Computer Graphics

- Spline and Subdivision Surfaces -

Hendrik Lensch

Overview

- **Last Time**
  – Image-Based Rendering

- **Today**
  – Parametric Curves
  – Lagrange Interpolation
  – Hermite Splines
  – Bezier Splines
  – DeCasteljau Algorithm
  – Parameterization

B-Splines

- **Goal**
  - Spline curve with local control and high continuity
- **Given**
  - Degree: n
  - Control points: $P_0, \ldots, P_m$ (Control polygon, $m \geq n+1$)
  - Knots: $t_0, \ldots, t_{m+n+1}$ (Knot vector, weakly monotonic)
  - The knot vector defines the parametric locations where segments join
- **B-Spline Curve**

$$\underline{P}(t) = \sum_{i=0}^{m} N_i^n(t) \underline{P}_i$$

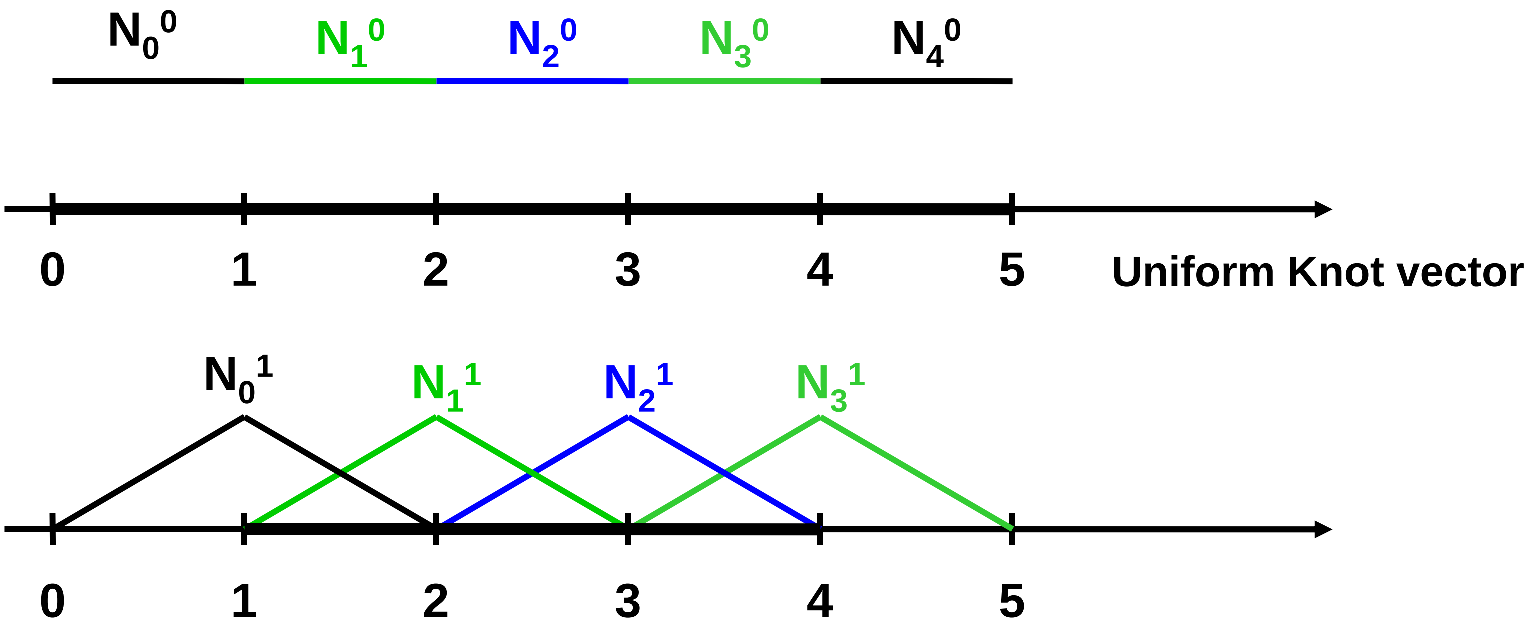
  - Continuity:
    - C_{n-1} at simple knots
    - C_{n-k} at knot with multiplicity k

