Meshes and Subdivision

Richard (Hao) Zhang

CMPT 464/764: Geometric Modeling in Computer Graphics

Lecture 4

Outline of 3D shape representations

- Implicit representations
- Parametric representations

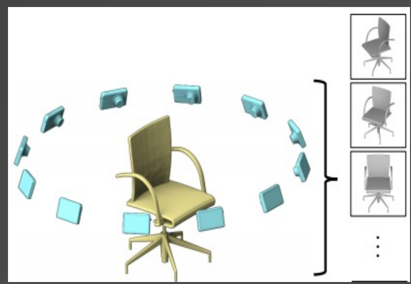
Smooth curves and surfaces

- Meshes (subdivision)
- Point clouds
- Voxels

Discrete representations

- Projective representations

3D → 2D



- Structured representations

Parts + relations = structures
Cover all lower-level part reps

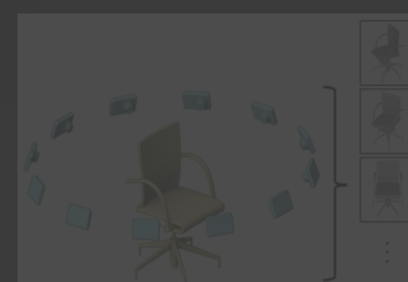
Today

- Implicit representations
- Parametric representations
- Meshes (subdivision)
- Point clouds
- Voxels
- Projective representations
- Structured representations

Smooth curves and surfaces

Discrete representations

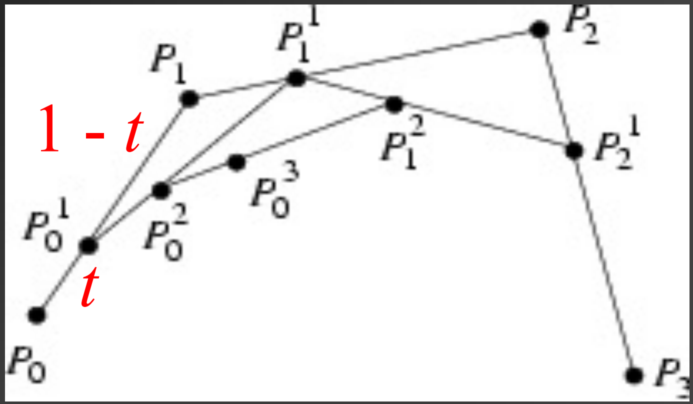
3D → 2D



Parts + relations = structures

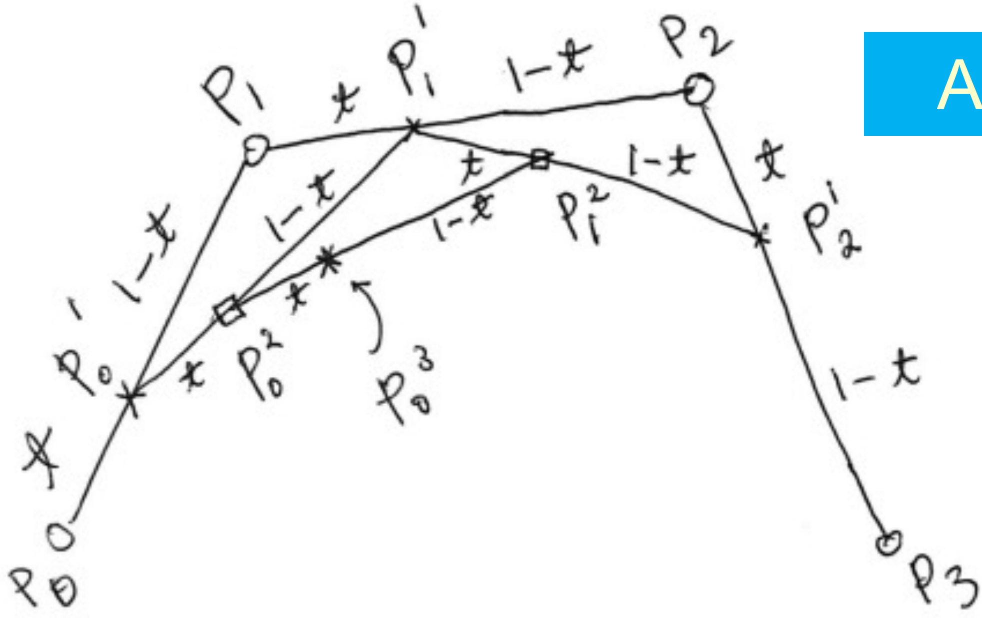
Cover all lower-level part reps

Exercise: identify this curve



- $0 \le t \le 1$: express P_0^1, P_1^1, P_2^1 as linear combinations of $P_0, \ldots, P_3$
- Then express P_0^2 and P_1^2 as linear combinations of P_0^1, P_1^1, and P_2^1
- Finally, express P_0^3 as a linear combination of P_0^2 and P_1^2
- **What is this curve?**

Answer: Cubic Bézier Curve



Bernstein Polynomials of degree 3 :

t^3

$3t^2(1-t)$

$3t(1-t)^2$

$(1-t)^3$

$$P_0^1 = tP_1 + (1-t)P_0$$

$$P_1^1 = tP_2 + (1-t)P_1$$

$$P_2^1 = tP_3 + (1-t)P_2$$

$$P_0^2 = tP_1^1 + (1-t)P_0^1 = t^2P_2 + 2t(1-t)P_1 + (1-t)^2P_0$$

$$P_1^2 = tP_2^1 + (1-t)P_1^1 = t^2P_3 + 2t(1-t)P_2 + (1-t)^2P_1$$

$$P_0^3 = tP_1^2 + (1-t)P_0^2 = \underline{t^3}P_3 + \underline{3t^2(1-t)}P_2 + \underline{3t(1-t)^2}P_1 + \underline{(1-t)^3}P_0$$

Standard derivation of Cubic Bézier

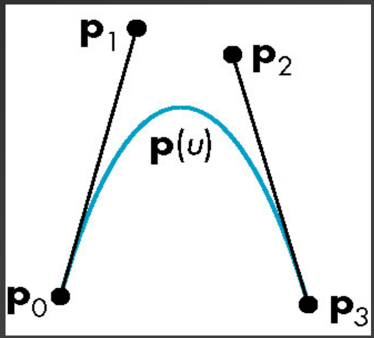
- Defined by four control points P_0, P_1, P_2, and P_3

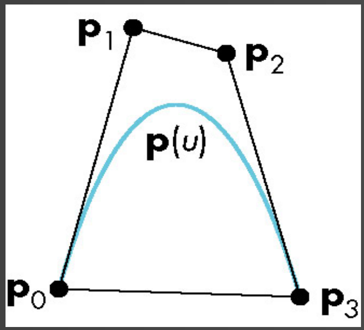
$x(0) = P_0$

$x(1) = P_3$

$x'(0) = 3(P_1 - P_0)$

$x'(1) = 3(P_3 - P_2)$





[Angel 02]

Exercise: derive the cubic Bézier change of basis matrix, following our derivation for the cubic Hermite last week

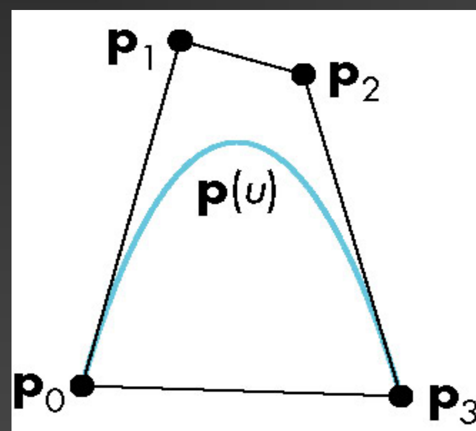
Cubic Bézier change-of-basis matrix

Symmetric matrix!

$$M_{Bezier} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Cubic Bézier: convex hull property

- **Convex hull property**: Bézier curve lies within the convex hull of the four control points – exhibits good design **control**



- Convex hull of a set of points on the plane: tightest convex polygon enclosing the set – why would it be generally useful in graphics?

Convex hull property

- A cubic (degree-n) curve satisfies the convex hull property if it lies within the convex hull of its four (n + 1) control points

- Convex hull property is satisfied if and only if the basis polynomials $b_1(t)$, $b_2(t)$, $b_3(t)$, $b_4(t)$ form a **partition of unity**, that is:

  1. $0 \le b_1(t), b_2(t), b_3(t), b_4(t) \le 1$ for $t \in [0, 1]$, and

  2. $b_1(t) + b_2(t) + b_3(t) + b_4(t) = 1$

- Each curve point is a **convex combination** of the control points

Bézier bases: Berstein polynomials

$B_0(t) = (1 - t)^3$, $B_1(t) = 3t(1 - t)^2$,

$B_2(t) = 3t^2(1 - t)$, $B_3(t) = t^3$

- Well-known as the **Bernstein Polynomials** of degree 3
- Bernstein polynomials of degree n

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

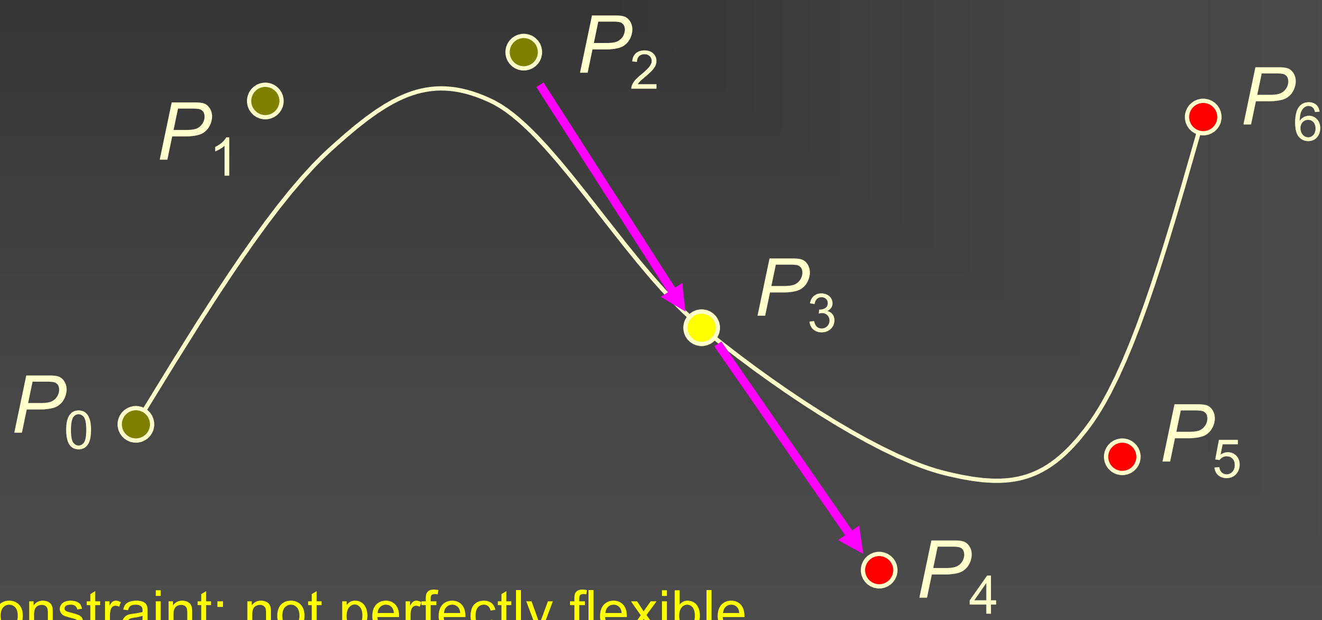
- We have (a recursion)

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t)$$

- Partition of unity easy to see: $\Sigma_i B_i(t) = [t + (1 - t)]^n$

Piecewise cubic Bézier curves

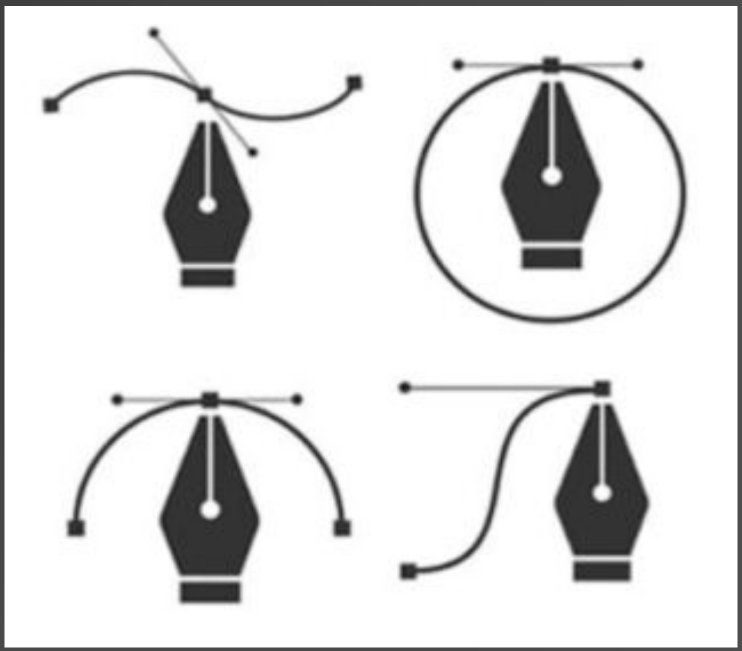
- How to ensure **C**^{1} or **G**^{1} for piecewise Bézier curves?
- Each segment is parameterized over [0, 1] as usual



