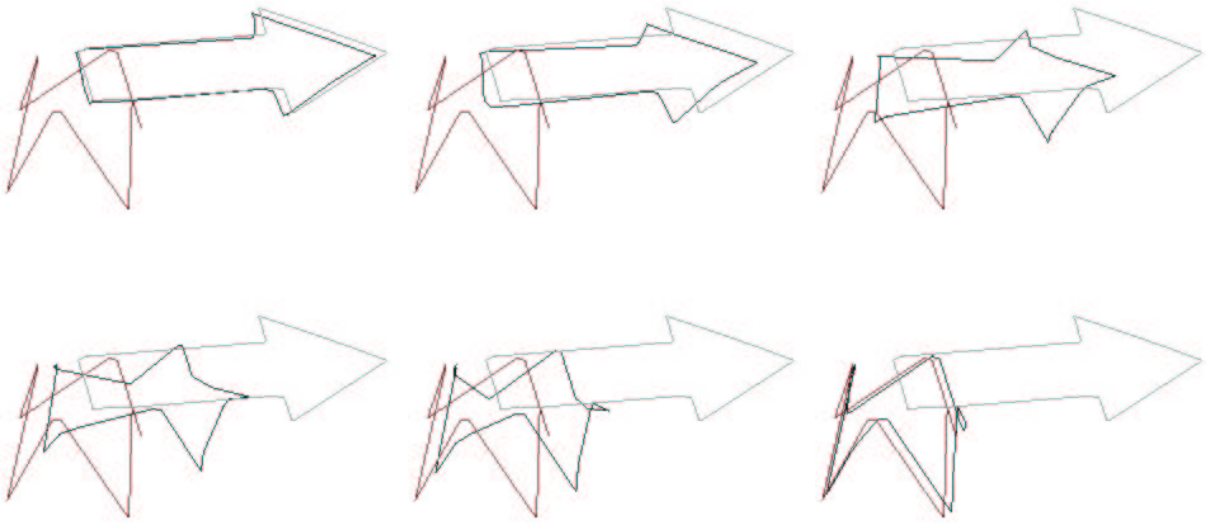
Figure 1: connecting curves

2.2 Morphing Curves

Suppose we have two closed curves, C_1 and C_2. Then we can change one curve to another smoothly by obtaining their corresponding Fourier series and calculating the weighted sum

$$rC_1 \oplus (1-r)C_2, 0 \leq r \leq 1 \tag{9}$$

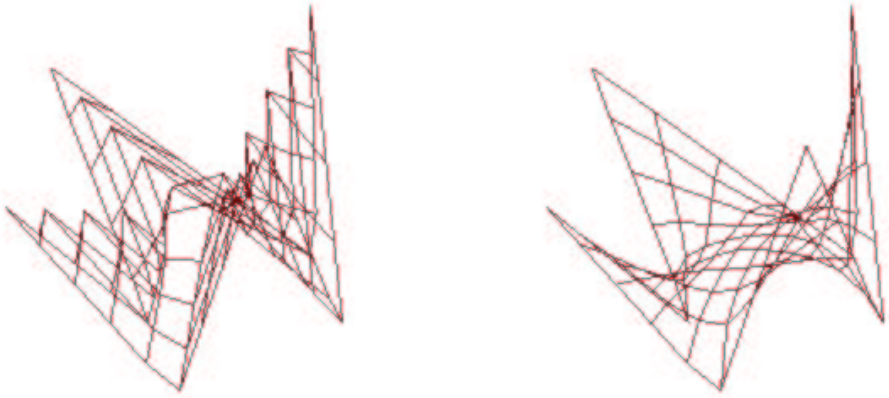
From the above properties, the resulting curve is also of period T and is still a closed curve in space.



morphing between an arrow and a crown shape
Figure 2: morphing curves

2.3 Defining Surfaces

Now, using Equation 9 and taking C_2 as the zero series, we can easily construct a surface given any closed curve in 3-D since this would be equivalent to scaling the curve at it's center. This provides a conveniant way to generate a grid given a 3-D closed curve. Therefore, defining a surface from a 3-D curve. Moreover, we can view the original frame as a rigid wire frame and every control points that are not on the original frame to be connections of springs. That is, we have many tiny little springs defining the surface. Further suppose that all the springs have natural length 0. This network of springs will tend to minimize the total energy in the system. Since the energy in a spring is proportional to square of the extended length or the spring, this reduced to minimization of the total square of length of the springs in the system, which in our case, is just the sum of the square of the length of all the connections. By solving this minimization problem, we can obtain the minimal energy surface of the frame. This is equivalent to the soap bubble surface on an arbitrary frame. The solution method is explained in a later section. Observe that the solution to the minimal surface is often approximated well by the scaled curves surface, so convergence is good.



defining a grid from a frame the minimal energy surface for the grid
Figure 3: defining surfaces

By viewing the system this way, we can define more surfaces from the wire frame by applying some force field to the system.