B-Spline Basis Functions

- **Recursive Definition**

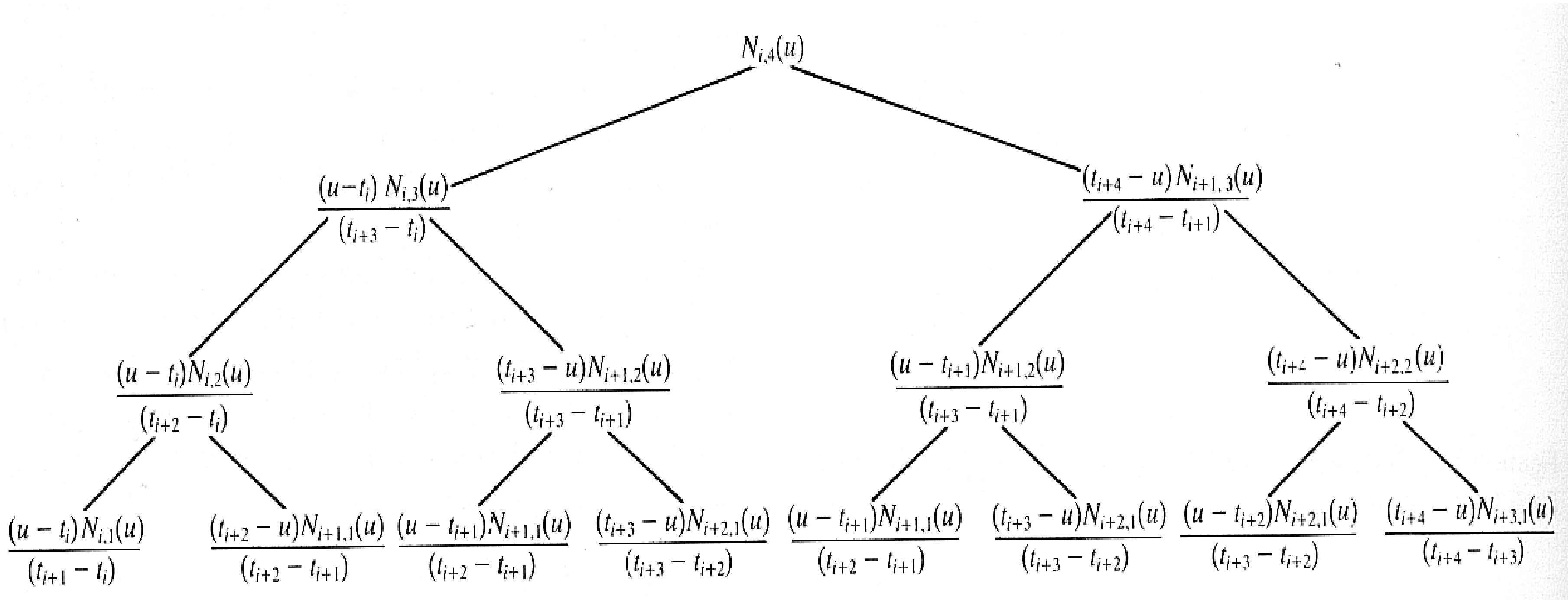
$$N_i^0(t) = \begin{cases} 1 & \text{if } t_i < t < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_i^n(t) = \frac{t - t_i}{t_{i+n} - t_i} N_i^{n-1}(t) - \frac{t - t_{i+n+1}}{t_{i+n+1} - t_{i+1}} N_{i+1}^{n-1}(t)$$



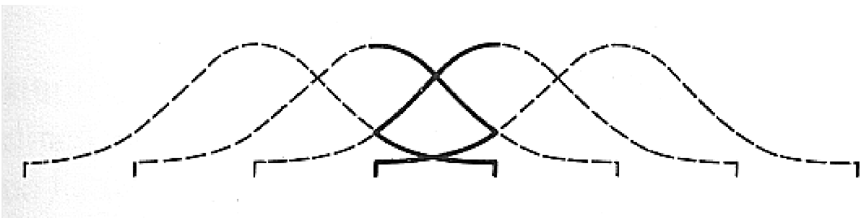
B-Spline Basis Functions

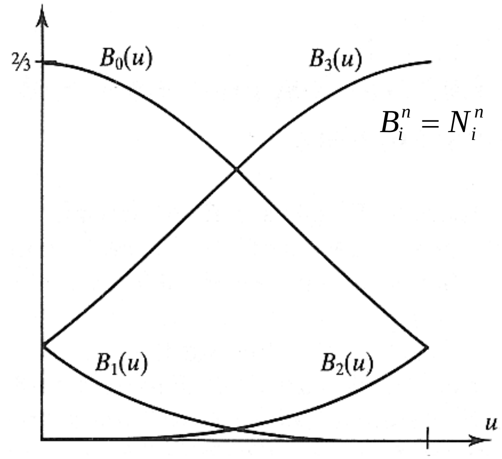
- **Recursive Definition**
  - Degree increases in every step
  - Support increases by one knot interval



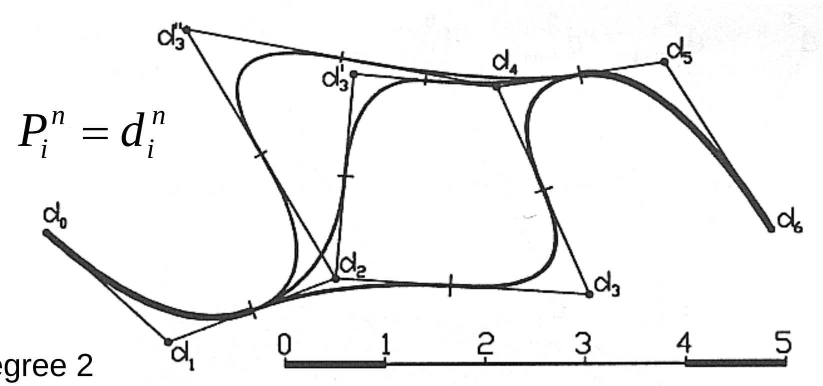
B-Spline Basis Functions

- **Uniform Knot Vector**
  - All knots at integer locations
    - UBS: Uniform B-Spline
  - Example: cubic B-Splines