A constraint: not perfectly flexible

Some practical use of Bézier curves

- To define motion paths and tracks

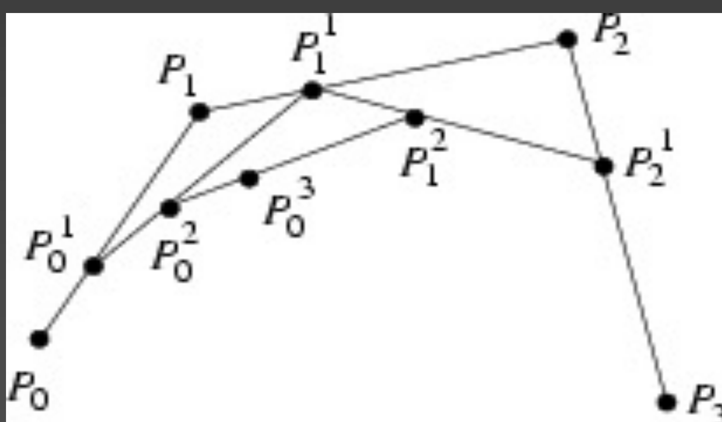


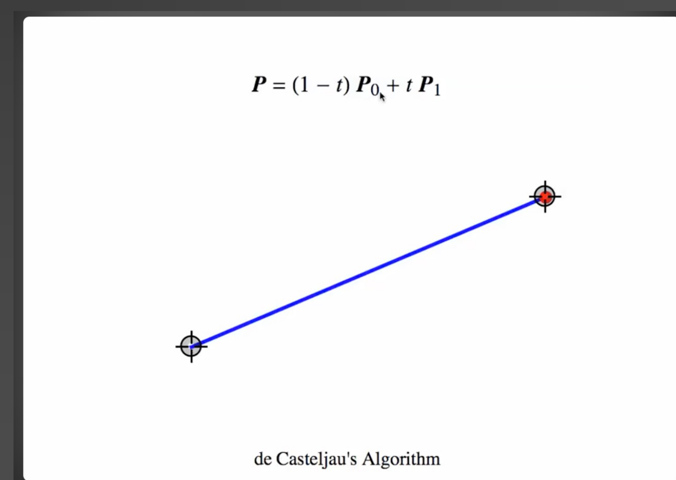
- The Photoshop pen tool

How would you have rendered Bézier?

- Treat as a generic polynomial curve and apply standard polynomial evaluation for each sample point along the curve

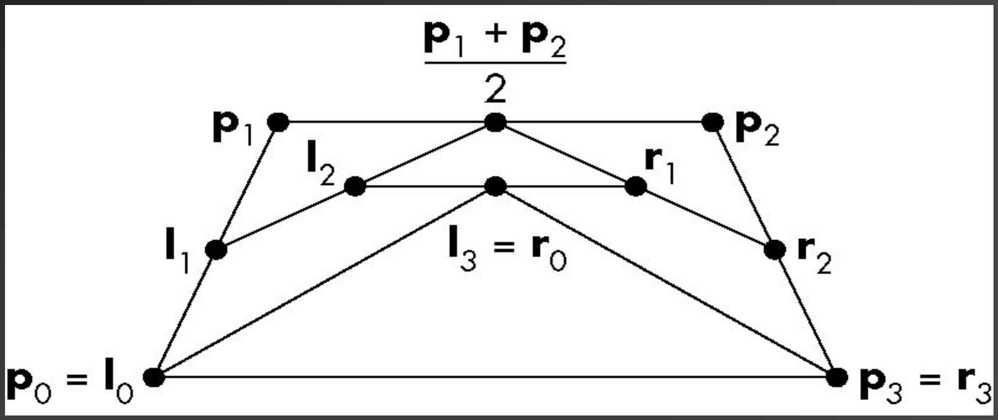
- But Bézier curves are special and there is a nice alternative, using the de Casteljau's procedure below (also see Youtube link)





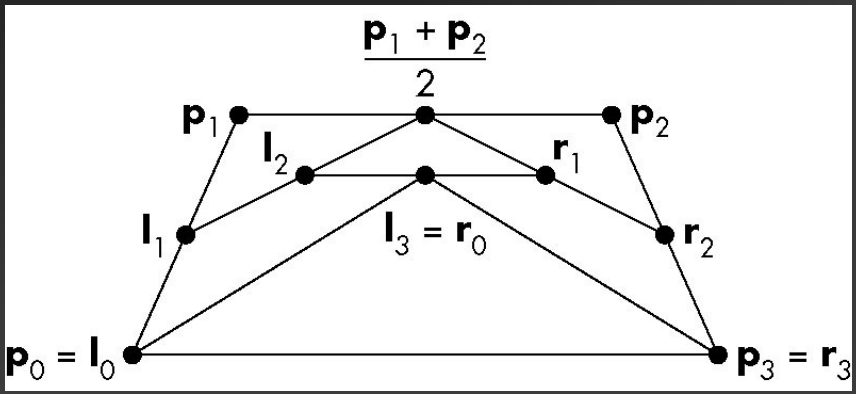
https://www.youtube.com/watch?v=YATikPP2q70

Bézier curve via de Casteljau



- Original four control points P_0, P_1, P_2, P_3 become seven new control points l_0, l_1, l_2, $l_3 = r_0$, r_1, r_2, r_3
- Each set of new control points control half of the Bezier curve
- In the limit, the control points obtained form the Bézier curve determined by P_0, P_1, P_2, P_3

de Casteljau = subdivision



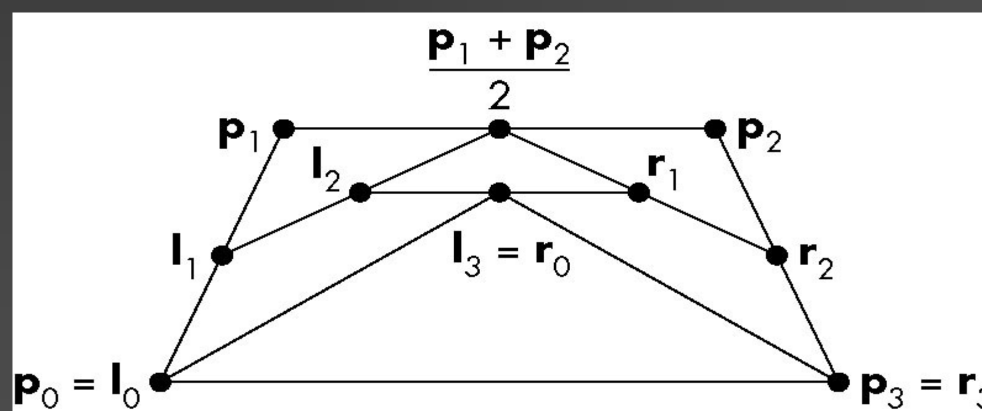
$$\begin{bmatrix} l_0 \\ l_1 \\ l_2 \\ l_3 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 1/4 & 1/2 & 1/4 & 0 \\ 1/8 & 3/8 & 3/8 & 1/8 \\ 0 & 1/4 & 1/2 & 1/4 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

In general, $\mathbf{p}^{(k+1)} = S\mathbf{p}^{(k)}$, S is a **subdivision matrix**

- This is a **subdivision** scheme:
  - Subdivide to obtain new points (**refinement** procedure)
  - New points (l's and r's) are **weighted averages** of the old (P's)
  - Note: de Casteljau's is not interpolatory except at the boundary

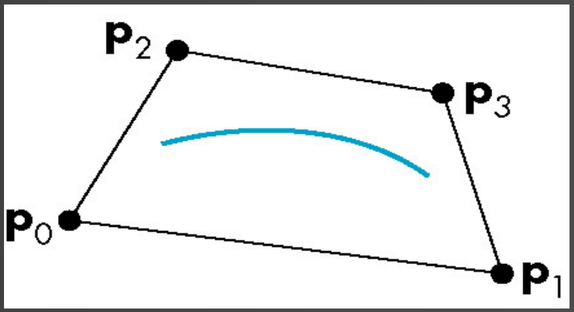
Cubic Bézier via subdivision

- Keep subdividing until sufficiently fine, then connect adjacent control points obtained to form polygonal curve
- A recursive algorithm
- Involve only additions and divisions by 2 — shifts
- **Very fast**
- **Multi-resolution**

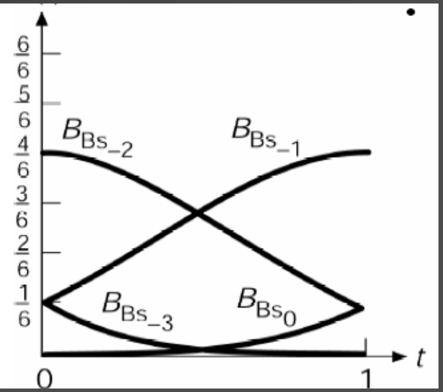


Second example: cubic B-splines

- Each cubic B-spline segment is specified by four control points
- Satisfies the convex hull property
- No interpolation in general
- **Big advantage: C^2 continuous**
- The cubic B-spline change of basis matrix



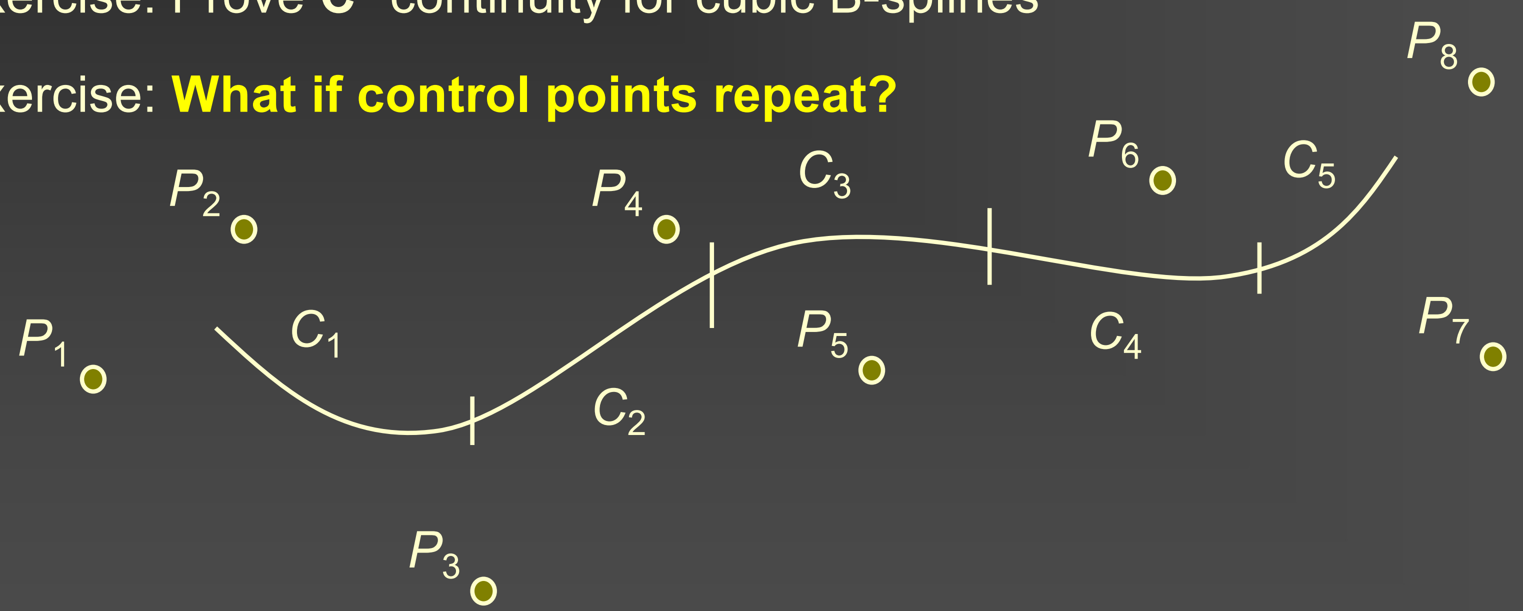
Basis functions



$$M_{B-spline} = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

Piece-wise cubic B-splines

- Two consecutive segments share three control points
- m control points $\rightarrow$ $m - 3$ segments
- Exercise: Prove $\mathbf{C}^2$ continuity for cubic B-splines
- Exercise: **What if control points repeat?**



B-Splines through subvidision

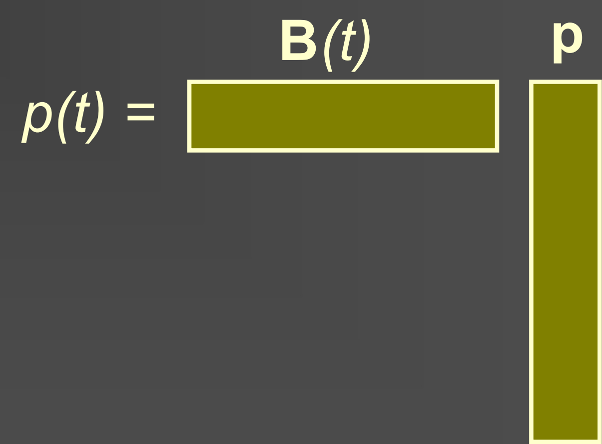
- B-splines can also be generated via subdivision, in the same form

$$\mathbf{c}^{(k+1)} = S\mathbf{c}^{(k)}$$

- Consider any curve represented in l-th degree B-spline basis (the B's)

$$p(t) = \sum_i p_i B_l^i(t)$$

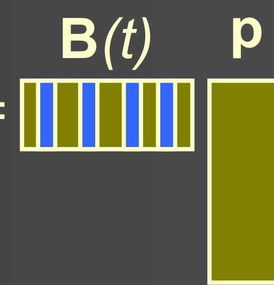
where l is the B-spline degree, i the index, and p_i's are control points.



