Subdivision curves

Reading

Recommended:

- Stollnitz, DeRose, and Salesin. *Wavelets for Computer Graphics: Theory and Applications,* 1996, section 6.1-6.3, A.5.

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

$$\mathbf{MV} = \mathbf{V}\Lambda$$

Subdivision curves

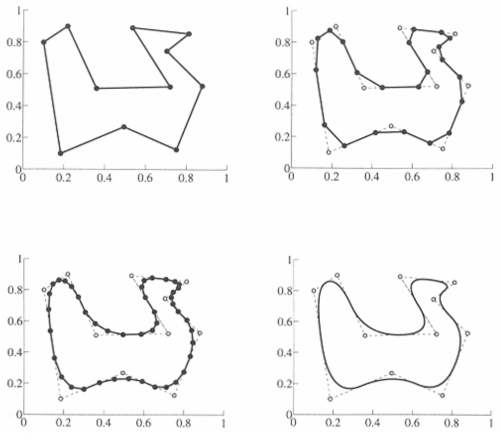
Idea:

- repeatedly refine the control polygon

$$P^1 \rightarrow P^2 \rightarrow P^3 \rightarrow \cdots$$

- curve is the limit of an infinite process

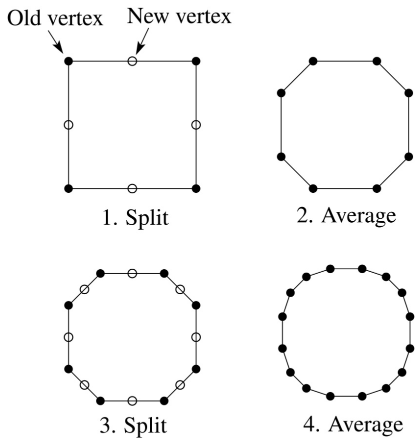
$$Q = \lim_{j \to \infty} P^j$$



Chaikin's algorithm

Chakin 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



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:

$$r = (\ldots, r_{-1}, r_0, r_1, \ldots)$$

In the case of Chaikin's algorithm:

$$r =$$

Lane-Riesenfeld algorithm (1980)

Use averaging masks from Pascal's triangle:

$$r = \frac{1}{2^n}\left(\binom{n}{0}, \binom{n}{1}, \ldots, \binom{n}{n}\right)$$

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?

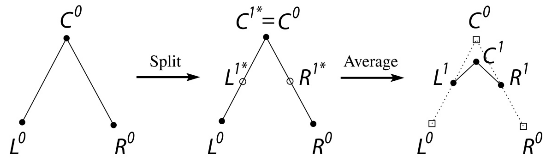
Can we push a vertex to its limit position without infinite subdivision? Yes!

Local subdivision matrix

Consider the cubic B-spline subdivision mask:

$$\frac{1}{4}\begin{pmatrix}1 & 2 & 1\end{pmatrix}$$

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:

$$\begin{pmatrix} L^j \\ C^j \\ R^j \end{pmatrix} = \frac{1}{8}\begin{pmatrix} 4 & 4 & 0 \\ 1 & 6 & 1 \\ 0 & 4 & 4 \end{pmatrix}\begin{pmatrix} L^{j-1} \\ C^{j-1} \\ R^{j-1} \end{pmatrix}$$

$$\mathbf{Q}^j = \mathbf{S}\mathbf{Q}^{j-1}$$

Where the *L, R, C*'s are (for convenience) *row* vectors and **S** is the **local subdivision matrix**.

We can think about the behavior of each coordinate independently. For example, the x-coordinate:

$$\begin{pmatrix} x_L^j \\ x_C^j \\ x_R^j \end{pmatrix} = \frac{1}{8}\begin{pmatrix} 4 & 4 & 0 \\ 1 & 6 & 1 \\ 0 & 4 & 4 \end{pmatrix}\begin{pmatrix} x_L^{j-1} \\ x_C^{j-1} \\ x_R^{j-1} \end{pmatrix}$$

$$X^j = \mathbf{S}X^{j-1}$$

Local subdivision matrix, cont'd

Tracking just the *x* components through subdivision:

$$X^j = \mathbf{S}X^{j-1} = \mathbf{S}\cdot\mathbf{S}X^{j-2} = \mathbf{S}\cdot\mathbf{S}\cdot\mathbf{S}X^{j-3} = \cdots = \mathbf{S}^j X^0$$

The limit position of the x's is then:

$$X^\infty = \lim_{j\to\infty} \mathbf{S}^j X^0$$

OK, so how do we apply a matrix an infinite number of times??

