



# 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

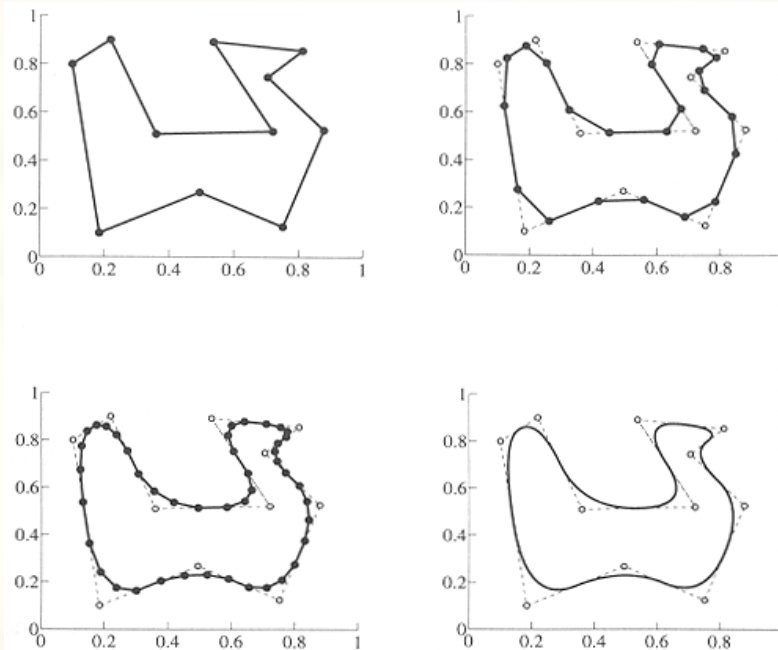
## ■ Idea:

- repeatedly refine the control polygon

$$P^1 \rightarrow P^2 \rightarrow P^2 \rightarrow \dots$$

- curve is the limit of an infinite process

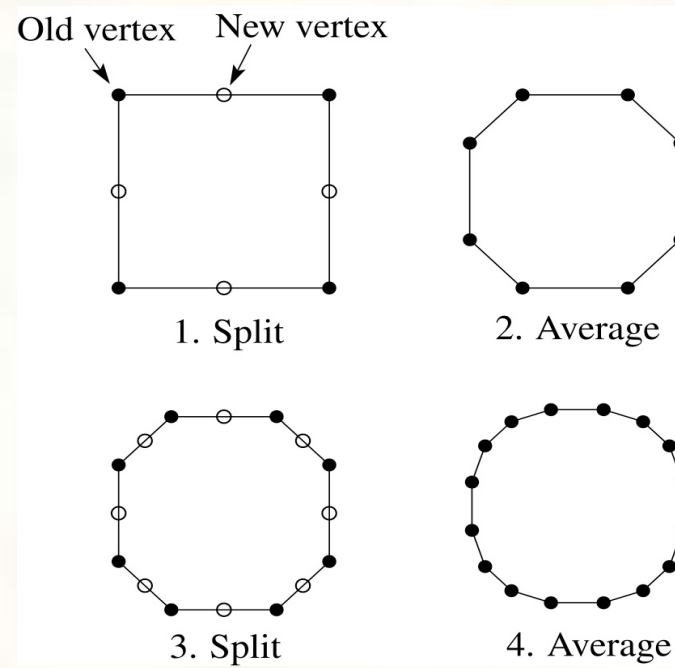
$$Q = \lim_{j \rightarrow \infty} P^j$$





# Chaikin's algorithm

- Chaikin 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:

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

- In the case of Chaikin's algorithm:

$$r = \left( \frac{1}{2}, \frac{1}{2} \right)$$



# Can we generate other B-splines?

- Answer: Yes Lane-Riesenfeld algorithm (1980)
- Use averaging masks from Pascal's triangle:

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

- Gives B-splines of degree  $n+1$ .