2.4 Morphing Surfaces

Once we defined surfaces from closed curves, because we can morph closed curves easily, we can also morph surfaces easily by morphing the closed curves that defined the surfaces and regenerating the morphed surface from the morphed curve.

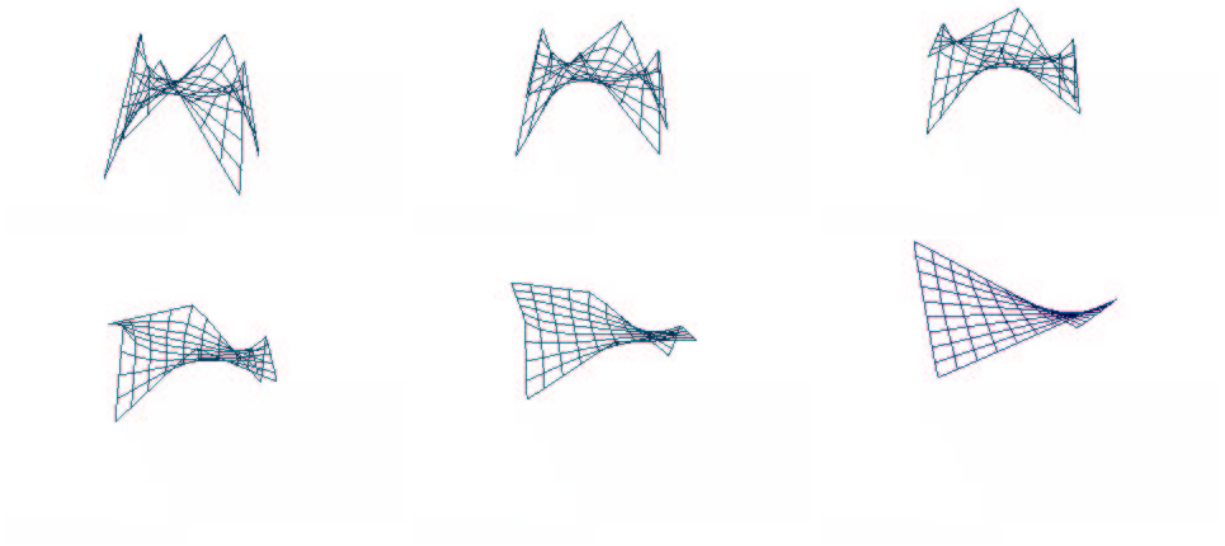


Figure 4: morphing between two surfaces

2.4.1 "Swap" Surface

Given a collection of closed curves, we can connect the curves easily by first transforming them into their corresponding Fourier series and regenerating the desired number of control points. Furthermore, we can generate some interesting surfaces if we morph the consecutive curves together. Using Equation 9, we can generate a number of intermediate curves between two curves. Each intermediate curve is a smooth transform from the previous one on to the next one. Connecting all the successive curves would yield a smooth "swap" surface, which is similar to a swept surface except that the cross sections can be different.

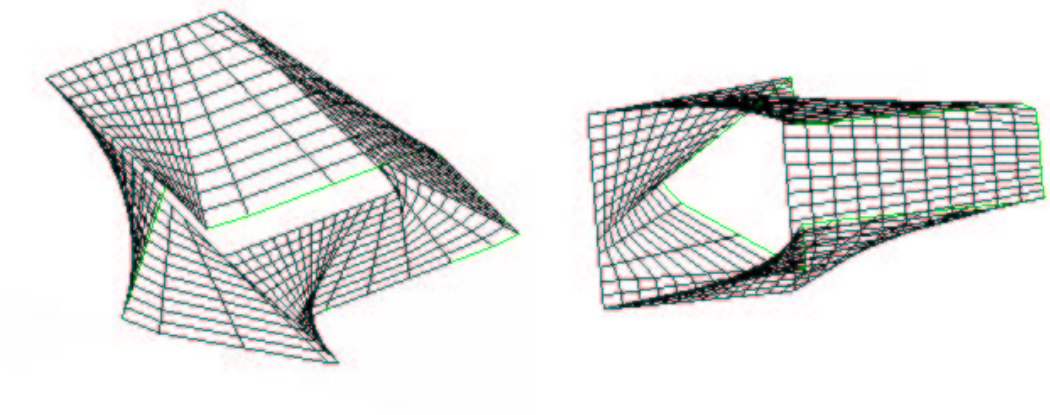


Figure 5: swap surface of an arrow and a square

In addition, one could also scale the intermediate curves, using

$$r(t)(tC_1 \oplus (1-t)C_2)$$

where $r(t)$ is the scaling factor and obtain some more interesting surfaces. One interesting thing to do is to make an object by a collection of closed curves. Then morph each curve to some other curves. This way, one can morph between objects easily.