Eigenvectors and eigenvalues

To solve this problem, we need to look at the eigenvectors and eigenvalues of **S**. First, a review…

Let *v* be a vector such that:

$$\mathbf{S}\mathbf{v} = \lambda\mathbf{v}$$

We say that **v** is an eigenvector with eigenvalue λ.

An *nxn* matrix can have *n* eigenvalues and eigenvectors:

$$\begin{gathered} \mathbf{S}\mathbf{v}_1 = \lambda_1\mathbf{v}_1 \\ \vdots \\ \mathbf{S}\mathbf{v}_n = \lambda_n\mathbf{v}_n \end{gathered}$$

If the eigenvectors are linearly independent (which means that *S* is *non-defective*), then they form a basis, and we can re-write *X* in terms of the eigenvectors:

$$X = \sum_i^n a_i\mathbf{v}_i$$

To infinity, but not beyond…

Now let's apply the matrix to the vector X:

$$X^1 = \mathbf{S}X^0 = \mathbf{S}\sum_i^n a_i\mathbf{v}_i = \sum_i^n a_i\mathbf{S}\mathbf{v}_i = \sum_i^n a_i\lambda_i\mathbf{v}_i$$

Applying it *j* times:

$$X^j = \mathbf{S}^j X = \mathbf{S}^j\sum_i^n a_i\mathbf{v}_i = \sum_i^n a_i\mathbf{S}^j\mathbf{v}_i = \sum_i^n a_i\lambda_i^j\mathbf{v}_i$$

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

$$\lambda_1 > \lambda_2 > \lambda_3 \geq \cdots \geq \lambda_n \geq 0$$

Now let *j* go to infinity:

$$X^\infty = \lim_{j\to\infty}\mathbf{S}^j X^0 = \lim_{j\to\infty}\sum_i^n a_i\lambda_i^j\mathbf{v}_i$$

If $\lambda_1 > 1$, then:

If $\lambda_1 < 1$, then:

If $\lambda_1 = 1$, then:

Evaluation masks

What are the eigenvalues and eigenvectors of our cubic B-spline subdivision matrix?

$$\lambda_1 = 1 \qquad \lambda_2 = \frac{1}{2} \qquad \lambda_3 = \frac{1}{4}$$

$$\mathbf{v}_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \qquad \mathbf{v}_2 = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \qquad \mathbf{v}_3 = \begin{pmatrix} 2 \\ -1 \\ 2 \end{pmatrix}$$

We're OK!

But where did the x-coordinates end up?

What about the y-coordinates?

Evaluation masks, cont'd

To finish up, we need to compute a_1. First, we can reorganize the expansion of X into the eigenbasis:

$$X^0 = a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_n\mathbf{v}_n = \begin{bmatrix} \vdots & \vdots & & \vdots \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \\ \vdots & \vdots & & \vdots \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \mathbf{V}A$$

We can then solve for the coefficients in this new basis:

$$A = \mathbf{V}^{-1}X^0$$

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \cdots & \mathbf{u}_1^T & \cdots \\ \cdots & \mathbf{u}_2^T & \cdots \\ & \vdots & \\ \cdots & \mathbf{u}_n^T & \cdots \end{bmatrix} X^0$$

Now we can compute the limit position of the x-coordinate:

$$x_C^\infty = a_1 = \mathbf{u}_1^T X^0$$

We call $\mathbf{u}_1$ the **evaluation mask**.

Evaluation masks, cont'd

Note that we need not 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:

$$x^\infty = \mathbf{S}^\infty X^j = \mathbf{u}_1^T X^j$$

The same result obtains for the y-coordinate:

$$y^\infty = \mathbf{S}^\infty Y^j = \mathbf{u}_1^T Y^j$$

Left eigenvectors

What are these *u*-vectors? Consider the eigenvector relation:

$$\mathbf{S}\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

We can re-write this as a matrix:

$$\mathbf{S}\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix} = \begin{bmatrix} \lambda_1\mathbf{v}_1 & \lambda_2\mathbf{v}_2 & \lambda_3\mathbf{v}_3 \end{bmatrix}$$

$$\mathbf{S}\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$$

$$S\mathbf{V} = \mathbf{V}\Lambda$$

where **V** is the concatenation of the eigenvectors into a matrix and Λ is a diagonal matrix filled with the eigenvalues of **S**.