- In matrix form, we have $p(t) = \mathbf{B}(t)\,\mathbf{p}$, where
  $\mathbf{p}$: column vector of control points
  $\mathbf{B}(t)$: row vector of B-spline bases

B-Splines via subdivision

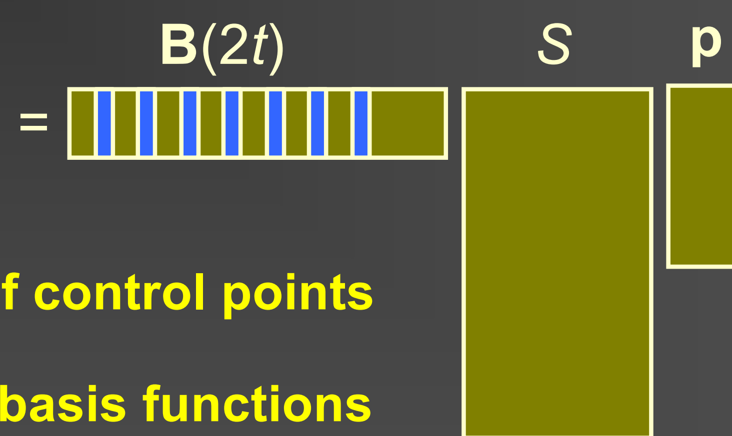
- Continue from matrix representation: $p(t) = \mathbf{B}(t)\,\mathbf{p}$ =



- Eventually, we shall rewrite

$$p(t) = \mathbf{B}(t)\,\mathbf{p} = \mathbf{B}(2t)\,S\,\mathbf{p}$$

where



  - S is the subdivision matrix
  - $\mathbf{p}' = S\,\mathbf{p}$ is the **new, refined set of control points**
  - $\mathbf{B}(2t)$ represent **refined B-spline basis functions**

- Let us focus on **uniform B-splines**

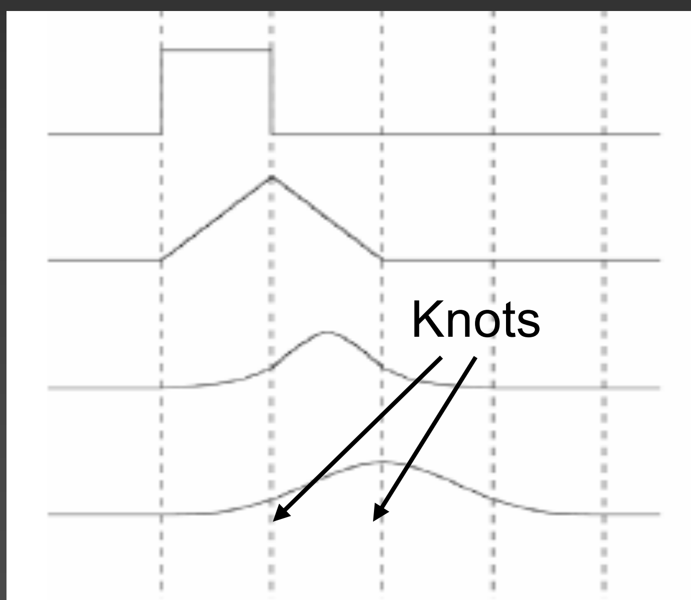
What are splines?

- An m-th degree spline is a **piecewise polynomial** of degree m that is $\mathbf{C}^{m-1}$
- A spline curve is defined by a **knot sequence;** the knots are at parametric t values where the **polynomial pieces join**
- Most common are **uniform** knot spacing, i.e., $t = 0, 1, 2, \ldots$ Nonuniform knot spacing or repeated knots are also possible
- A **spline basis** often serves as a blending function with **local control**
- Resulting spline curve is given by a set of control points blended by **shifted or translated** versions of the spline basis

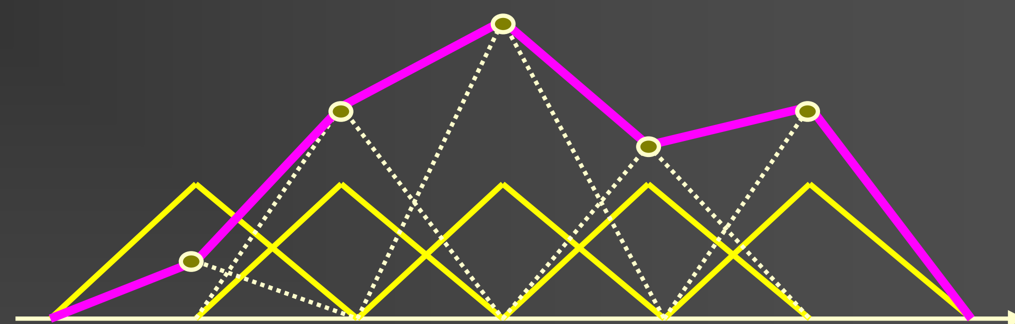
Example: uniform B-splines

- B-splines: one particular class of spline curves

Degree 0-3 uniform B-splines



Note **local control** and **increased continuity**

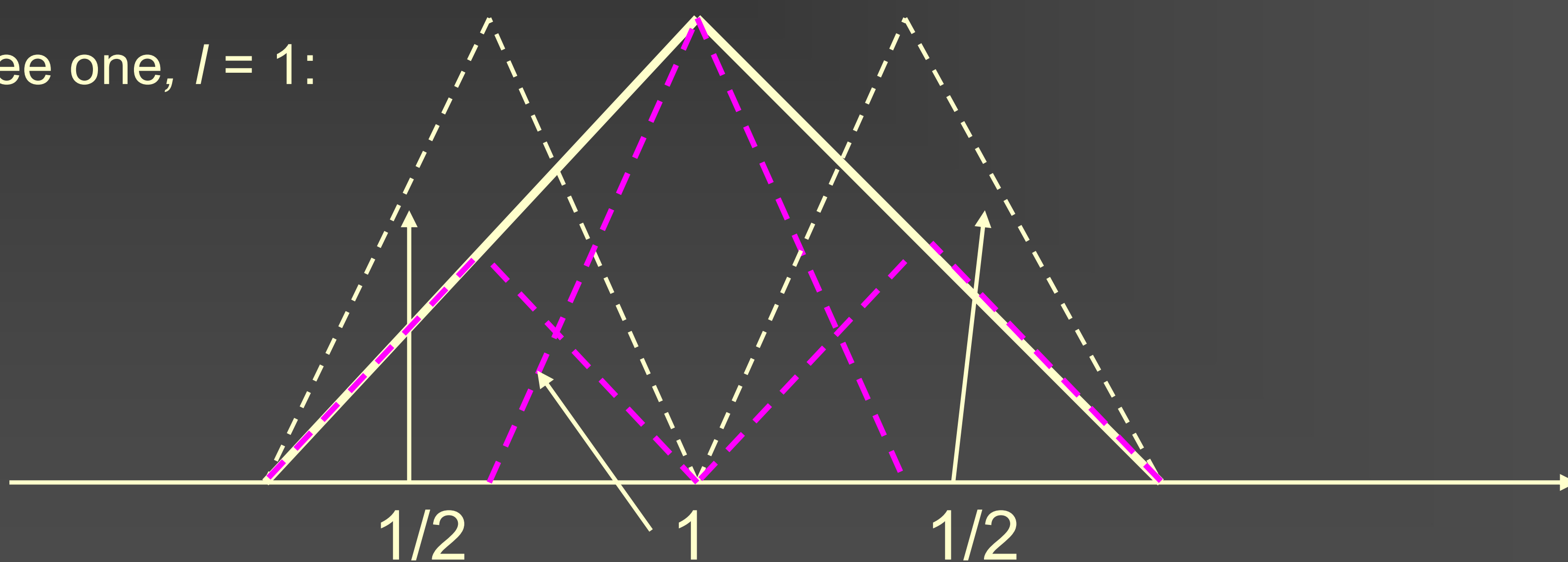


A piecewise linear curve (C^0) obtained by blending five uniform degree-1 B-splines with control points

Key property of uniform B-splines

- A uniform B-spline can be written as a **linear combination** of translated (k) and **dilated** or **compressed** ($2t$) copies of itself
- This is the key to connect B-splines to subdivision

Degree one, $l = 1$:



“Self refinement”

- B-spline of degree l, $B_l(t)$, is $\mathbf{C}^{l-1}$ continuous, $l \geq 1$
- The i-th B-spline, B^i_l, is simply a **translate** of the B-spline $B_l(t)$ or $B_l^0(t)$: $B^i_l(t) = B_l(t - i)$ — right shift of i units
- B-splines satisfy the **refinement equation**

$$B_l(t) = \frac{1}{2^l}\sum_{k=0}^{l+1}\binom{l+1}{k}B_l(2t-k)$$

— binomial coefficients

$$B(t) \xrightarrow{\text{compress then translate by } k/2} B(2t-k)$$

  - A uniform B-spline can be written as a linear combination of translated (k) and **dilated / compressed** ($2t$) copies of **itself**
  - Again, this is the key to connect B-splines to subdivision

B-spline via subdivision

- Using the refinement equation from last slide, we have

$$\mathbf{B}(t) = \mathbf{B}(2t)\, S$$

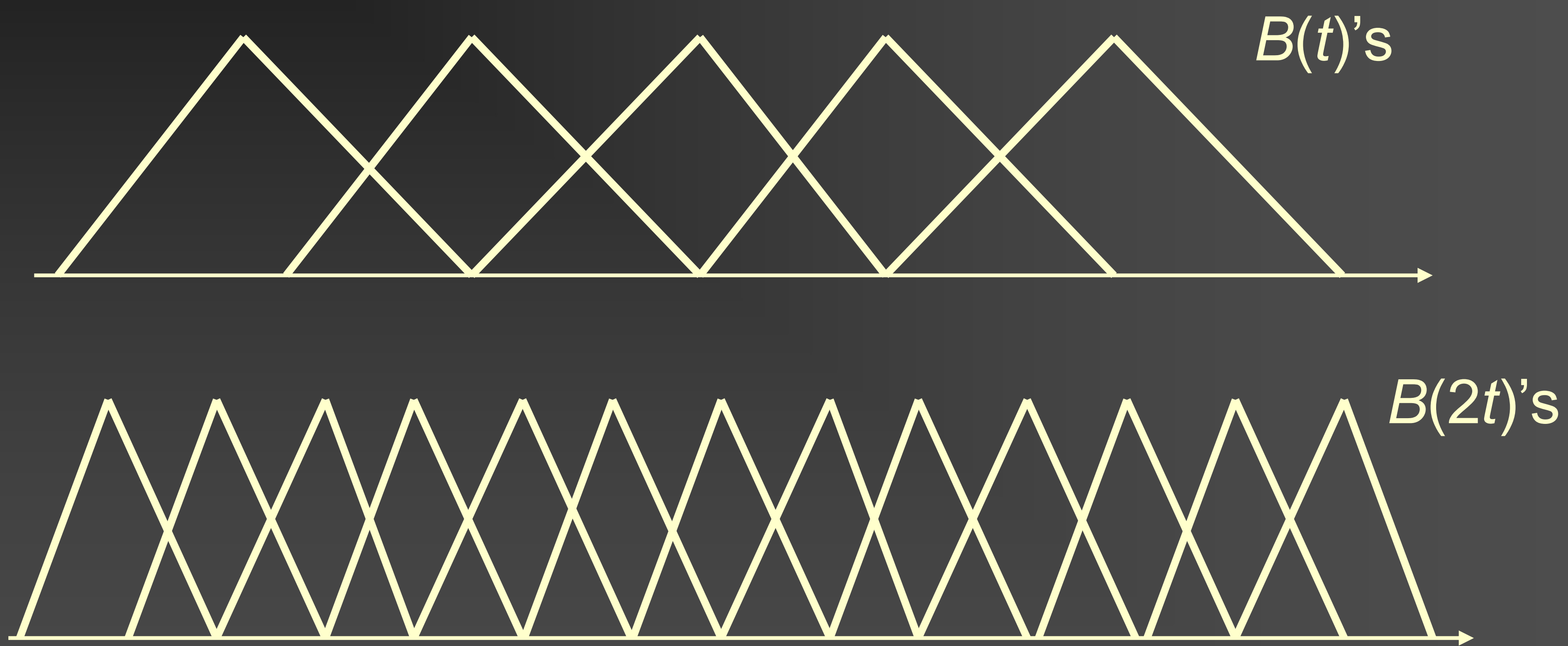
where the entries of S are given by

$$S_{2i+k,i} = s_k = \frac{1}{2^l}\binom{l+1}{k}$$

- Thus, $p(t) = \mathbf{B}(t)\,\mathbf{p} = \mathbf{B}(2t)\, S\mathbf{p}$

We have changed B-spline bases $\mathbf{B}(t)$ to $\mathbf{B}(2t)$, where each element of $\mathbf{B}(2t)$ is half as wide as one in $\mathbf{B}(t)$ and the sequence in $\mathbf{B}(2t)$ are spaced twice as dense

Refinement of B-splines



Linear B-spline case; this extends to B-splines of any degree.

What have we done?