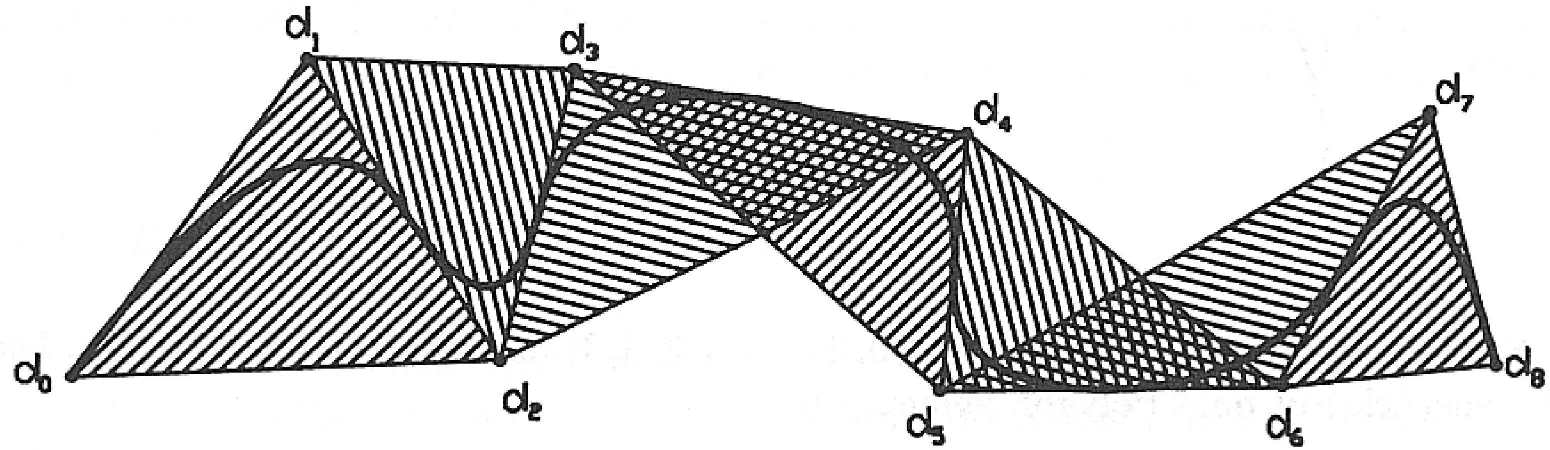
- **Local Support = Localized Changes**
  - Basis functions affect only (n+1) Spline segments
  - Changes are localized



Degree 2

B-Spline Basis Functions

- **Convex Hull Property**
  – Spline segment lies in convex Hull of (n+1) control points