Left eigenvectors (cont'd)

Now lets multiply both sides by $\mathbf{V}^{-1}$ from the left and right and then simplify:

$$\mathbf{V}^{-1}(S\mathbf{V})\mathbf{V}^{-1} = \mathbf{V}^{-1}(\mathbf{V}\Lambda)\mathbf{V}^{-1}$$
$$\mathbf{V}^{-1}S = \Lambda\mathbf{V}^{-1}$$
$$\mathbf{U}S = \Lambda\mathbf{U}$$

If we "de-construct" this relation, we get:

$$\mathbf{U}S = \Lambda\mathbf{U}$$
$$\begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \mathbf{u}_3^T \end{bmatrix} S = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \mathbf{u}_3^T \end{bmatrix}$$
$$\begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \mathbf{u}_3^T \end{bmatrix} S = \begin{bmatrix} \lambda_1\mathbf{u}_1^T \\ \lambda_2\mathbf{u}_2^T \\ \lambda_3\mathbf{u}_3^T \end{bmatrix}$$

Thus, we find that the *u*-vectors obey the relation:

$$\mathbf{u}_i^T\mathbf{S} = \lambda_i\mathbf{u}_i^T$$

These are the "**left eigenvectors**" of **S**. (Alternatively, they are the eigenvectors of $\mathbf{S}^T$.)

Recipe for subdivision curves

The evaluation mask for the cubic B-spline is:

$$\frac{1}{6}\begin{pmatrix} 1 & 4 & 1 \end{pmatrix}$$

Now we can cook up a simple procedure for creating subdivision curves:

- Subdivide (split+average) the control polygon a few times. Use the averaging mask.
- Push the resulting points to the limit positions. Use the evaluation mask.

Tangent analysis

What is the tangent to the cubic B-spline curve?

First, let's consider how we represent the x and y coordinate neighborhoods:

$$X^0 = a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + a_3\mathbf{v}_3$$
$$Y^0 = b_1\mathbf{v}_1 + b_2\mathbf{v}_2 + b_3\mathbf{v}_3$$

We can view the point neighborhoods then as:

$$\mathbf{Q}^0 = \begin{bmatrix} X^0 & Y^0 \end{bmatrix} = \mathbf{v}_1\begin{bmatrix} a_1 & b_1 \end{bmatrix} + \mathbf{v}_2\begin{bmatrix} a_2 & b_2 \end{bmatrix} + \mathbf{v}_3\begin{bmatrix} a_3 & b_3 \end{bmatrix}$$

After *j* subdivisions, we would get:

$$\begin{aligned} \mathbf{Q}^j &= S^j\left\{\mathbf{v}_1\begin{bmatrix} a_1 & b_1 \end{bmatrix} + \mathbf{v}_2\begin{bmatrix} a_2 & b_2 \end{bmatrix} + \mathbf{v}_3\begin{bmatrix} a_3 & b_3 \end{bmatrix}\right\} \\ &= \lambda_1^j\mathbf{v}_1\begin{bmatrix} a_1 & b_1 \end{bmatrix} + \lambda_2^j\mathbf{v}_2\begin{bmatrix} a_2 & b_2 \end{bmatrix} + \lambda_3^j\mathbf{v}_3\begin{bmatrix} a_3 & b_3 \end{bmatrix} \end{aligned}$$

We can write this more explicitly as:

$$\begin{bmatrix} L \\ C \\ R \end{bmatrix} = \lambda_1^j\begin{bmatrix} v_{1,L} \\ v_{1,C} \\ v_{1,R} \end{bmatrix}\begin{bmatrix} a_1 & b_1 \end{bmatrix} + \lambda_2^j\begin{bmatrix} v_{2,L} \\ v_{2,C} \\ v_{2,R} \end{bmatrix}\begin{bmatrix} a_2 & b_2 \end{bmatrix} + \lambda_3^j\begin{bmatrix} v_{3,L} \\ v_{3,C} \\ v_{3,R} \end{bmatrix}\begin{bmatrix} a_3 & b_3 \end{bmatrix}$$

Tangent analysis (cont'd)

The tangent to the curve is along the direction:

$$\mathbf{t} = \lim_{j\to\infty}\left(R^j - C^j\right)$$

What's wrong with this definition?

