Subdivision curves and surfaces

Brian Curless
CSE 557
Fall 2013

Reading

Required:

Note: there is an error in Stollnitz, et al., section A.5. Equation A.3 should read:

\[ MV = VΛ \]

This is already fixed in the handout.

Subdivision curves

Idea:
• repeatedly refine the control polygon
  \[ P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow \cdots \]
• curve is the limit of an infinite process
  \[ Q = \lim_{n \to \infty} P_n \]

Chaikin’s algorithm

Chakrin introduced the following “corner-cutting” scheme in 1974:
• Start with a piecewise linear curve
• Insert new vertices at the midpoints (the splitting step)
• Average each vertex with the “next” (clockwise) neighbor (the averaging step)
• Go to the splitting step

Old vertex
New vertex

1. Split
2. Average
3. Split
4. Average
**Averaging masks**

The limit curve is a quadratic B-spline!

Instead of averaging with the nearest neighbor, we can generalize by applying an **averaging mask** during the averaging step:

\[
\mathbf{r} = [\ldots \ r_1 \ r_0 \ r_1 \ ]
\]

In the case of Chaikin’s algorithm:

\[
\mathbf{r} =
\]

**Lane-Riesenfeld algorithm (1980)**

Use averaging masks from Pascal’s triangle:

\[
\mathbf{r} = \frac{1}{2^n} \binom{n}{0} \binom{n}{1} \cdots \binom{n}{n}
\]

Gives B-splines of degree \(n+1\).

\(n=0:\)

\(n=1:\)

\(n=2:\)

**Subdivide ad nauseum?**

After each split-average step, we are closer to the **limit curve**.

How many steps until we reach the final (limit) position?

Can we push a vertex to its limit position in one step?

**Local subdivision matrix**

Consider the cubic B-spline subdivision mask:

\[
\begin{bmatrix}
1 & 1 & 1 \\
\frac{1}{4} & \frac{1}{2} & \frac{1}{4}
\end{bmatrix}
\]

Now consider what happens during splitting and averaging in a small neighborhood:

We can write equations that relate points at one subdivision level to points at the previous:
Local subdivision matrix

We can write this as a recurrence relation in matrix form:

\[
\begin{align*}
L & = \begin{bmatrix} L_j \end{bmatrix} \\
C & = \begin{bmatrix} C_j \end{bmatrix} \\
R & = \begin{bmatrix} R_j \end{bmatrix}
\end{align*}
\]

where the \( L, R, C \)'s are (for convenience) row vectors.

In 2D, we can write out all the elements as follows:

\[
\begin{align*}
\begin{bmatrix} L^x \ L^y \\ C^x \ C^y \\ R^x \ R^y \end{bmatrix} &= \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/8 & 3/4 & 1/8 \\ 0 & 1/2 & 1/2 \end{bmatrix} \begin{bmatrix} L_{j-1}^x \\ L_{j-1}^y \\ C_{j-1}^x \\ C_{j-1}^y \\ R_{j-1}^x \\ R_{j-1}^y \end{bmatrix}
\end{align*}
\]

We can re-write this as:

\[
A_{\ell} = MA_{\ell-1}
\]

and \( M \) is the local subdivision matrix.

Eigenvectors and eigenvalues

We now need to look at the eigenvectors and eigenvalues of \( M \). Let \( \mathbf{v} \) be a vector such that:

\[
M\mathbf{v} = \lambda \mathbf{v}
\]

We say that \( \mathbf{v} \) is an eigenvector of \( M \) with eigenvalue \( \lambda \).