- $n=0$ :  $1$

- $n=1$ :  $1 \quad 1$   
 $1 \quad 1$

- $n=2$ :  $1$   
 $1 \quad 1$   
 $1 \quad 2 \quad 1$



# 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!**

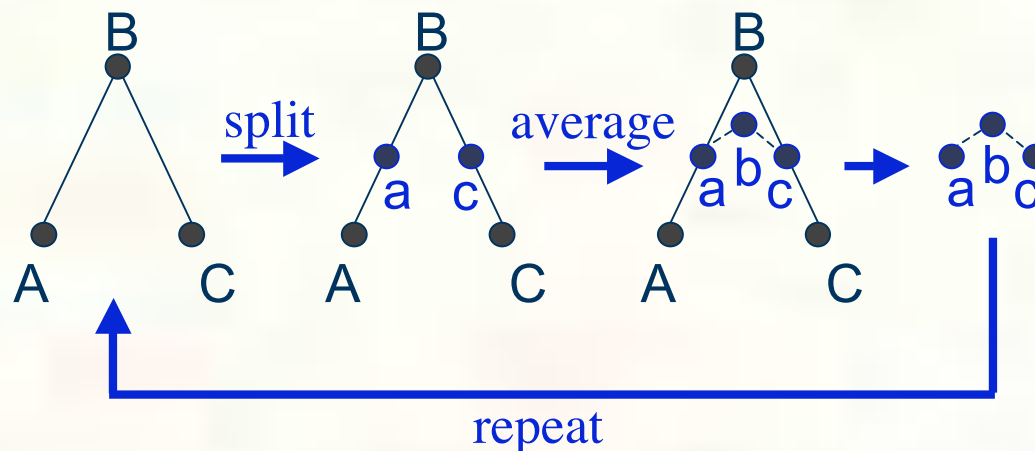


# One subdivision step

- Consider the cubic B-spline subdivision

mask:  $\frac{1}{4}(1 \ 2 \ 1)$

- Now consider what happens during splitting and averaging:





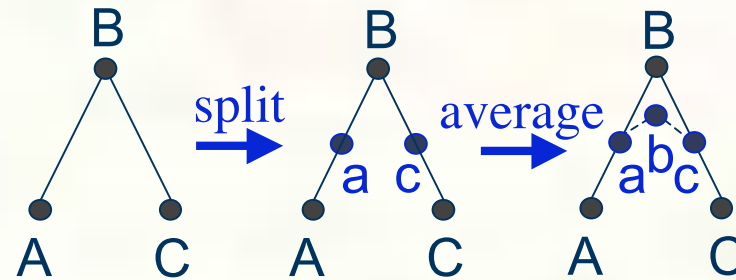


# Math for one subdivision step

- Subdivision mask:

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

- One subdivision step:



Split:  $\mathbf{a} = \frac{1}{2}(\mathbf{A} + \mathbf{B})$

$$\mathbf{c} = \frac{1}{2}(\mathbf{B} + \mathbf{C})$$

Average:

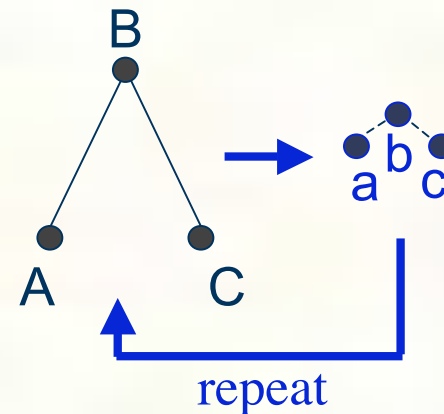
$\mathbf{a}$  and  $\mathbf{c}$  do not change

$$\mathbf{b} = \frac{1}{4}(\mathbf{a} + 2\mathbf{B} + \mathbf{c}) = \frac{1}{8}(\mathbf{A} + 6\mathbf{B} + \mathbf{C})$$



# Consolidated math for one step

- Subdivision mask:  $\frac{1}{4}(1 \ 2 \ 1)$
- One subdivision step:



- Consolidated math for one subdivision step:

$$P_{j+1} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix} = \frac{1}{8} \begin{bmatrix} 4 & 4 & 0 \\ 1 & 6 & 1 \\ 0 & 4 & 4 \end{bmatrix} \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \\ \mathbf{C} \end{bmatrix} P_j$$

Local subdivision matrix 'S'



# Local subdivision matrix, cont'd

- Tracking the point's value through subdivision:

$$P_j = SP_{j-1} = S \cdot SP_{j-2} = S \cdot S \cdot SP_{j-3} = \dots = S^j P_0$$

- The limit position of the point is then:

$$P_\infty = S^\infty P_0$$

- or as we'd say in calculus...

$$P_\infty = \lim_{j \rightarrow \infty} S^j P_0$$

- 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  $\mathbf{S}$ . First, a review...
  - Let  $\mathbf{v}$  be a vector such that  $\mathbf{S}\mathbf{v} = \lambda\mathbf{v}$
  - We say that  $\mathbf{v}$  is an eigenvector with eigenvalue  $\lambda$ .
  - An  $n \times n$  matrix can have  $n$  eigenvalues and eigenvectors:

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

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

$$P = \sum_{i=1}^n a_i\mathbf{v}_i$$



# To infinity, but not beyond...

- So, applying  $S$  to  $P$ :

$$P_1 = SP_0 = S \sum_i^n a_i v_i = \sum_i^n a_i S v_i = \sum_i^n a_i \lambda_i v_i$$

- Applying it  $j$  times:

$$P_j = S^j P_0 = S^j \sum_i^n a_i v_i = \sum_i^n a_i S^j v_i = \sum_i^n a_i \lambda_i^j v_i$$

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

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

- Now let  $j$  go to infinity:  $P_\infty = \lim_{j \rightarrow \infty} S^j P_0 = \lim_{j \rightarrow \infty} \sum_i^n a_i \lambda_i^j v_i$

- 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?

$$\lambda_1 = 1$$

$$v_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$\lambda_2 = \frac{1}{2}$$

$$v_2 = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$$

$$\lambda_3 = \frac{1}{4}$$

$$v_3 = \begin{pmatrix} 2 \\ -1 \\ 2 \end{pmatrix}$$

- We're OK!
- But what is the final position?

$$P_\infty = \lim_{j \rightarrow \infty} (a_1 \lambda_1^j v_1 + a_2 \lambda_2^j v_2 + a_3 \lambda_3^j v_3)$$

$$P_\infty =$$

- Almost done... from earlier we know that we can find 'a<sub>i</sub>', we but didn't give specifics.