Instead, we'll find the *normalized* tangent direction :

$$\mathbf{t} = \lim_{j\to\infty}\frac{R^j - C^j}{\left\|R^j - C^j\right\|}$$

Now, let's look at the "right" and "center" points in isolation:

$$R^j = \lambda_1^j v_{1,R}\begin{bmatrix} a_1 & b_1 \end{bmatrix} + \lambda_2^j v_{2,R}\begin{bmatrix} a_2 & b_2 \end{bmatrix} + \lambda_3^j v_{3,R}\begin{bmatrix} a_3 & b_3 \end{bmatrix}$$
$$C^j = \lambda_1^j v_{1,C}\begin{bmatrix} a_1 & b_1 \end{bmatrix} + \lambda_2^j v_{2,C}\begin{bmatrix} a_2 & b_2 \end{bmatrix} + \lambda_3^j v_{3,C}\begin{bmatrix} a_3 & b_3 \end{bmatrix}$$

The difference between these is:

$$\begin{aligned} R^j - C^j = {} & \lambda_1^j\left(v_{1,R} - v_{1,C}\right)\begin{bmatrix} a_1 & b_1 \end{bmatrix} + \\ & \lambda_2^j\left(v_{2,R} - v_{2,C}\right)\begin{bmatrix} a_2 & b_2 \end{bmatrix} + \lambda_3^j\left(v_{3,R} - v_{3,C}\right)\begin{bmatrix} a_3 & b_3 \end{bmatrix} \\ = {} & \lambda_2^j\left(v_{2,R} - v_{2,C}\right)\begin{bmatrix} a_2 & b_2 \end{bmatrix} + \lambda_3^j\left(v_{3,R} - v_{3,C}\right)\begin{bmatrix} a_3 & b_3 \end{bmatrix} \end{aligned}$$

The tangent mask

And now computing the tangent:

$$
\begin{aligned}
\lim_{j\to\infty}\frac{R^j - C^j}{\left\|R^j - C^j\right\|} &= \lim_{j\to\infty}\frac{\lambda_2^j\left(v_{2,R}-v_{2,C}\right)\left[a_2\quad b_2\right]+\lambda_3^j\left(v_{3,R}-v_{3,C}\right)\left[a_3\quad b_3\right]}{\left\|\lambda_2^j\left(v_{2,R}-v_{2,C}\right)\left[a_2\quad b_2\right]+\lambda_3^j\left(v_{3,R}-v_{3,C}\right)\left[a_3\quad b_3\right]\right\|}\\
&= \lim_{j\to\infty}\frac{\left(v_{2,R}-v_{2,C}\right)\left[a_2\quad b_2\right]+\left(\frac{\lambda_3}{\lambda_2}\right)^j\left(v_{3,R}-v_{3,C}\right)\left[a_3\quad b_3\right]}{\left\|\left(v_{2,R}-v_{2,C}\right)\left[a_2\quad b_2\right]+\left(\frac{\lambda_3}{\lambda_2}\right)^j\left(v_{3,R}-v_{3,C}\right)\left[a_3\quad b_3\right]\right\|}\\
&= \frac{\left(v_{2,R}-v_{2,C}\right)\left[a_2\quad b_2\right]}{\left\|\left(v_{2,R}-v_{2,C}\right)\left[a_2\quad b_2\right]\right\|}\\
&= \frac{\left[a_2\quad b_2\right]}{\left\|\left[a_2\quad b_2\right]\right\|}\\
&= \frac{\left[\mathbf{u}_2^T X^0\quad \mathbf{u}_2^T Y^0\right]}{\left\|\left[\mathbf{u}_2^T X^0\quad \mathbf{u}_2^T Y^0\right]\right\|}\\
&= \frac{\mathbf{u}_2^T\mathbf{Q}^0}{\left\|\mathbf{u}_2^T\mathbf{Q}^0\right\|}
\end{aligned}
$$

Thus, we can compute the tangent using the *second* left eigenvector! This analysis holds for general subdivision curves and gives us the **tangent mask**.

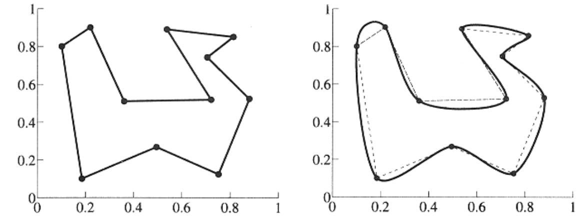
DLG interpolating scheme (1987)

Slight modification to subdivision algorithm:

- splitting step introduces midpoints
- averaging step *only changes midpoints*

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

$$r = \frac{1}{16}(-2, 5, 10, 5, -2)$$



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