A 3x3 matrix can have 3 eigenvalues and eigenvectors:

\[
\begin{align*}
M\mathbf{v}_1 &= \lambda_1 \mathbf{v}_1 \\
M\mathbf{v}_2 &= \lambda_2 \mathbf{v}_2 \\
M\mathbf{v}_3 &= \lambda_3 \mathbf{v}_3
\end{align*}
\]

In matrix form:

\[
M \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix} = \begin{bmatrix} \lambda_1 & \lambda_2 & \lambda_3 \\ \lambda_2 & \lambda_3 & 0 \\ \lambda_3 & 0 & \lambda_1 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix}
\]

\[
MV = \mathbf{v} \Lambda
\]

To infinity, but not beyond...

Now let's apply \( M \) to original neighborhood \( \mathbf{A}_0 \):

\[
\mathbf{A}_1 = M\mathbf{A}_0 = A_0
\]

Now let's advance another subdivision:

\[
\mathbf{A}_2 = M^2\mathbf{A}_0 = MA_1 = A_1
\]

Do it \( j \) times:

\[
\mathbf{A}_j = M^j\mathbf{A}_0
\]

What if we do this an infinite number of times?

\[
\mathbf{A}_\infty = \lim_{j \to \infty} M^j\mathbf{A}_0
\]

Let's assume the eigenvalues are non-negative and sorted so that:

\[
\lambda_1 > \lambda_2 \geq \lambda_3 \geq \cdots \lambda_n \geq 0
\]

If \( \lambda_1 > 1 \), then:

If \( \lambda_1 < 1 \), then:

If \( \lambda_1 = 1 \), then:
Evaluation masks

For cubic B-splines, the local subdivision matrix $M$ is:

$$
M = \begin{bmatrix}
\frac{1}{2} & \frac{1}{2} & 0 \\
\frac{1}{8} & \frac{3}{4} & \frac{1}{8} \\
0 & \frac{1}{2} & \frac{1}{2}
\end{bmatrix}
$$

It's eigenvalues and eigenvectors are:

$$
\lambda_1 = 1 > \lambda_2 > \lambda_3, \text{ so we're OK!}
$$

$$
\mathbf{v}_1 = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{v}_3 = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}
$$

$$
\Lambda = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/4 \end{bmatrix}, \quad \mathbf{V} = \begin{bmatrix} 1 & -1 & 2 \\ 1 & 0 & -1 \\ 1 & 1 & 2 \end{bmatrix}
$$

We can now compute the limit position of the neighborhood $A_0$:

$$
A_\infty = \mathbf{V} \Lambda^{-1} \mathbf{V}^{-1} A_0
$$

Recipe for subdivision curves

The row vector $\mathbf{u}_j^t$ that pushes the original vertex to the limit position is called the evaluation mask:

$$
\mathbf{u}_j^t = \begin{bmatrix} \frac{1}{6} \\ \frac{2}{3} \\ \frac{1}{6} \end{bmatrix}
$$

Note that we do not need start with the 0th level control points and push them to the limit.

If we subdivide and average the control polygon $j$ times, we can push the vertices of the refined polygon to the limit as well:

$$
A_\infty = \mathbf{M}^n A_0 = \mathbf{V} \Lambda^{-1} \mathbf{V}^{-1} A_0
$$

We can now compute the limit position of the neighborhood $A_0$:

$$
\mathbf{A}_\infty = \mathbf{V} \Lambda^{-1} \mathbf{V}^{-1} A_0
$$

Tangent analysis

$$
\mathbf{A}_j = \begin{bmatrix} L_j^t \\ C_j^t \\ R_j^t \end{bmatrix}
$$

Note that we do not need to start with the 0th level control points and push them to the limit.