# Evaluation masks, cont'd

- To finish up, we need to compute  $a_1$ .

- Remember:  $P_0 = a_1 v_1 + a_2 v_2 + \dots + a_n v_n$

- Rewrite as:

$$P_0 = \begin{bmatrix} \vdots & \vdots & & \vdots \\ v_1 & v_2 & \dots & v_n \\ \vdots & \vdots & & \vdots \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \mathbf{V}A$$

- We need to solve for the vector ‘ $A$ ’.

(This is really just a change of basis for representing the vector  $P$ ).

$$A = \mathbf{V}^{-1}P_0$$

The solution is:

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \dots & u_1^T & \dots \\ \dots & u_2^T & \dots \\ & \vdots & \\ \dots & u_n^T & \dots \end{bmatrix} P_0$$

- Now we can compute the limit position:  $P_\infty = a_1 = u_1^T P_0$
- We call  $u_1$  the **evaluation mask**.



# Evaluation masks, cont'd

- Note that we need not start with the 0<sup>th</sup> 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:

$$P_\infty = S^\infty P_j = u_1^T P_j$$

- So far we've been looking at math for a subdivision function  $f(x)$ .
- For a 2D parametric subdivision curve,  $(x(u), y(u))$ , just apply these formulas separately for the  $x(u)$  and  $y(u)$  functions.





# Recipe for subdivision curves

---

- The evaluation mask for the cubic B-spline is:

$$\frac{1}{6}(1 \quad 4 \quad 1)$$

- 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.



# Derivative of subdiv. function

- What is the tangent to the cubic B-spline function?
- Consider the formula for  $P$  again:

$$P_j = a_1 \lambda_1^j v_1 + a_2 \lambda_2^j v_2 + a_3 \lambda_3^j v_3$$

$$P_j = a_1 (1)^j \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + a_2 \left(\frac{1}{2}\right)^j \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} + a_3 \left(\frac{1}{4}\right)^j \begin{pmatrix} 2 \\ -1 \\ 2 \end{pmatrix}$$

Where:

$$P_j = \begin{bmatrix} \text{left} \\ \text{center} \\ \text{right} \end{bmatrix} \quad P' = \lim_{\Delta x \rightarrow 0} \frac{\text{center} - \text{left}}{\Delta x} = \lim_{j \rightarrow \infty} \frac{\text{center} - \text{left}}{\frac{1}{2^j}}$$

$$\text{Derivative is just: } P' = \lim_{j \rightarrow \infty} \left( a_2 \left(\frac{1}{2}\right)^j \frac{0+1}{\frac{1}{2^j}} \right) = a_2 = u_2^T P_0$$



# Tangent analysis for 2D curve

- What is the tangent to a parametric cubic B-spline **2D curve**?
- Using a similar derivation to what we just did for a 1D function (but omitting details):

$$\mathbf{t} = \lim_{j \rightarrow \infty} \frac{P_{Center,j} - P_{Left,j}}{\|P_{Center,j} - P_{Left,j}\|} = \frac{\mathbf{u}_2^T P_0}{\|\mathbf{u}_2^T P_0\|}$$

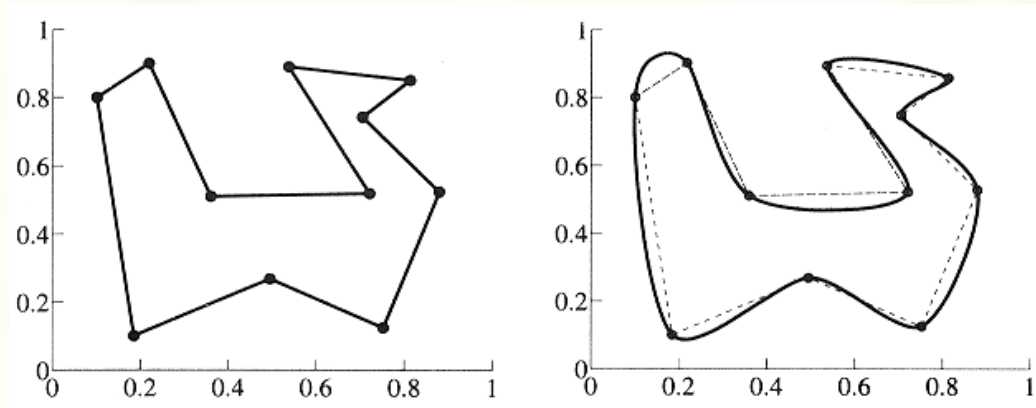
- Thus, we can compute the tangent using the *second* left eigenvector! This analysis holds for general subdivision curves and gives us the **tangent mask**.



# Approximation vs. Interpolation of Control Points

- Previous subdivision scheme *approximated* control points. Can we *interpolate* them?  
Yes: **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.