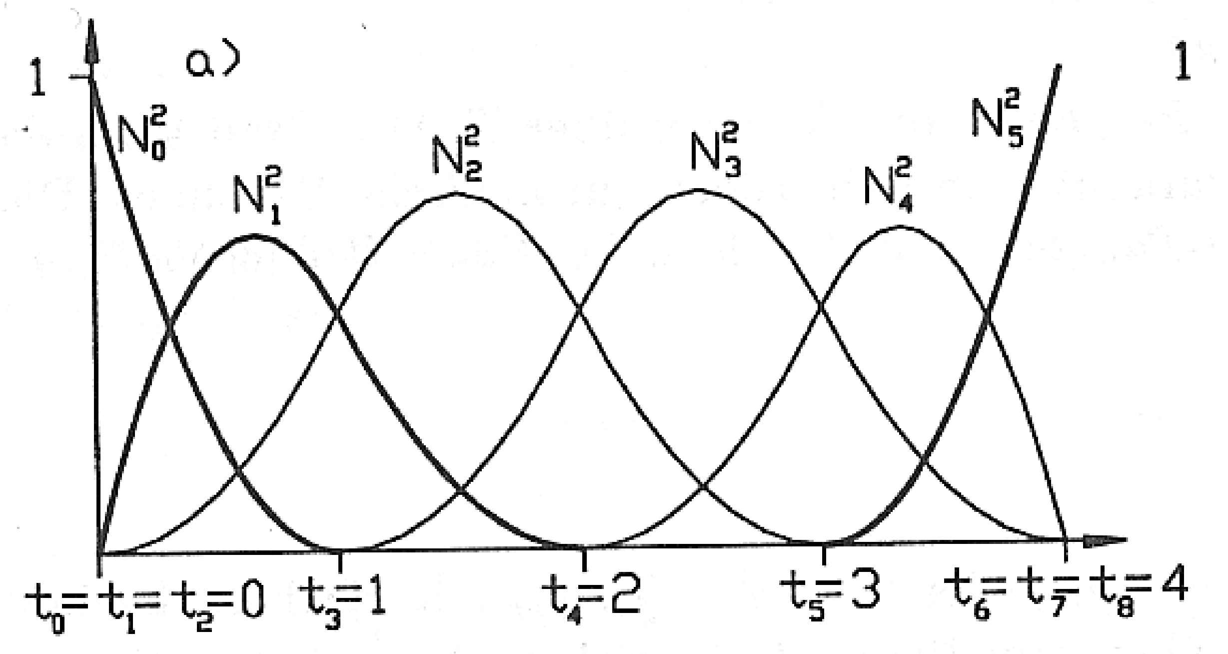


Degree 2

  – (n+1) control points lie on a straight line ➔ curve touches this line
  – n control points coincide ➔ curve interpolates this point and is tangential to the control polygon (e.g. beginning and end)

Normalized Basis Functions

- **Basis Functions on an Interval**
  - Partition of unity: $\sum_i N_i^n(t) = 1$
  - Knots at beginning and end with multiplicity
  - Interpolation of end points and tangents there
  - Conversion to Bézier segments via knot insertion



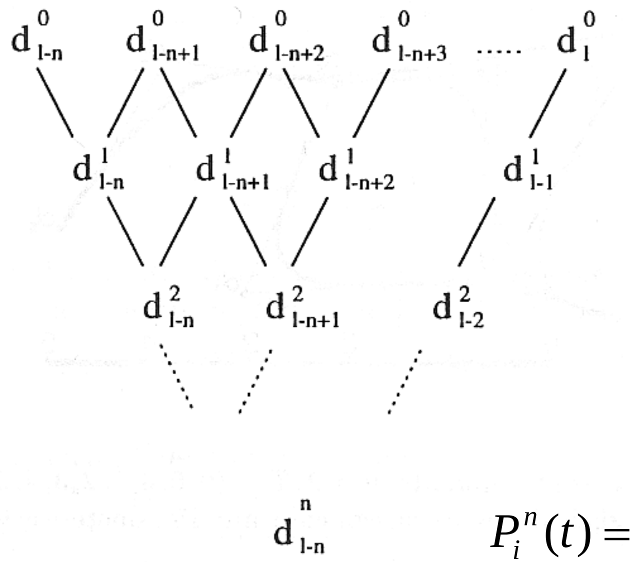
deBoor-Algorithm

- **Evaluating the B-Spline**
- **Recursive Definition of Control Points**
  - Evaluation at t: $t_l < t < t_{l+1}$: $i \in \{l-n, \ldots, l\}$
    - Due to local support only affected by (n+1) control points

$$\underline{P}_i^r(t) = \left(1 - \frac{t - t_{i+r}}{t_{i+n+1} - t_{i+r}}\right)\underline{P}_i^{r-1}(t) + \frac{t - t_{i+r}}{t_{i+n+1} - t_{i+r}}\underline{P}_{i+1}^{r-1}(t)$$

$$\underline{P}_i^0(t) = \underline{P}_i$$

- **Properties**
  - Affine invariance
  - Stable numerical evaluation
    - All coefficients > 0



$$P_i^n(t) = d_i^n$$

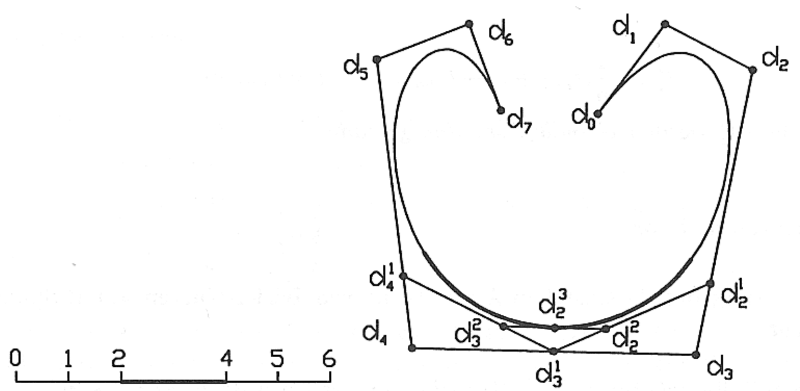
Knot Insertion

- **Algorithm similar to deBoor**
  - Given a new knot t
    - $t_l \le t < t_{l+1}$: $i \in \{l-n, ..., l\}$
  - $T^* = T \cup \{t\}$
  - New representation of the same curve over T*

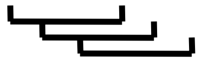
$$\underline{P}^*(t) = \sum_{i=0}^{m+1} N_i^n(t)\underline{P}_i^*$$

$$P_i^* = (1-a_i)P_{i-1} + a_i P_i$$

$$a_i = \begin{cases} 1 & i \le l-n \\ \dfrac{t-t_i}{t_{i+n}-t_i} & l-n+1 \le i \le l \\ 0 & i \ge l+1 \end{cases}$$



Consecutive insertion of three knots at t=3 into a cubic B-Spline
First and last knot have multiplicity n
T=(0,0,0,0,1,2,4,5,6,6,6,6), l=5