If we subdivide and average the control polygon $j$ times, we can push the vertices of the refined polygon to the limit as well:

$$
\mathbf{A}_\infty = \mathbf{M}^n A_0 = \mathbf{V} \Lambda^{-1} \mathbf{V}^{-1} A_0
$$

We can now compute the limit position of the neighborhood $A_0$:

$$
\mathbf{A}_\infty = \mathbf{V} \Lambda^{-1} \mathbf{V}^{-1} A_0
$$
Tangent analysis (cont’d)

\[ t_j = \left( \lambda_j w_j u_j^1 + \lambda_{j+1} w_j u_j^2 + \lambda_{j+2} w_j u_j^3 \right) A_0 \]

\[ \lambda_1 > \lambda_2 > \lambda_3 \geq \cdots \geq \lambda_n \geq 0 \]

DLG interpolating scheme (1987)

Slight modification to subdivision algorithm:
- splitting step introduces midpoints
- averaging step only changes midpoints

For DLG (Dyn-Levin-Gregory), use:

\[ r_{\text{old}} = (1) \quad r_{\text{new}} = \frac{1}{16} (-2, 5, 10, 5, -2) \]

Since we are only changing the midpoints, the points after the averaging step do not move.

Building complex models

We can extend the idea of subdivision from curves to surfaces...

Subdivision surfaces

Chaikin’s use of subdivision for curves inspired similar techniques for subdivision surfaces.

Iteratively refine a control polyhedron (or control mesh) to produce the limit surface

\[ S = \lim_{j \to \infty} P_j \]

using splitting and averaging steps.
**Triangular subdivision**

There are a variety of ways to subdivide a polygon mesh.

A common choice for triangle meshes is 4:1 subdivision – each triangular face is split into four smaller triangles:

![Original and After splitting](image)

**Loop averaging step**

Once again we can use **masks** for the averaging step:

![Vertex neighborhood and Averaging mask](image)

\[ q = \frac{\alpha(n)Q + Q_1 + \cdots + Q_n}{\alpha(n) + n} \]

where

\[ \alpha(n) = \frac{n(1 - \beta(n))}{\beta(n)} \quad \beta(n) = \frac{5}{4} \left( \frac{3 + 2\cos(2\pi / n)}{2} \right)^2 \]

These values, due to Charles Loop, are carefully chosen to ensure smoothness – namely, tangent plane or normal continuity.

Note: tangent plane continuity is also known as \( G^1 \) continuity for surfaces.

**Loop evaluation and tangent masks**

As with subdivision curves, we can split and average a number of times and then push the points to their limit positions.

\[ \epsilon(n) = \frac{3n}{\beta(n)} \]

\[ \tau(n) = \cos(2\pi/n) \]

How do we compute the normal?

**Recipe for subdivision surfaces**

As with subdivision curves, we can now describe a recipe for creating and rendering subdivision surfaces:

- Subdivide (split-average) the control polyhedron a few times. Use the averaging mask.
- Compute two tangent vectors using the tangent masks.
- Compute the normal from the tangent vectors.
- Push the resulting points to the limit positions. Use the evaluation mask.
- Render!
Adding creases without trim curves

For NURBS surfaces, adding sharp features like creases required the use of trim curves.

For subdivision surfaces, we can just modify the subdivision masks. E.g., we can mark some edges and vertices as “creases” and modify the subdivision mask for them (and their children):

This gives rise to $G^0$ continuous surfaces (i.e., having positional but not tangent plane continuity).

Creases without trim curves, cont.

Here’s an example using Catmull-Clark surfaces (based on subdividing quadrilateral meshes):

This particular example uses the hybrid technique of DeRose, et al., which applies sharp subdivision rules at some creases for a finite number of steps, and then switches to smooth subdivision, giving more gentle creases. This technique was used in Geri’s Game.

Catmull-Clark subdivision

4:1 subdivision of triangles is sometimes called a face scheme for subdivision, as each face begets more faces.

An alternative face scheme starts with arbitrary polygon meshes and inserts vertices along edges and at face centroids:

Note: after the first subdivision, all polygons are quadrilaterals in this scheme.

Interpolating subdivision surfaces

Interpolating schemes are defined by

- splitting
- averaging only new vertices

The following averaging mask is used in butterfly subdivision:

Setting $t=0$ gives the original polyhedron, and increasing small values of $t$ makes the surface smoother, until $t=1/8$ when the surface is provably $G^1$. 