- **Refined the B-spline basis** functions, and at the same time,
- **Refined the set of control points p**
- Twice as many new control points $\mathbf{p}' = S\mathbf{p}$:
  - One new point (an **odd point**) is inserted between two consecutive control points in **p**
  - Each control point in **p** (an **even point**) is either retained (**interpolatory**) or moved (**approximating**) in $\mathbf{p}'$
- S is the **subdivision matrix**

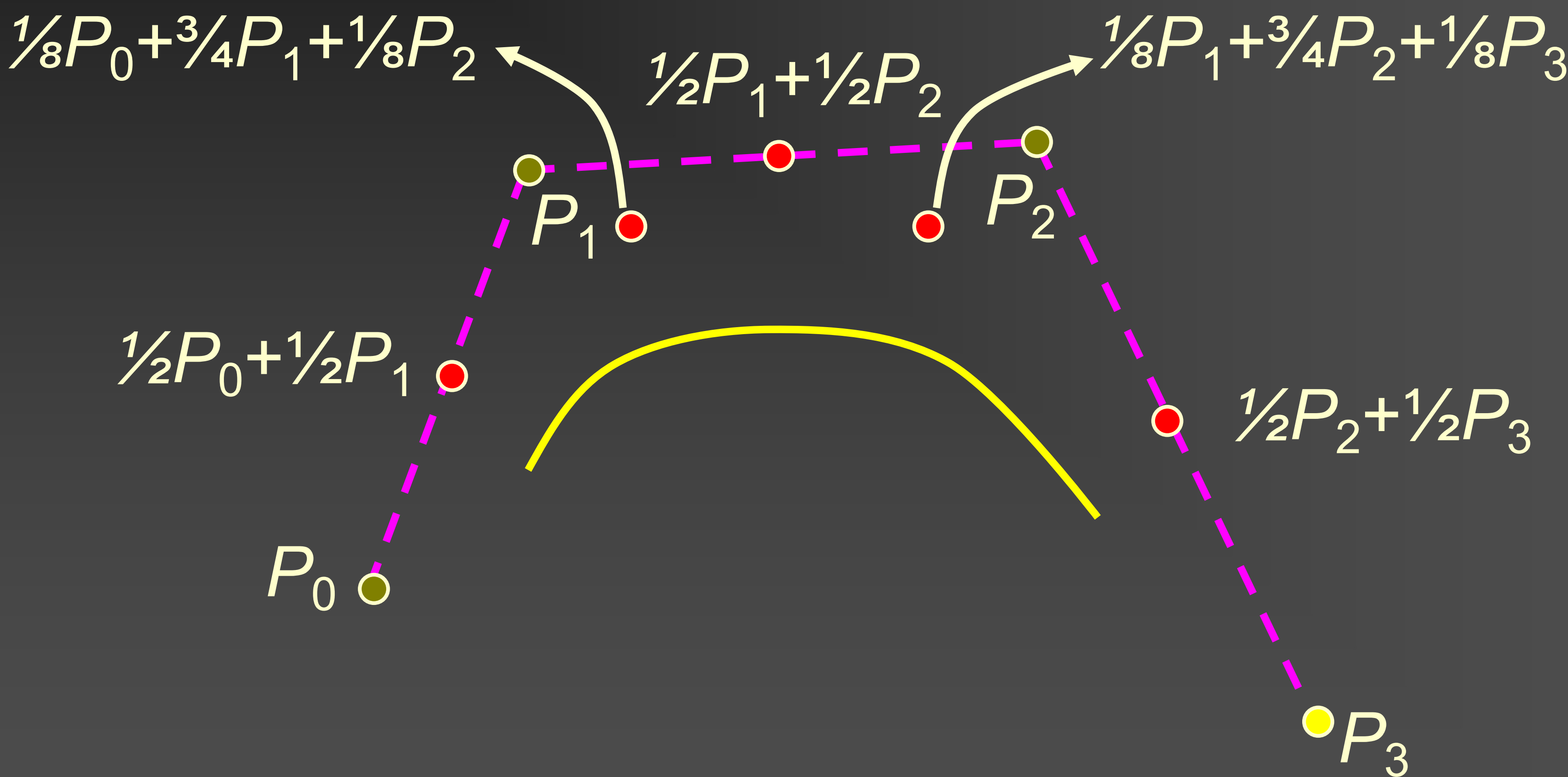
Subdivision matrix *S*

$$
\begin{array}{r}
\\
\\
\\
\\
\text{even points} \\
\text{odd points} \\
\\
\\
\\
\end{array}
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
1/2 & 1/2 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 1/2 & 1/2 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1/2 & 1/2 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1/2 & 1/2 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\qquad
\begin{array}{r}
\\
\\
\\
\\
\\
\text{even points} \\
\text{odd points} \\
\\
\\
\\
\\
\end{array}
\begin{bmatrix}
4/8 & 0 & 0 & 0 & 0 \\
6/8 & 1/8 & 0 & 0 & 0 \\
4/8 & 4/8 & 0 & 0 & 0 \\
1/8 & 6/8 & 1/8 & 0 & 0 \\
0 & 4/8 & 4/8 & 0 & 0 \\
0 & 1/8 & 6/8 & 1/8 & 0 \\
0 & 0 & 4/8 & 4/8 & 0 \\
0 & 0 & 1/8 & 6/8 & 1/8 \\
0 & 0 & 0 & 4/8 & 4/8 \\
0 & 0 & 0 & 1/8 & 6/8 \\
0 & 0 & 0 & 0 & 4/8
\end{bmatrix}
$$

for linear uniform B-spline

for uniform cubic B-splines

Cubic B-splines via subdivision



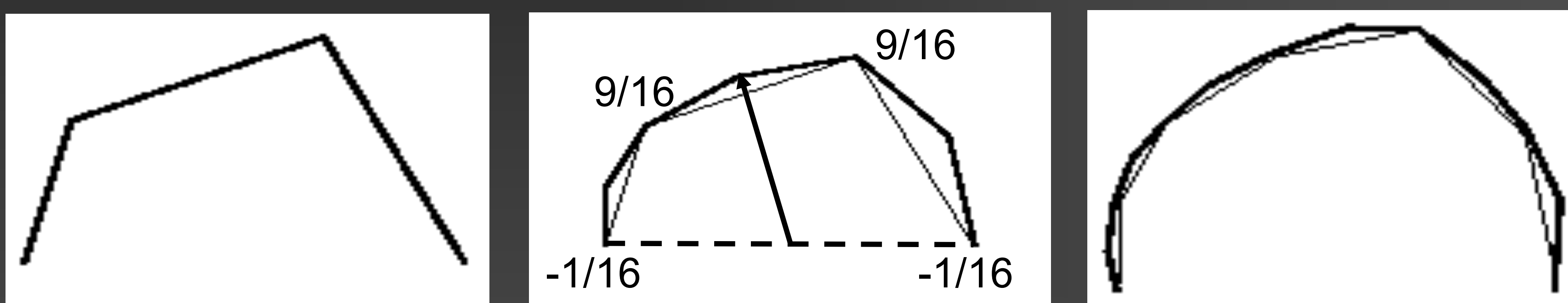
Convergence of subdivision (aside)

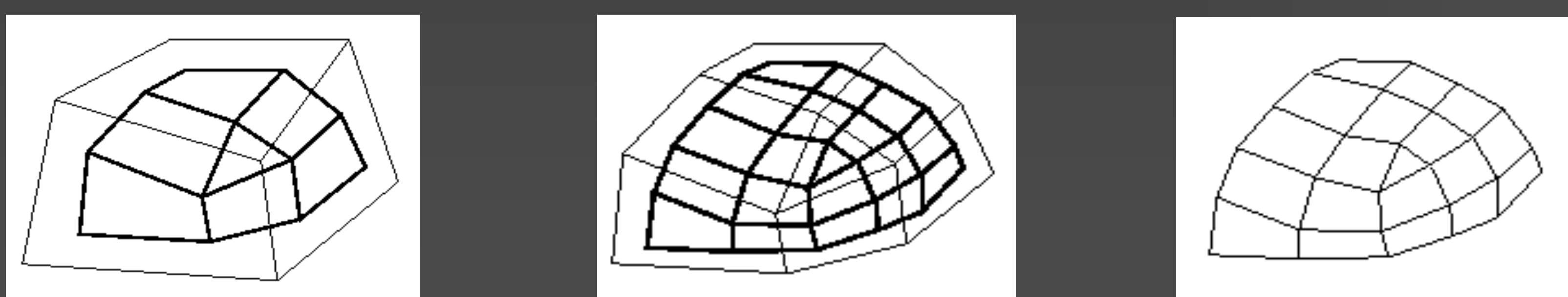
$$\mathbf{p}^{j} = S^{j}\,\mathbf{p}^{0}$$

- The recursively refined set of control points converge to the actual spline curve $p(t) = \sum \mathbf{p}_i\, B_i^{l}(t)$
- Have **geometric rate of convergence**, i.e., difference decrease by constant factor (see notes) — $\|\varepsilon^{j}\| < c\gamma^{j}$
- Can thus obtain spline curves via subdivision, just like de Casteljau for Bezier curves!

Idea of subdivision

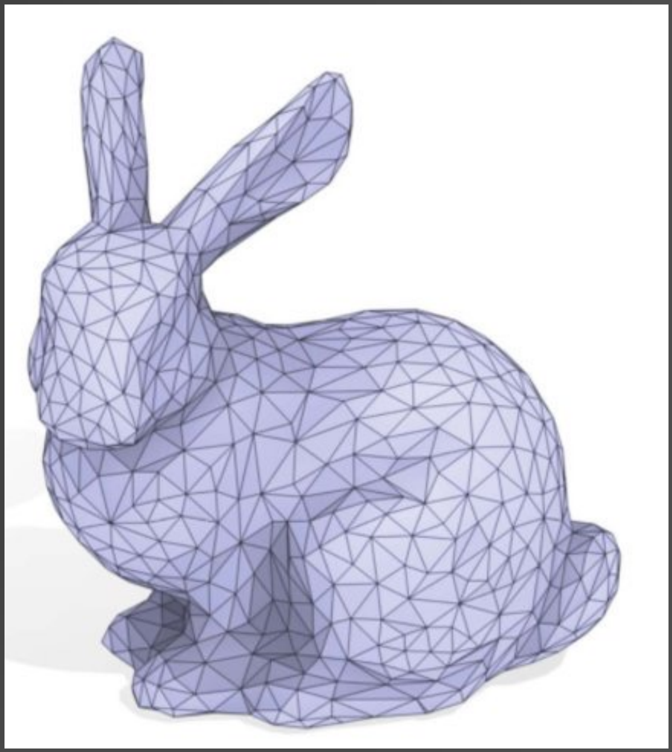
- A subdivision curve (or surface) is the **limit** of a sequence of successively refined control polygon (or control **mesh**)





What are (polygonal) meshes?

- Polygonal mesh: composed of a set of polygons pasted along their edges – triangles most common
- Still **most popular** in graphics and CAD

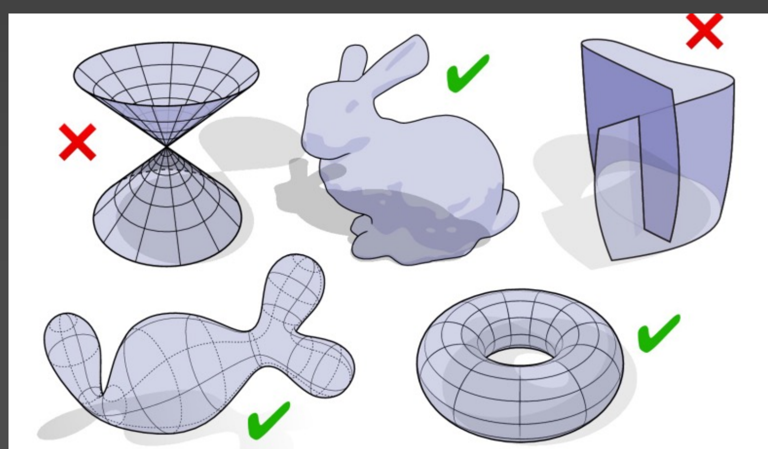


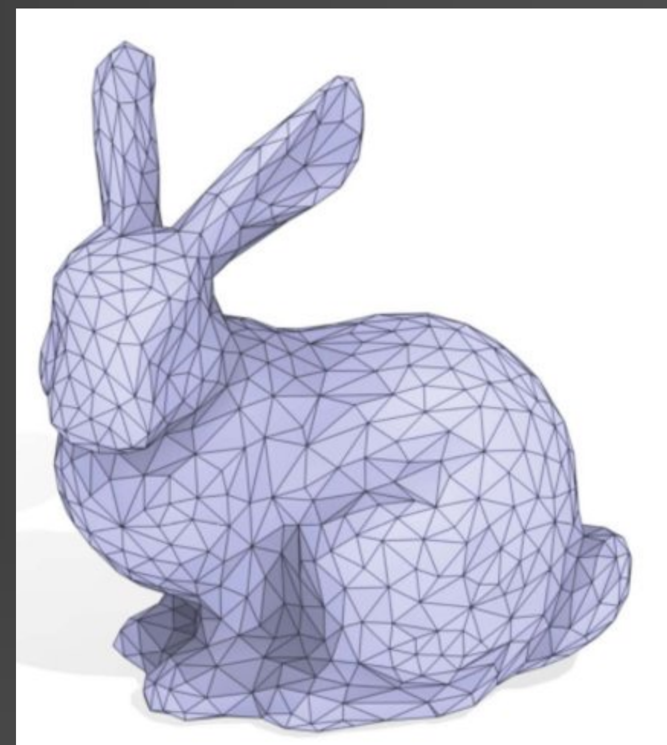
A triangle bunny mesh

What are (polygonal) meshes?