- **Applications**
  - Refinement of curve, display

Conversion to Bézier Spline

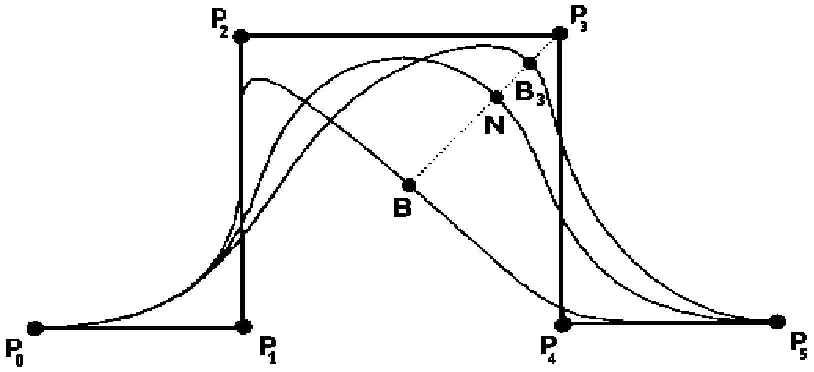
- **B-Spline to Bézier Representation**
  - Remember:
    - Curve interpolates point and is tangential at knots of multiplicity n
  - In more detail: If two consecutive knots have multiplicity n
    - The corresponding spline segment is in Bézier from
    - The (n+1) corresponding control polygon form the Bézier control points

NURBS

- **Non-uniform Rational B-Splines**
  - Homogeneous control points: now with weight w_i
    - $\underline{P}_i{}' = (w_i\, x_i,\ w_i\, y_i,\ w_i\, z_i,\ w_i) = w_i \underline{P}_i$

$$\underline{P}'(t) = \sum_{i=0}^{m} N_i^n(t)\underline{P}_i'$$

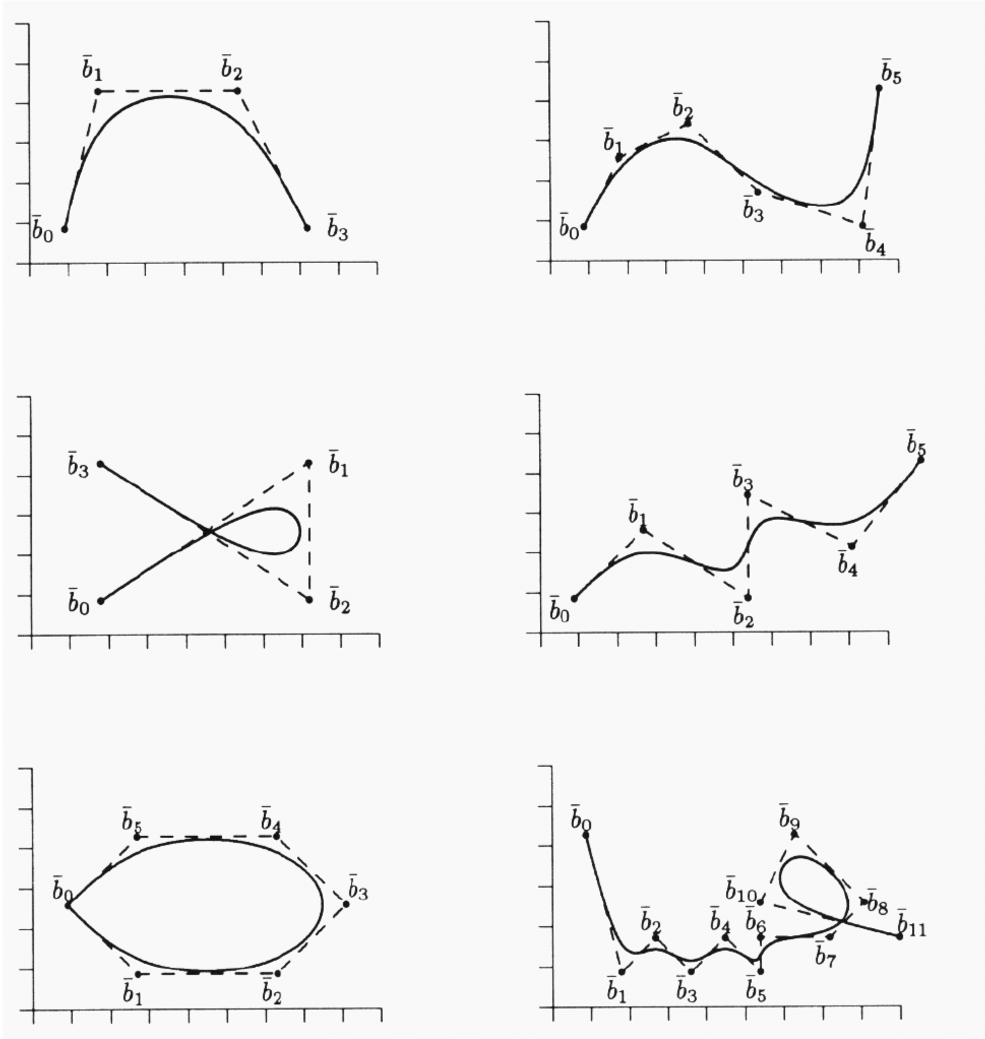
$$\underline{P} = \frac{\sum_{i=0}^{m} N_i^n(t)\underline{P}_i w_i}{\sum_{i=0}^{m} N_i^n(t) w_i} = \sum_{i=0}^{m} R_i^n(t)\underline{P}_i w_i, \quad \text{mit} \quad R_i^n(t) = \frac{N_i^n(t) w_i}{\sum_{i=0}^{m} N_i^n(t) w_i}$$