Another interesting thing is that since the Fourier series of a curve depends on the initial point one picked, if one picked a different initial point, one would get a different series. We can use this fact and connect two curves with different initial points and obtain a twisted swap surface.

3 DX Modules

I have written DataExplorer Modules to do the above operations. We can take any closed polyline curves in space in DX format and transform it into a Fourier series. We then can manipulate the series mathematically and then inverse transform it back to a curve in space and display it on the screen. In the minimization part, I used conjugate gradient method [2]. I have tried some other methods such as the multidimensional Simplex method [2] but they are too slow. Since we know the derivatives of the function we are trying to minimize, conjugate gradient seems to be the method to use. See appendix for a description of the modules.

4 Conclusion

We have looked at the representation of closed space curves as Fourier series. Every closed curve in space can be represented by a function of period T, therefore, can be represented by a Fourier series. [1] has proven that the set of such Fourier series and the operation $\oplus$ as defined above formed a linear/vector space over the field of real numbers $\Re$. Therefore, we can manipulate the set of all closed space curves mathematically. We proposed several ways to make use of such properties. Morphing curves, defining interesting surfaces. I have written utilities to perform the transformation and operations described above using DataExplorer as a tool. Computer graphics techniques has proven to be a great help to scientists in visualization and understanding. The modules will help others understand the operations on the transformed curves and visualize the mathematics behind it and will help constructing novel surfaces for graphics applications.

5 Appendix - DX modules description

This set of modules transform closed polyline curves in space, manipulate them in Fourier space and inverse transform them back to 3 space. This provides a neat way to manipulate closed curves mathematically. [1] has defined mathematical operations for the set of transformed curves and has proven that they form a group. These modules provide a way to visualize and understand the mathematics behind the transformation. For a demo net, see morph.net.

poly - Fourier Transform of closed polylines

Take a group of polyline curves and transform them into their corresponding Fourier series representation. Output the groups containing their corresponding constant terms and the coefficients. How close the Fourier series approximate the original polyline curve depends on the number of harmonics generated. However, with fewer harmonics, one can obtain some interesting curves.

Input	1:	curve to be transformed, must have positions and connections.
Input	2:	number of harmonics desired.
Output	1:	Fourier transformed terms.
Output	2:	constant terms.

posit - Inverse Fourier Transform the Fourier Series

Take a group of Fourier Series and inverse transform them into their corresponding closed curves. Number of polyline segments determines how "jaggy" the resulting curve is.

Input	1:	the Fourier series.
Input	2:	the constant terms.
Input	3:	number of polyline segments desired.
Output	1:	the resulting curves.

polygrid - Generate a grid surface from the Fourier Series

Take a group of Fourier Series and generate grid surfaces corresponding to the Fourier Series. Notice this is not the minimal energy surface. The number of polyline segments per frame only affect the triangle grid but not the quad grid. Then number of frames desired directly affect the number of control points generated in a quad grid, thus directly affecting the minsurface module, and it is exponentially related to the computational time required in the minsurface module.

Input	1:	the Fourier series.
Input	2:	the constant terms.
Input	3:	number of polyline segments per frame.
Input	4:	number of frames desired.
Input	5:	kind of grid, quads or triangles.
Input	6:	frame displacement.
Output	1:	the resulting grid.
Output	2:	values for minsurface module (required).

minsurface - Calculate the minimal energy surface

Use conjugate gradient method to calculate the minimal energy surface given a grid surface generated by polygrid.

Input	1:	the approximate surface.
Input	2:	values from polygrid (required).
Input	3:	tolerance 1, for top loop of conjugate gradient.
Input	4:	tolerance 2, for sub loops.
Input	5:	number of points from the end to keep constant.
Input	6:	a constant field to apply to the system.
Input	7:	field strength.
Input	8:	type of field, plane or point.
Output	1:	the resulting surface.

polycone - Generate a morph and swap surface from a group of Fourier Series

Given a group of Fourier Series, morph and scale between the successive curves to obtain a morph and swap surface. Additional examples in coneEx.net.

Input	1:	the group of Fourier series.
Input	2:	the group of constant terms.
Input	3:	number of polyline segments per frame desired.
Input	4:	number of frames in between 2 curves.
Input	5:	scaling factor for each frame.
Output	1:	the resulting grid surface.

References

[1] Ghosh, Pijush K. and Jain, Pradeep K., "An Algebra of Geometric Shapes", **IEEE Computer Graphics & Applications**, Sept. 1993, pp. 50-59.

[2] Press, William H. at el., *Numerical Recipes in C, 2nd Edition*, Cambridge University Press, 1992.