- Polygonal mesh: composed of a set of polygons pasted along their edges – triangles most common
- Still **most popular** in graphics and CAD
- Basic mesh components and properties:

  vertices, edges, faces, valences, normal, curvature, boundaries, **manifold or not**

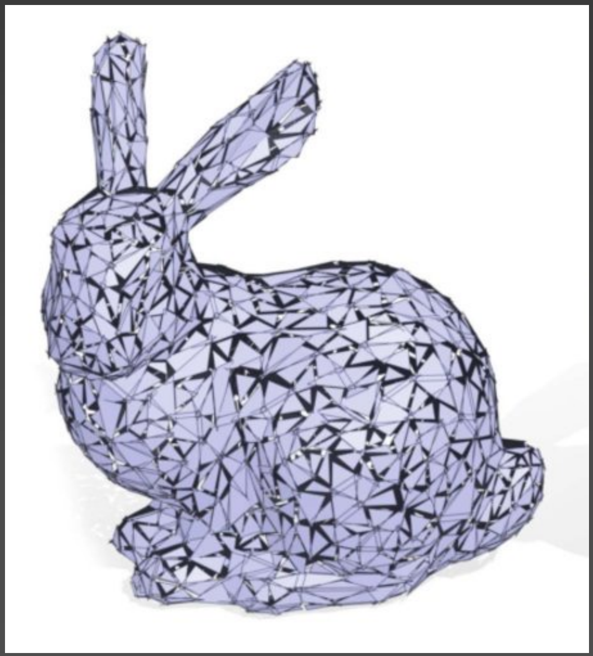




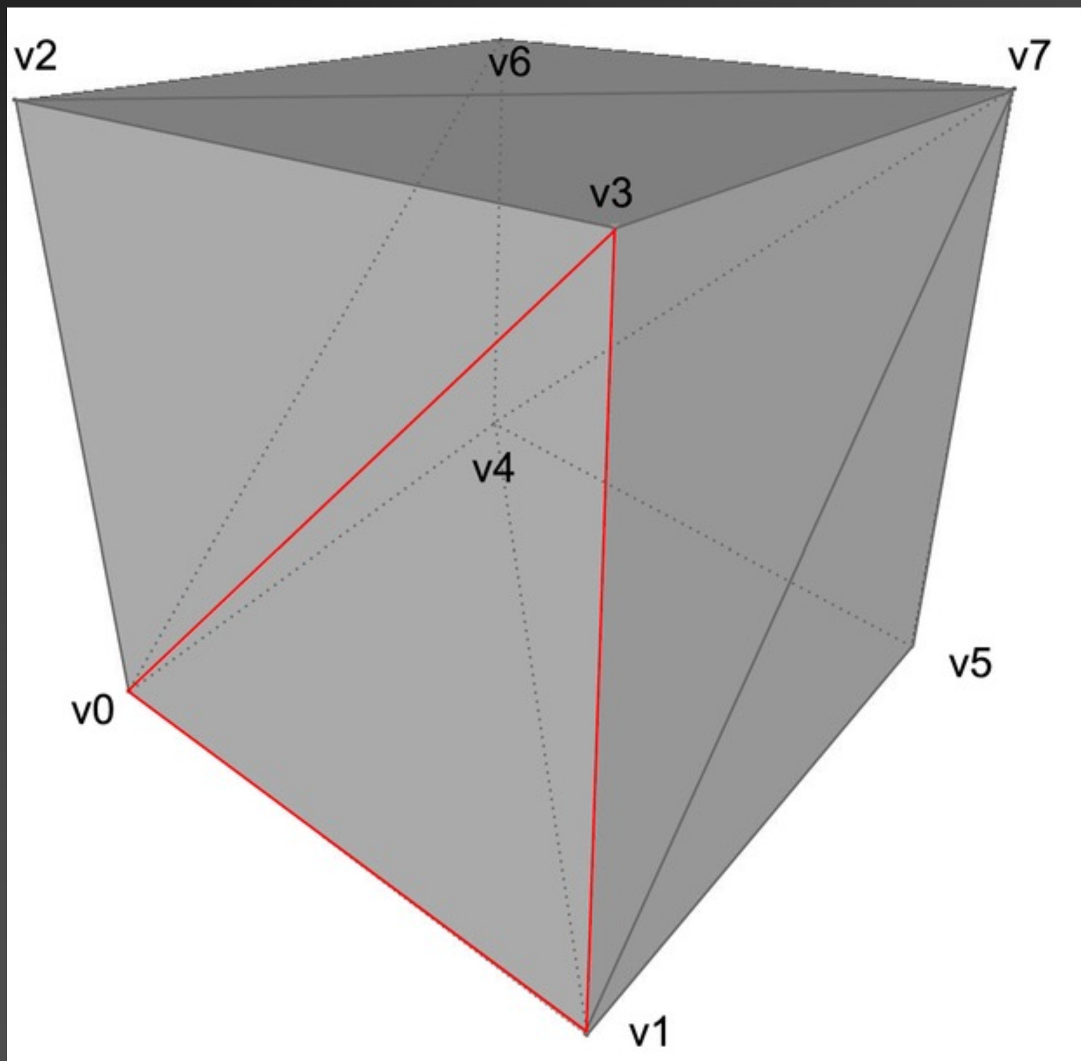
A triangle bunny mesh

Polygon soup

- For each triangle, just store 3 coordinates, no connectivity information
- Not much different from point clouds
- MobileNeRF is a polygon soup
- 3DGS is a dense “soup” of Gaussians



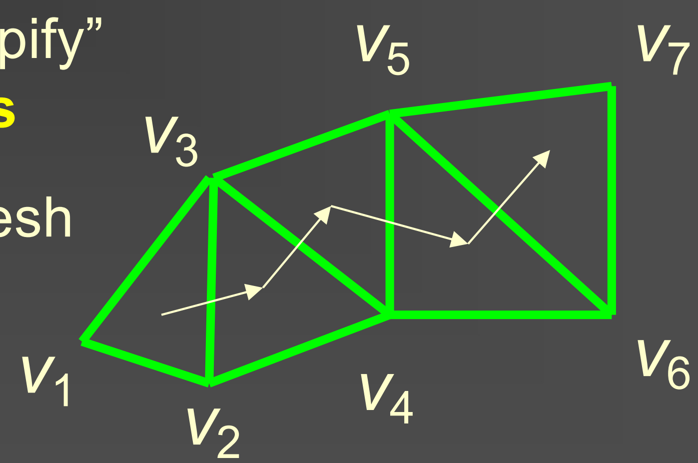
Mesh storage format: OBJ




```
v 0.0 0.0 0.0
v 0.0 0.0 1.0
v 0.0 1.0 0.0
v 0.0 1.0 1.0
v 1.0 0.0 0.0
v 1.0 0.0 1.0
v 1.0 1.0 0.0
v 1.0 1.0 1.0
f 0 6 4
f 0 2 6
f 0 3 2
f 0 1 3
f 2 7 6
f 2 3 7
f 4 6 7
f 4 7 5
f 0 4 5
f 0 5 1
f 1 5 7
f 1 7 3
```

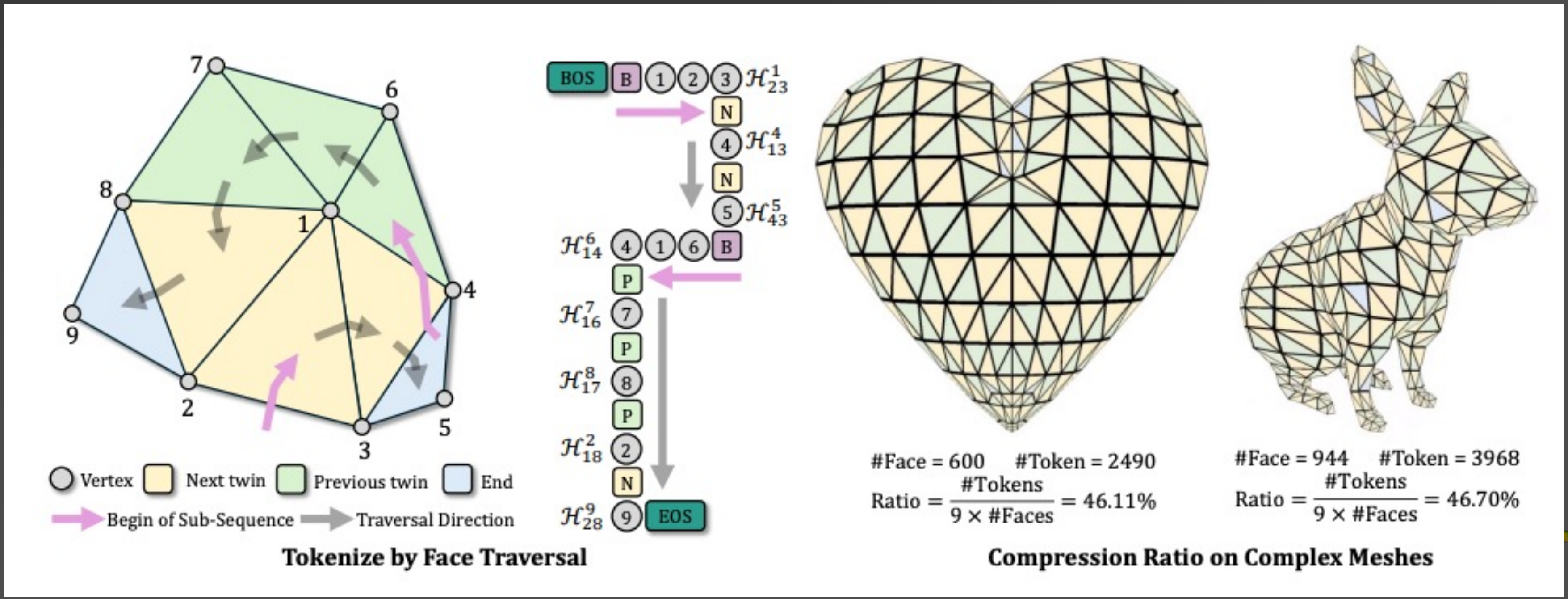
Efficient storage: triangle strips

- A triangle strip gives a **compact** way of representing a set of triangles
- For *n* triangles in a strip, instead of passing through and transform $3n$ vertices, only need $n+2$ vertices
- In a sequence, e.g., v_1, v_2, …, first three vertices form the first triangle; subsequent vertex forms new triangle with its preceding two vertices
- Algorithms have been developed to “stripify” a triangle mesh into **long triangle strips**
- Triangle strips: a way to “**serialize**” a mesh



2025: neural mesh generation (aside)

- Key: to utilize the powerful **transformer architecture**
- Need to "**tokenize**" a mesh by traversing triangle/polygon sequences
- Quite a few examples, e.g., EdgeRunner [Tang et al., ICLR 2025]



Back to subdivision

- An effective and efficient way to model and render smooth curves and surfaces, e.g., Bezier and B-splines, via **local refinement**
- Two aspects:
  - **Topological rule**: where to insert new vertices? Are old vertices kept?
  - **Geometrical rule**: spatial location of the new vertices – typically given as an average of nearby new or old vertices
- First introduced to graphics by Ed Catmull and Chaikin in the 1970’s
- One of the most intensely studied subjects of geometric modeling (1990’s) and ubiquitous in modeling and animation software today

Ed Catmull

Former President, Pixar and Walt Disney Animation Studios

Dr. Ed Catmull is a co-founder of Pixar Animation Studios, and served as President of Pixar for 33 years, while also serving as President of Walt Disney Animation Studios for 13 of those 33 years. Previously, Dr. Catmull was vice President of the Computer Division of Lucasfilm Ltd., where he managed development in the areas of computer graphics, video editing, video games and digital audio.

Catmull's book, the New York Times bestseller, “CREATIVITY, INC.: Overcoming the Unseen Forces That Stand in the Way of True Inspiration,” and which Fast Company described as "what just might be the most thoughtful management book ever," was published by Random House in 2014.

Dr. Catmull is an ACM Turing Award Laureate; he has also been honored with five Academy Awards, including two Oscars for his work and a Lifetime Achievement Award.

Dr. Catmull founded three of the leading centers of computer graphics research – including the Computer Division of Lucasfilm Ltd. and Pixar Animation Studios. These organizations have been home to many of the most academically respected researchers in the field and have produced some of the most fundamental advances in computer graphics, including image compositing, motion blur, subdivision surfaces, cloth simulation and rendering techniques, texture mapping and the z-buffer. Dr. Catmull is one of the architects of the RenderMan rendering software, which has been used in over 90% of Academy Award® winners for Visual Effects over the past 20 years.

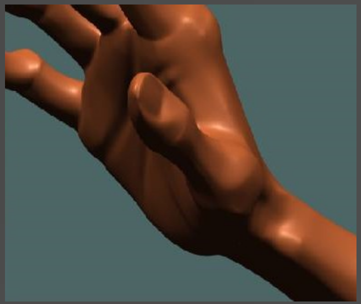
Dr. Catmull is active in several professional organizations. He has been a dedicated participant in ACM SIGGRAPH for nearly 40 years. Dr. Catmull is a member of The Academy of Motion Picture Arts and Sciences, the National Academy of Engineering, and the Visual Effects Society. He has received numerous awards from these organizations.

Dr. Catmull earned B.S. degrees in each of physics and computer science and a Ph.D. in computer science from the University of Utah. In addition, he has received honorary Doctorates from the University of Utah and Johns Hopkins University.

He retired as President of Pixar and Disney Animation in 2019.