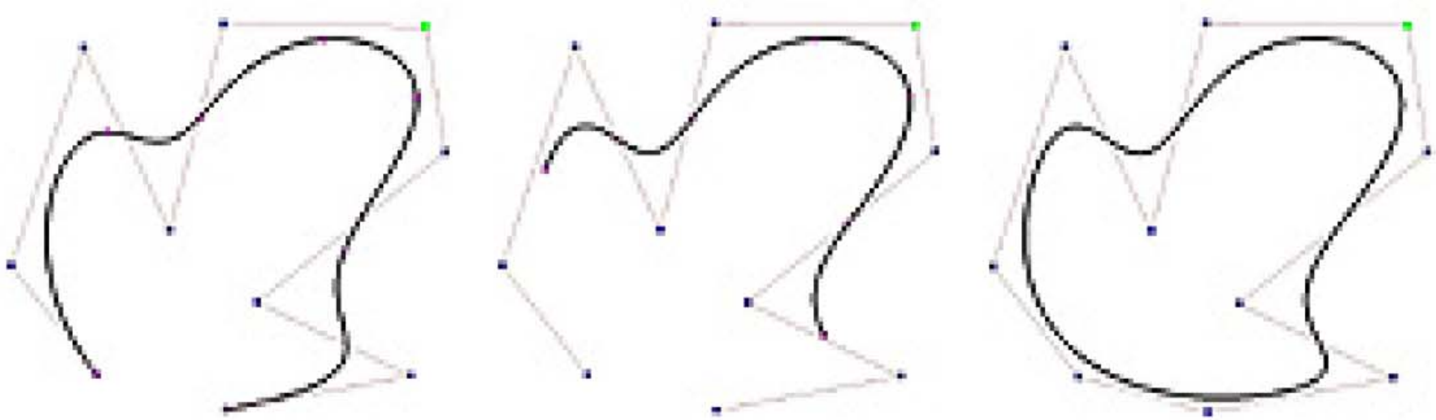
NURBS

- **Properties**
  - Piecewise rational functions
  - Weights
    - High (relative) weight attract curve towards the point
    - Low weights repel curve from a point
    - Negative weights should be avoided (may introduce singularity)
  - Invariant under projective transformations
  - Variation-Diminishing-Property (in functional setting)
    - Curve cuts a straight line no more than the control polygon does

Examples: Cubic B-Splines



Knots and Points



(a) Clamped (b) Open (c) Closed

multiplicity = n at beginning and end

[00012345678999]

strictly monotonous knot vector

[0123456789]

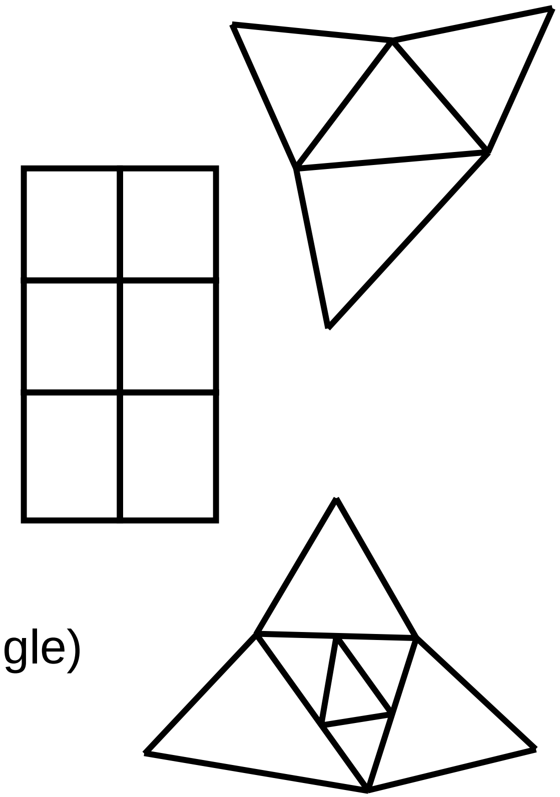
knots or points replicated

$[P_0, P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_0, P_1, P_2]$

Spline Surfaces

Parametric Surfaces

- **Same Idea as with Curves**
  - $\underline{P}: R^2 \rightarrow R^3$
  - $\underline{P}(u,v) = (x(u,v), y(u,v), z(u,v))^T \in R^3$ (also $P(R^4)$)
- **Different Approaches**
  - Triangular Splines
    - Single polynomial in (u,v) via barycentric coordinates with respect to a reference triangle (e.g. B-Patches)
  - Tensor Product Surfaces
    - Separation into polynomials in u and in v
  - Subdivision Surfaces
    - Start with a triangular mesh in R^3
    - Subdivide mesh by inserting new vertices
      - Depending on local neighborhood
    - Only piecewise parameterization (in each triangle)