Subdivision surfaces in animation

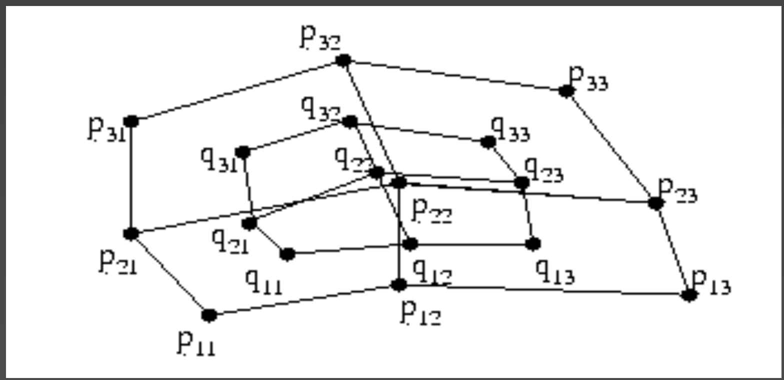
- Geri’s game: Academy award for animated short (1998)
- Use of subdivision surfaces Geri’s skins, clothing, etc.



https://www.youtube.com/watch?v=uMVtpCPx8ow

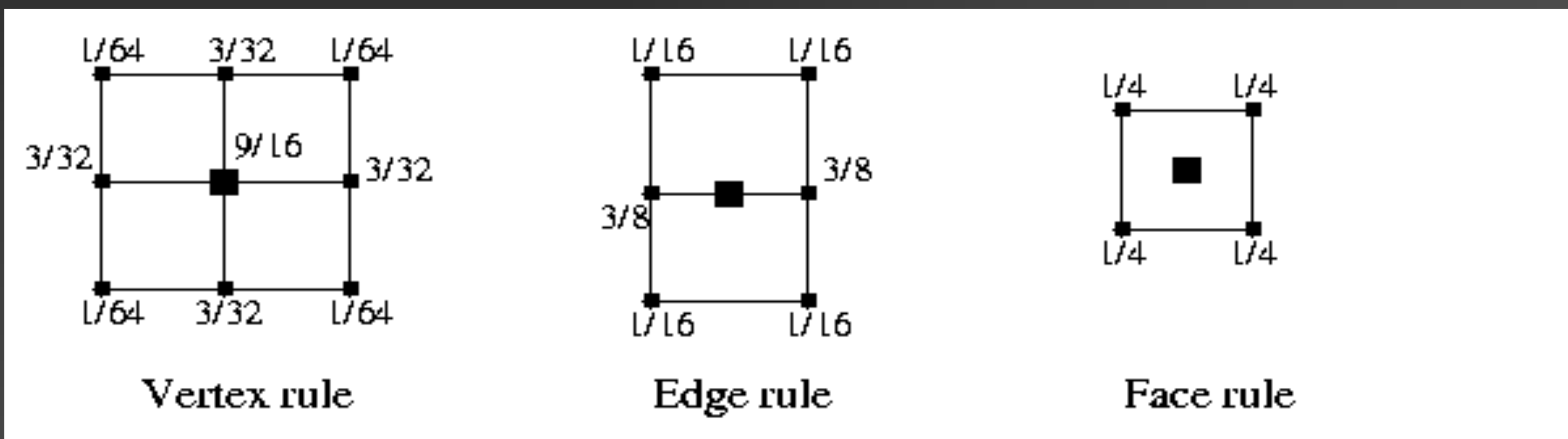
Surface example: Catmull-Clark

- Works on quadrilateral meshes
- Topological rules:
  - One new point per face and edge; retain the old vertices (not positions)
  - Connect face point with all adjacent edge points
  - Connect old vertex with all adjacent edge points



Catmull-Clark subdivision

- Geometric rules (**subdivision masks** shown below)

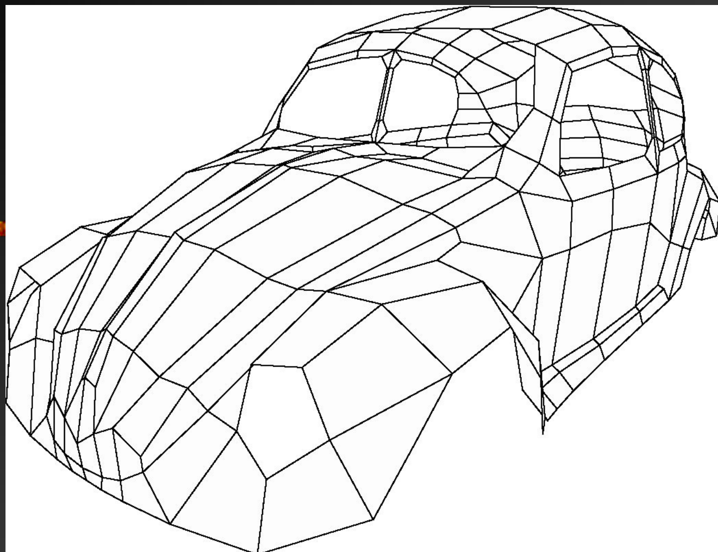
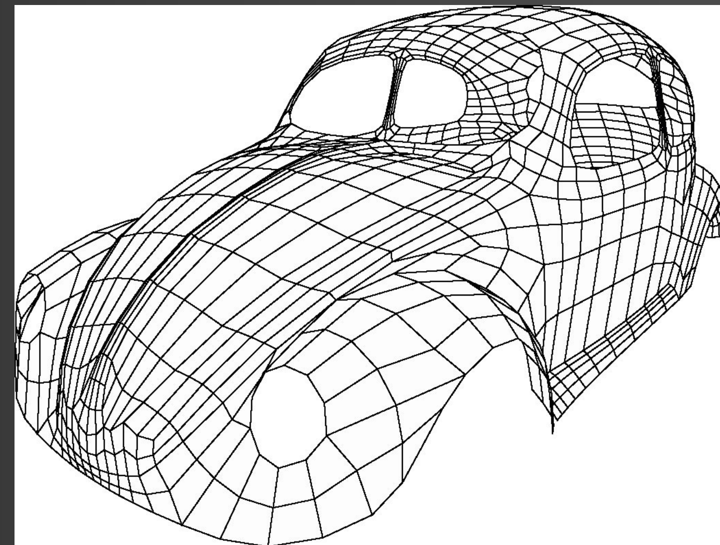
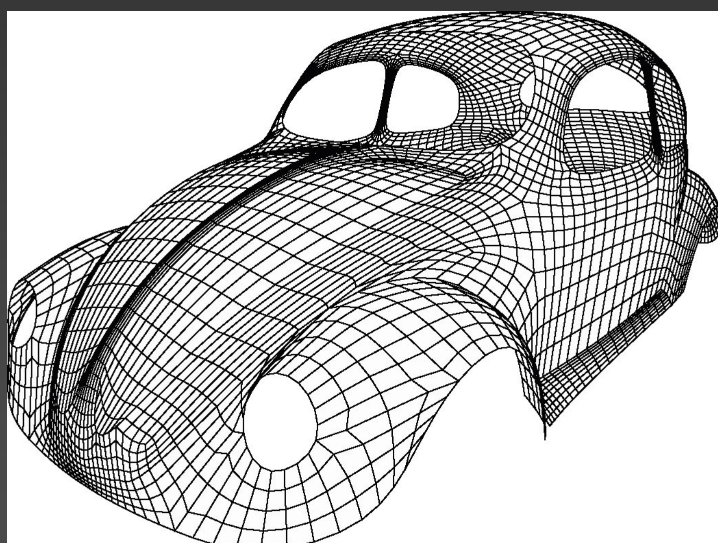


- This is all nice if the quadrilateral mesh connectivity is **regular**, i.e., a rectangular grid, but not always the case

Extraordinary vertices

- In a quadrilateral mesh, a vertex whose valence is not 4 is called an **extraordinary vertex**
- In a triangle mesh, an extraordinary vertex has valence $\neq 6$
- Geometric rules for extraordinary vertices are different

Exercise: For a closed triangle mesh, can all vertices have degree 6?


Catmull-Clark and B-splines

- Even if original mesh has faces other than quadrilaterals, after one subdivision, all faces become quadrilaterals
- **Number of extraordinary vertices never increase**
- Over rectangular (regular) region, the limit is **bicubic B-spline surface**, i.e., $\mathbf{C}^2$
- Continuity at extraordinary vertices: $\mathbf{C}^1$
- There are many other types of subdivision surfaces with different schemes giving different levels of continuity

Advantages

- **Efficient to compute/render** with simple algorithms: weighted averages within a local neighborhood
- Flexible **local control** of surface features
- **Provable smoothness** if well designed
- **One-piece and seamless**; can model surfaces with arbitrary topology (same topology as control mesh) with relative ease
- **Compact** representation: base mesh + (fixed) rules
- Natural **level-of-detail** (hierarchical) representation

Subdivision surface vs. mesh

- Subdivision surfaces are **smooth limit surfaces**
- But in practice, e.g., rendering, only a few subdivisions are needed to produced a **mesh** that is dense enough
- **Polygonal meshes**: a much more general geometric representation
  - Does not have to result from subdivision – **irregular connectivity** vs. **subdivision connectivity**
  - Typically obtained from discretization of math representation or reconstruction out of a **point cloud**

Derivation of B-spline basis

- … via **convolution**
- Recall: B-spline bases defined by a **knot sequence**
- In uniform case (**uniform B-splines**), i.e., uniform spacing of the knots, B-spline basis can be defined via repeated **convolution**

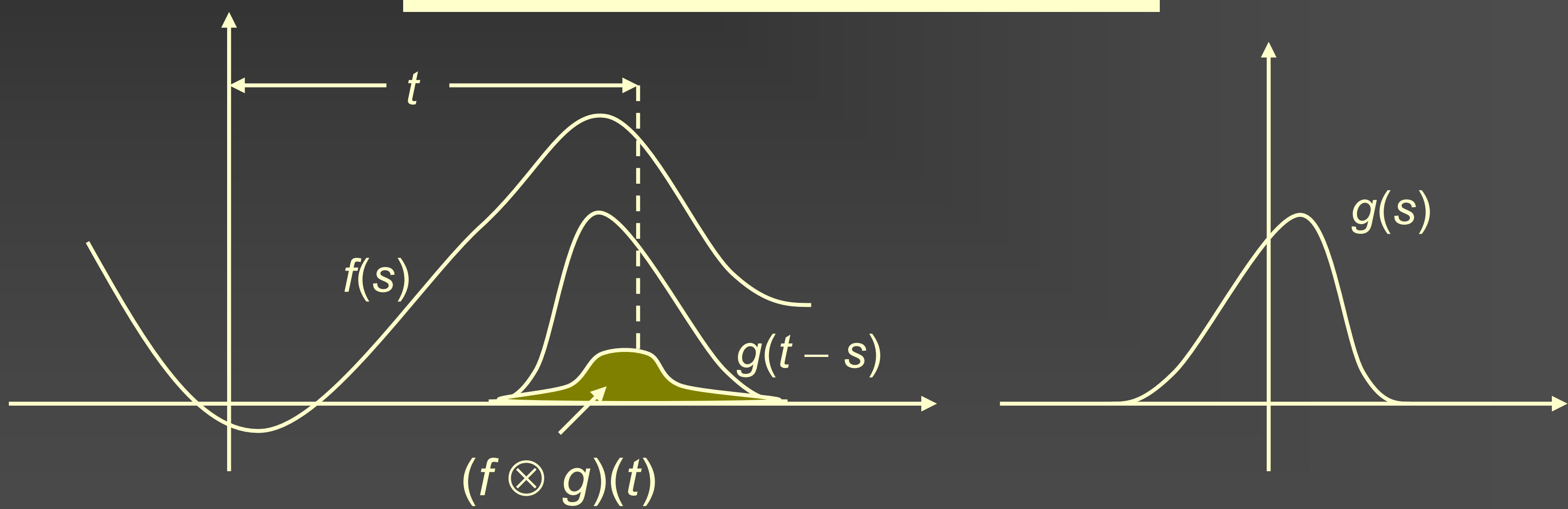
$$B_l(t) = (B_{l-1} \otimes B_0)(t) = \int B_{l-1}(s) B_0(t-s)\,ds$$

- $B_0(t)$, degree-0 B-spline, is the **box function** at $t = 0$

Convolution

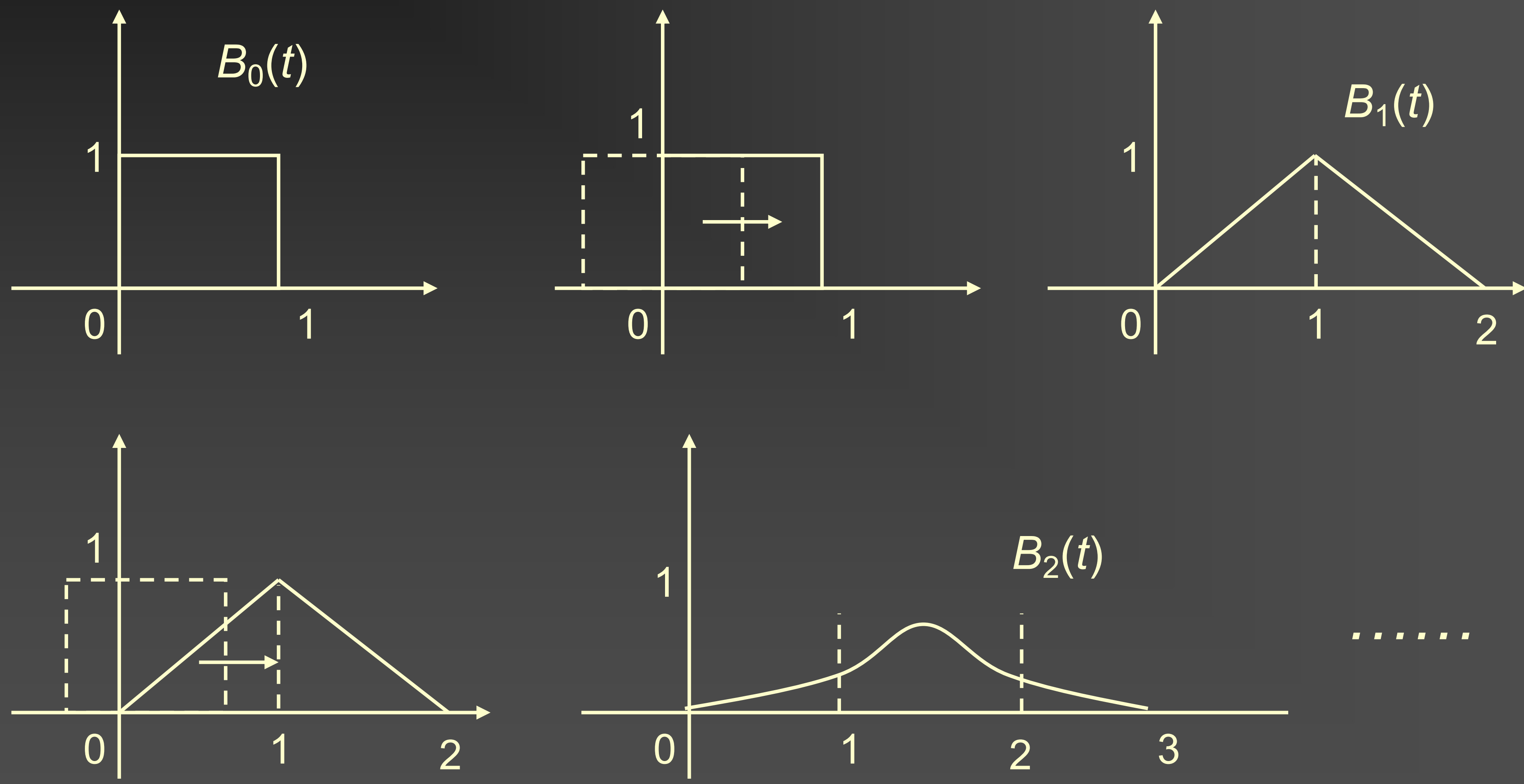
- An integral that computes a “running weighted average”

$$(f \otimes g)(t) = \int f(s) g(t-s) ds$$



- Kernel/weighting function *g* is often **symmetric** about 0

B-splines via convolution



A few words on convolution (aside)

- Function g first **reversed**: differ from **cross correlation**
  - To ensure **commutativity**: $f \otimes g = g \otimes f$
  - Convolution is also **associative**: $f \otimes (g \otimes h) = (f \otimes g) \otimes h$
  - And **distributive over addition**: $f \otimes (g + h) = f \otimes g + f \otimes h$
- Discrete convolution in 1D: **serial products**
  - $\{f_0, f_1, \ldots, f_{m-1}\} \otimes \{g_0, g_1, \ldots, g_{n-1}\} = \{f_0 g_0, f_0 g_1 + f_1 g_0, \ldots, f_{m-1} g_{n-1}\}$
  - Length of resulting sequence: $n + m - 1$
  - Matrix formulation: multiplication by a **Toeplitz matrix**
  - **Circular convolution** defined by a **circulant matrices**, i.e., $C_{ij} = C_{kl}$ if and only if $i - j \equiv k - l \pmod{n}$

Important properties of subdivision

- **Convergence:**
  - Sequence of control polygons/meshes approach some continuous limit curve/surface
- **Interpolation – only for some subdivision schemes**
  - Possible with interpolating subdivision schemes, e.g., Butterfly (next)
- **Local control:**
  - Allows local change to a shape, e.g., through lifting of a single vertex
  - Local change does not influence the shape globally
  - This is a result of having **local subdivision rules**, i.e., geometric results only depend on information in a small local neighborhood

Important properties (continued)

- **Affine invariance:**
  - To transform a shape, it is sufficient to explicitly transform its (compact) set of control points
  - New shape is reconstructed (via subdivision) in transformed domain
  - This is related to the **row sum** of the subdivision matrix
- **Smoothness:**
  - The limit curve/surface should be smooth: a local property
  - Related to **eigenvalues** of the subdivision matrix

Subdivision matrix is key

- Subdivision matrix *S* characterizes the scheme
- Most relevant properties are derived from the subdivision matrix, e.g., local control (sparseness), convergence, smoothness, etc.
- First example, consider **affine invariance**
  - Requires $S\mathbf{1} = \mathbf{1}$, i.e., $[1\ 1\ \ldots\ 1]^T$ is an eigenvector of *S* with eigenvalue 1
  - Equivalently, *S* needs to have **unit row sum**
  - Proof?

Affine invariance (aside)

- Original vector of m points in dimension k: $u \in \mathrm{R}^{m \times k}$
- Vector of n points after subdivision: $v = Su \in \mathrm{R}^{n \times k}$, $n > m$
- Subdivision matrix $S \in \mathrm{R}^{n \times m}$
- Affine transformation of a point $p \in \mathrm{R}^{k \times 1}$ in dimension k: $p \rightarrow Ap + b$

Affine transform of subdivided points v:

$$v \rightarrow (Av^{\mathrm{T}} + b\mathbf{1}_n^{\mathrm{T}})^{\mathrm{T}} = [A(Su)^{\mathrm{T}} + b1_n^{\mathrm{T}}]^{\mathrm{T}}$$

$$= SuA^{\mathrm{T}} + \mathbf{1}_n b^{\mathrm{T}}$$

Subdivide affine transformed points u:

$$u \rightarrow S(Au^{\mathrm{T}} + b\mathbf{1}_m^{\mathrm{T}})^{\mathrm{T}}$$

$$= SuA^{\mathrm{T}} + S\mathbf{1}_m \mathrm{b}^{\mathrm{T}}$$

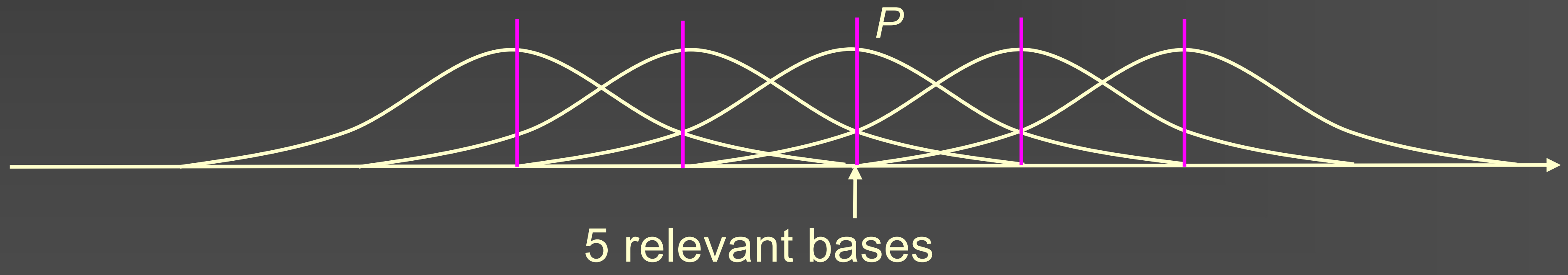
Results are equivalent if $S\mathbf{1}_m = \mathbf{1}_n$, implying **unit row sum** for S

Smoothness of limit curve/surface

- Analyze the behavior of a subdivision scheme **on or near a particular control point**
- To study smoothness, we care not only about point locations, but also **existence of tangent** line/plane at the point in question, etc.
- So far, we have assumed subdivision matrix is bi-infinite
- To obtain a finite subdivision matrix, need to decide which control points influence the **neighborhood** of the point of interest
- Typically, the neighborhood structure does not change through subdivision — **invariant neighborhood**

Invariant neighborhood

- Consider spline curves represented by spline basis functions
- To decide which control points influence the behavior of the spline curve near a particular point P …
- Look at how many spline bases influence P's neighborhood
- As an example, consider cubic B-splines



Invariant neighborhood in subdivision (aside)

- Let us look at subdivision …
- Generally, and without a picture to help, note that

Final curve, i.e., polygonal curve joining control points after *j*-th level subdivision

Linear B-spline basis at refinement level *j* (i.e., the hat function)

control points obtained after *j*-th level subdivision

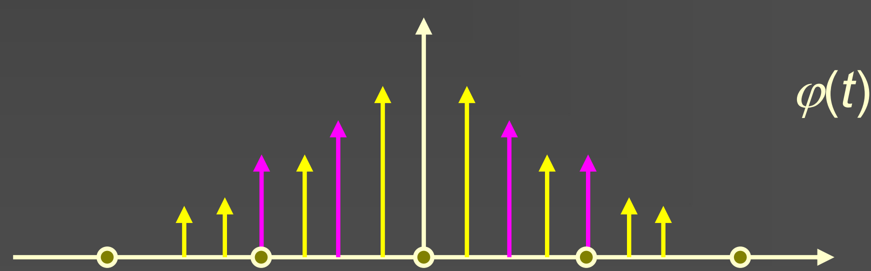
$$p^j(t) = B_1(2^j t)\, p^j = B_1(2^j t)\, S^j p^0$$
$$= B_1(2^j t)\, S^j \left[\sum_i p_i^0 \mathbf{e}_i\right] = \sum_i p_i^0 \left[B_1(2^j t)\, S^j \mathbf{e}_i\right] = \sum_i p_i^0 \varphi_i^j(t)$$

Canonical basis vectors (or impulse vectors)

Invariant neighborhood in subdivision (aside)

$$p^j(t) = \sum_i p_i^0 \varphi_i^j(t) \Rightarrow p^\infty(t) = \sum_i p_i^0 \varphi_i(t), \quad \varphi_i(t) = \lim_{j\to\infty} \varphi_i^j(t)$$

- Each $\varphi_i(t)$ is the result of subdividing an **impulse**
- For **stationary** subdivision (i.e., **fixed subdivision rules**), $\varphi_i(t) = \varphi_0(t - i)$, i.e., they are all the same, just translates of each other
- $\varphi(t)$: the **fundamental solution** of the subdivision
- To determine size of invariant neighborhood, look at the influence of the fundamental solution
- E.g., for cubic B-spline subdivision, influence is 4 unit intervals, so **5 nearby control points influence the center point**



Local subdivision matrix

- Subdivision matrix is $n \times n$ if invariant neighborhood size is n

Cubic B-spline subdivision:

$$\begin{bmatrix} p_{-2}^{j+1} \\ p_{-1}^{j+1} \\ p_{0}^{j+1} \\ p_{1}^{j+1} \\ p_{2}^{j+1} \end{bmatrix} = \frac{1}{8} \begin{bmatrix} 1 & 6 & 1 & 0 & 0 \\ 0 & 4 & 4 & 0 & 0 \\ 0 & 1 & 6 & 1 & 0 \\ 0 & 0 & 4 & 4 & 0 \\ 0 & 0 & 1 & 6 & 1 \end{bmatrix} \begin{bmatrix} p_{-2}^{j} \\ p_{-1}^{j} \\ p_{0}^{j} \\ p_{1}^{j} \\ p_{2}^{j} \end{bmatrix}$$

- E.g., **local subdivision matrix** for cubic B-spline is 5×5
- Let us use **eigenanalysis** of subdivision matrix S to determine limit behavior about the point p_0^∞

Eigenvalues and eigenvectors

- For cubic B-spines
  - Eigenvalues

$$\begin{bmatrix} \lambda_0 & \lambda_1 & \lambda_2 & \lambda_3 & \lambda_4 \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{8} & \frac{1}{8} \end{bmatrix}$$

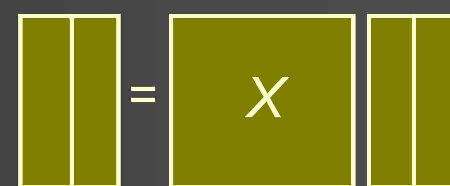
  - Complete set of eigenvectors

$$\begin{bmatrix} 1 & -1 & 1 & 1 & 0 \\ 1 & -1/2 & 2/11 & 0 & 0 \\ 1 & 0 & -1/11 & 0 & 0 \\ 1 & 1/2 & 2/11 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Eigen-analysis

- For eigenanalysis to apply, eigenvectors of S need to form a basis, i.e., **linear independence**
- Not all subdivision schemes satisfy this (e.g., four-point scheme)
- Assume set of eigenvectors x_i's are linearly independent, write the vector of (2D or 3D) control points as

$$p = \sum_{j=0}^{n-1} x_i a_i = X\mathbf{a}$$



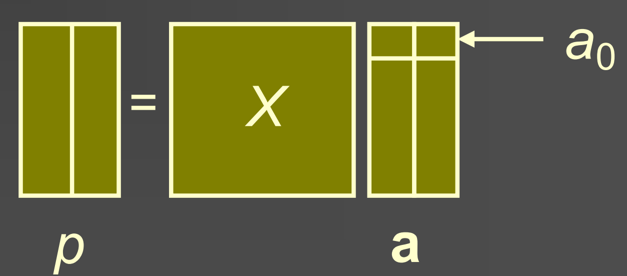
- Subdivision and repeated subdivision:

$$Sp^0 = S\sum_{i=0}^{n-1} x_i a_i = \sum_{i=0}^{n-1} \lambda_i x_i a_i$$

$$p^m = S^m p^0 = \sum_{i=0}^{n-1} \lambda_i{}^m x_i a_i$$

Eigenanalysis: convergence

- Assume that $\lambda_0 \geq \lambda_1 \geq \ldots \geq \lambda_{n-1}$, just an order …
- **Affine invariance** requires 1 to be an eigenvalue due to unit row sum
- If $\lambda_0 > 1$, then divergence. So λ_0 must be 1
- It can be shown that only one eigenvalue = 1 [Warren 95]
- If one and only one eigenvalue is 1, the limit point is a_0.
- How to compute: $a = X^{-1}p$, $X = [x_0, \ldots, x_{n-1}]$)