Tensor Product Surfaces

- **Idea**
  - Create a “curve of curves"
- **Simplest case: Bilinear Patch**
  - Two lines in space

$$\underline{P}^1(v) = (1-v)\underline{P}_{00} + v\underline{P}_{10}$$

$$\underline{P}^2(v) = (1-v)\underline{P}_{01} + v\underline{P}_{11}$$

  - Connected by lines

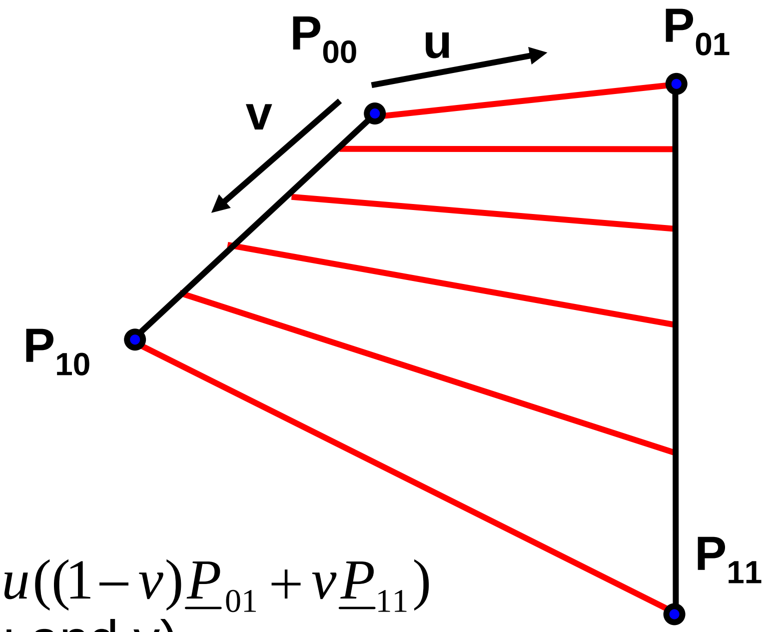
$$\underline{P}(u,v) = (1-u)\underline{P}^1(v) + u\underline{P}^2(v) =$$

$$(1-u)((1-v)\underline{P}_{00} + v\underline{P}_{10}) + u((1-v)\underline{P}_{01} + v\underline{P}_{11})$$

  - Bézier representation (symmetric in u and v)

$$\underline{P}(u,v) = \sum_{i,j=0}^{1} B_i^1(u) B_j^1(v) \underline{P}_{ij}$$

  - Control mesh P_{ij}



Tensor Product Surfaces

- **General Case**
  - Arbitrary basis functions in u and v
    - Tensor Product of the function space in u and v
  - Commonly same basis functions and same degree in u and v

$$\underline{P}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} B_i^m(u) B_j^n(v) \underline{P}_{ij}$$

- **Interpretation**
  - Curve defined by curves

$$\underline{P}(u,v) = \sum_{i=0}^{m} B_i^{'}(u) \underbrace{\sum_{j=0}^{n} B_j(v) \underline{P}_{ij}}_{P_i^{'}(v)}$$

  - Symmetric in u and v

Matrix Representation

- **Similar to Curves**
  - Geometry now in a „tensor“ (m x n x 3)

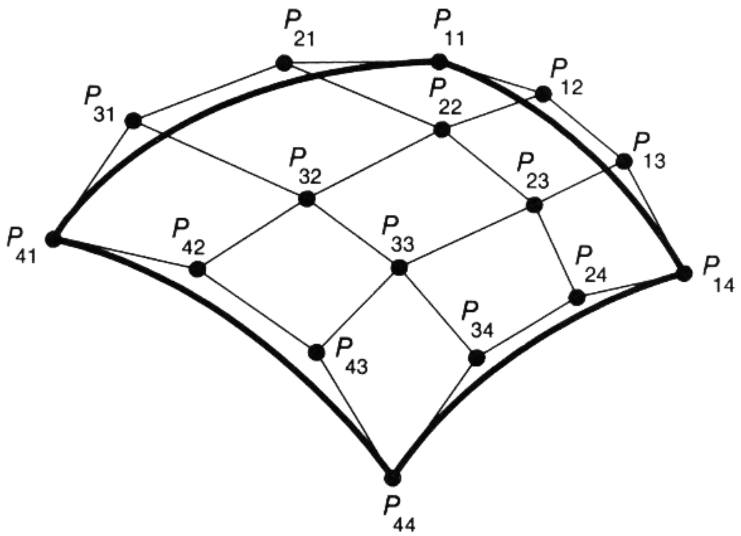
$$\underline{P}(u,v) = U\mathbf{G}_{monom}V^T = \begin{pmatrix} u^m & \cdots & u & 1 \end{pmatrix} \begin{pmatrix} G_{nn} & \cdots & G_{n0} \\ \vdots & \ddots & \vdots \\ G_{0n} & \cdots & G_{00} \end{pmatrix} \begin{pmatrix} v^n \\ \vdots \\ v \\ 1 \end{pmatrix} =$$

$$U\mathbf{B}'_U \mathbf{G}_{UV} \mathbf{B}_V^T V^T$$

  - Degree
    - u: m
    - v: n
    - Along the diagonal (u=v): m+n
      - Not nice → „Triangular Splines“