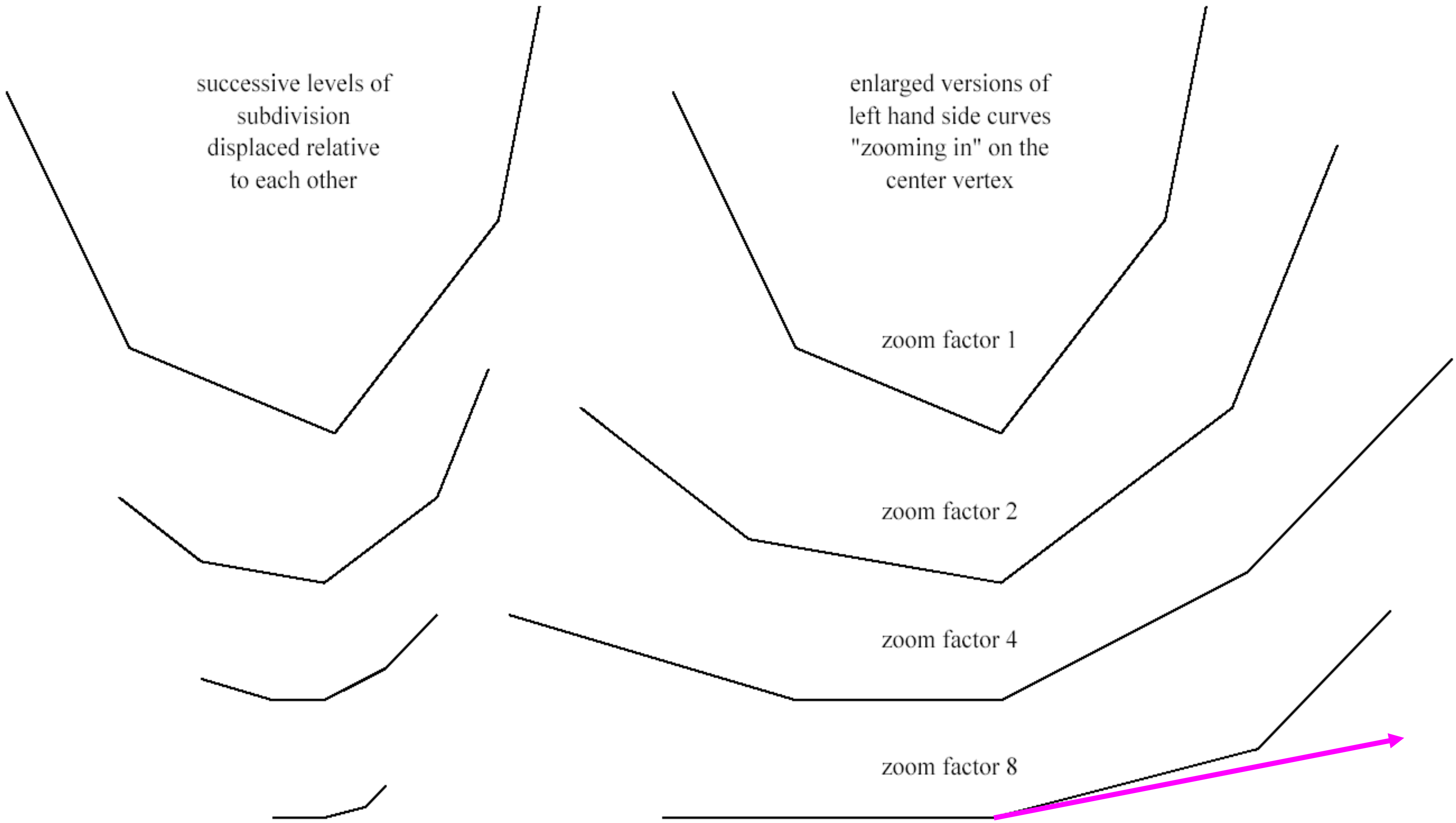
- How about tangent at limit point? – **think 2D: a_i's are 2D vectors**

Eigenanalysis: tangent

- Choose coordinate system so that a_0 is the origin

$$p^j = \sum_{i=1}^{n-1} \lambda_i^j x_i a_i \quad \text{and} \quad \frac{p^j}{\lambda_1^j} = x_1 a_1 + \sum_{i=2}^{n-1} \left(\frac{\lambda_i}{\lambda_1} \right)^j x_i a_i$$

- If λ_1, the subdominant eigenvalue, is unique, then there exists a tangent line, aligned with *vector* a_1, at p^∞
- How to compute the tangent? – Again, need to use the inverse of the eigenvector matrix X



Progressively aligned with tangent vector [pp. 44, Zorin 00]

Example: cubic B-splines

$$X = \begin{bmatrix} 1 & -1 & 1 & 1 & 0 \\ 1 & -1/2 & 2/11 & 0 & 0 \\ 1 & 0 & -1/11 & 0 & 0 \\ 1 & 1/2 & 2/11 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix} \qquad X^{-1} = \begin{bmatrix} 0 & 1/6 & 2/3 & 1/6 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 1.8333 & -3.6667 & 1.8333 & 0 \\ 1 & -3 & 3 & -1 & 0 \\ 0 & -1 & 3 & -3 & 1 \end{bmatrix}$$

Limit behavior

- $p_0^\infty = p_{-1}^0/6 + 2p_0^0/3 + p_{+1}^0/6$
- Tangent at p_0^∞ is $p_{+1}^0 - p_{-1}^0$

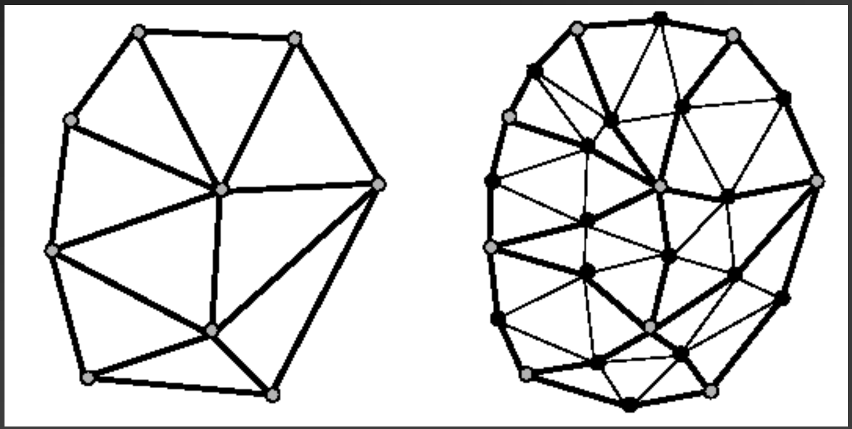
Summary of desirables

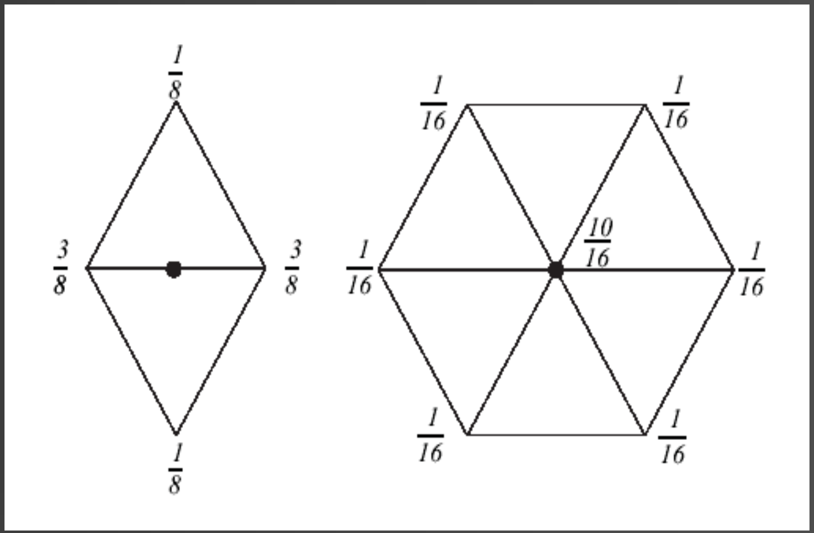
- Eigenvectors **form a basis**, i.e., complete set with linear independence
- Largest eigenvalue is 1 – affine invariance and convergence
- The subdominant eigenvalue is less than 1 – convergence
- All the other eigenvalues are less than the subdominant eigenvalue – existence of tangent, but does not say about $\mathbf{C}^1$…
- Note: most of these are **sufficient** conditions, i.e., not necessary (4-pt)

Eigen-analysis of subdivision surfaces

- Local control – same as for curves
- Affine invariance – same – need row sum of subdivision matrix to be 1
- Sufficient conditions for tangent existence a bit different
- There may be **extraordinary vertices**
  - Subdivision rules are often **different** there so as to ensure nice properties at and near these vertices
  - One fundamental solution per extraordinary case

Example: (Charles) Loop Scheme





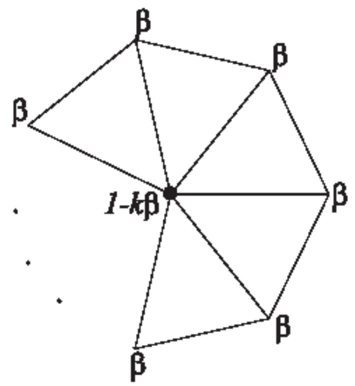


Figure 3.3: *Loop scheme: coefficients for extraordinary vertices. The choice of β is not unique; Loop [16] suggests* $\frac{1}{k}(5/8 - (\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{k})^2)$.

[pp. 48-50, Zorin 00]

Analysis

- Similar to the case for curves, however …
- There will be at least **one subdivision matrix for each valence** (can also change between levels – non-stationary)
- Notion of invariant neighborhoods still applies

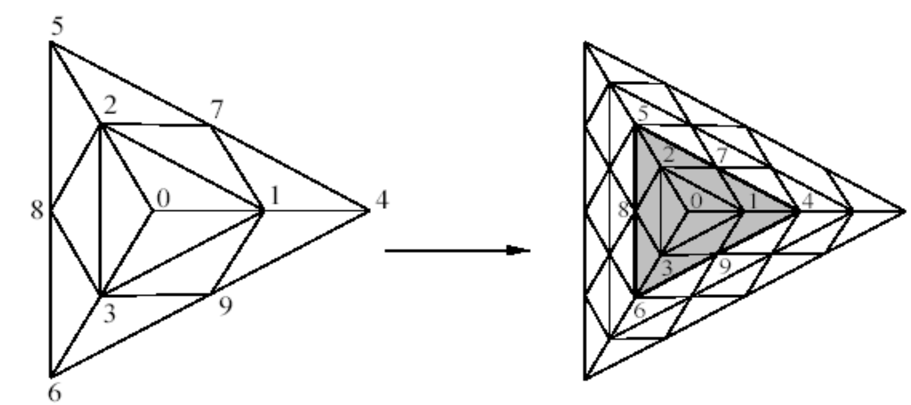


Figure 3.6: *The Loop subdivision scheme near a vertex of degree 3. Note that* $3 \times 3 + 1 = 10$ *points in two rings are required.*

Eigenanalysis

- Express control vector as linear sum of the eigenvectors of the subdivision matrix *S*, assuming linear independence

$$p = \sum_{j=0}^{n-1} x_i a_i$$

- Subdivision and repeated subdivision

$$Sp^0 = S\sum_{i=0}^{n-1} x_i a_i = \sum_{i=0}^{n-1} \lambda_i x_i a_i$$

$$p^m = S^m p^0 = \sum_{i=0}^{n-1} \lambda_i{}^m x_i a_i$$

- Note that a_i's are now 3D points

Eigenanalysis

- Again, assume that $\lambda_0 \geq \lambda_1 \geq \ldots \geq \lambda_{n-1}$
- For affine invariance and convergence, require $\lambda_0 = 1$ and be unique
- For existence of **tangent plane**, note that

$$\frac{p^j}{\lambda^j} = x_1 a_1 + x_2 a_2 + \left(\frac{\lambda_3}{\lambda}\right)^j x_3 a_3 + \ldots$$

if origin is at $a_0 = \mathbf{0}$, and

$$\lambda = \lambda_1 = \lambda_2 > \lambda_3$$

- The tangent plane will be spanned by vectors a_1 and a_2

Smoothness of subdivision surfaces (aside)

- Two notions: **C^1-continuous** vs. **tangent plane continuous**
  - Technical definition of $\mathbf{C}^1$ continuity of surface [pp. 56, Zorin 00]
  - Tangent-plane continuity (weaker) requires the limit of normals exist
  - Tangent-plane continuity + one-to-one projection between surface and tangent plane $\Rightarrow$ $\mathbf{C}^1$ continuity
- Essential/pioneering work for subdivision surfaces near extraordinary vertices:
  - Reif 's sufficient conditions for subdivision surfaces to be $\mathbf{C}^1$ — [Section 3.5, Zorin 00] as further reading