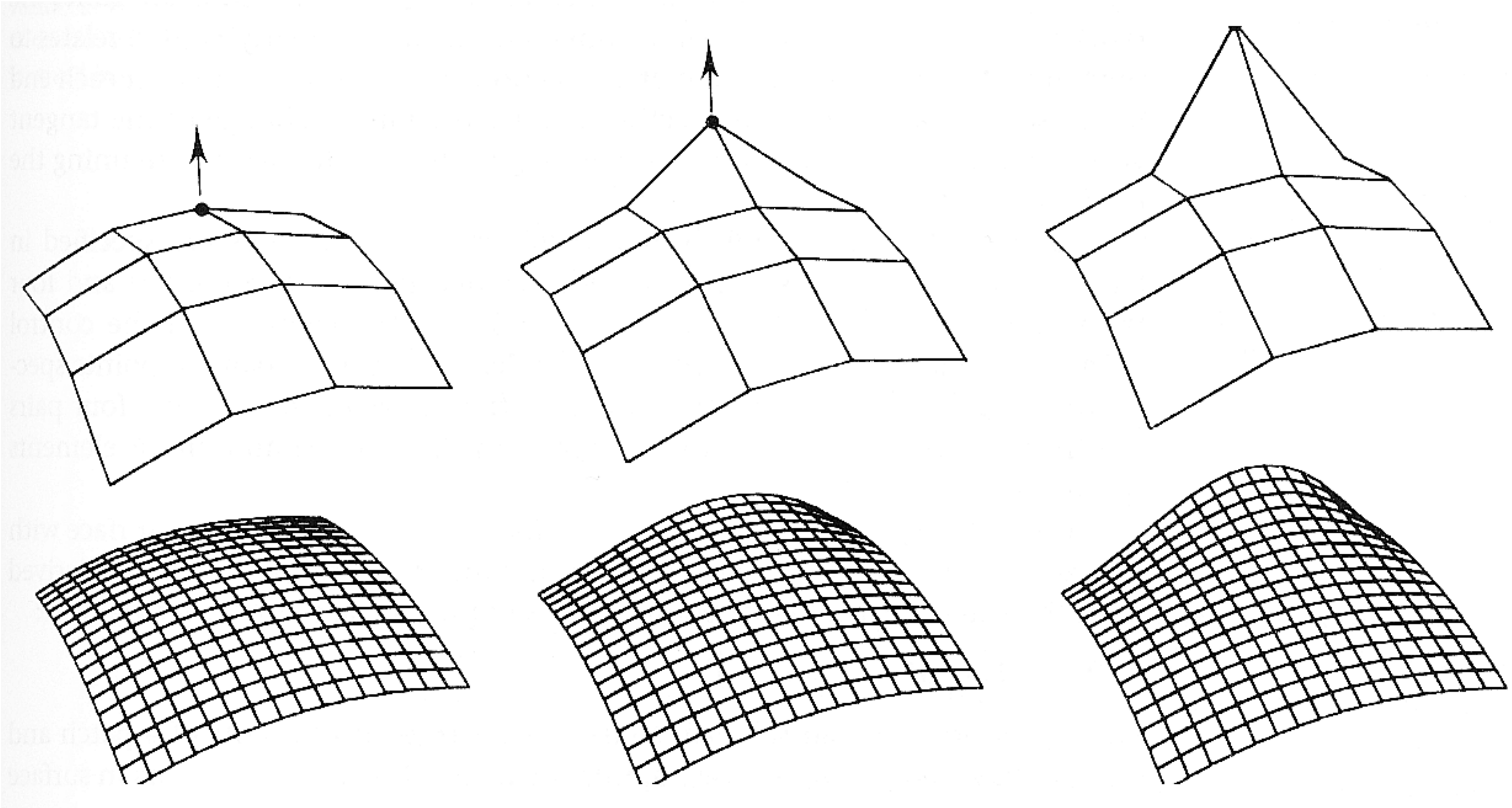
Tensor Product Surfaces

- **Properties Derived Directly From Curves**
- **Bézier Surface:**
  - Surface interpolates corner vertices of mesh
  - Vertices at edges of mesh define boundary curves
  - Convex hull property holds
  - Simple computation of derivatives
  - Direct neighbors of corners vertices define tangent plane
- **Similar for Other Basis Functions**



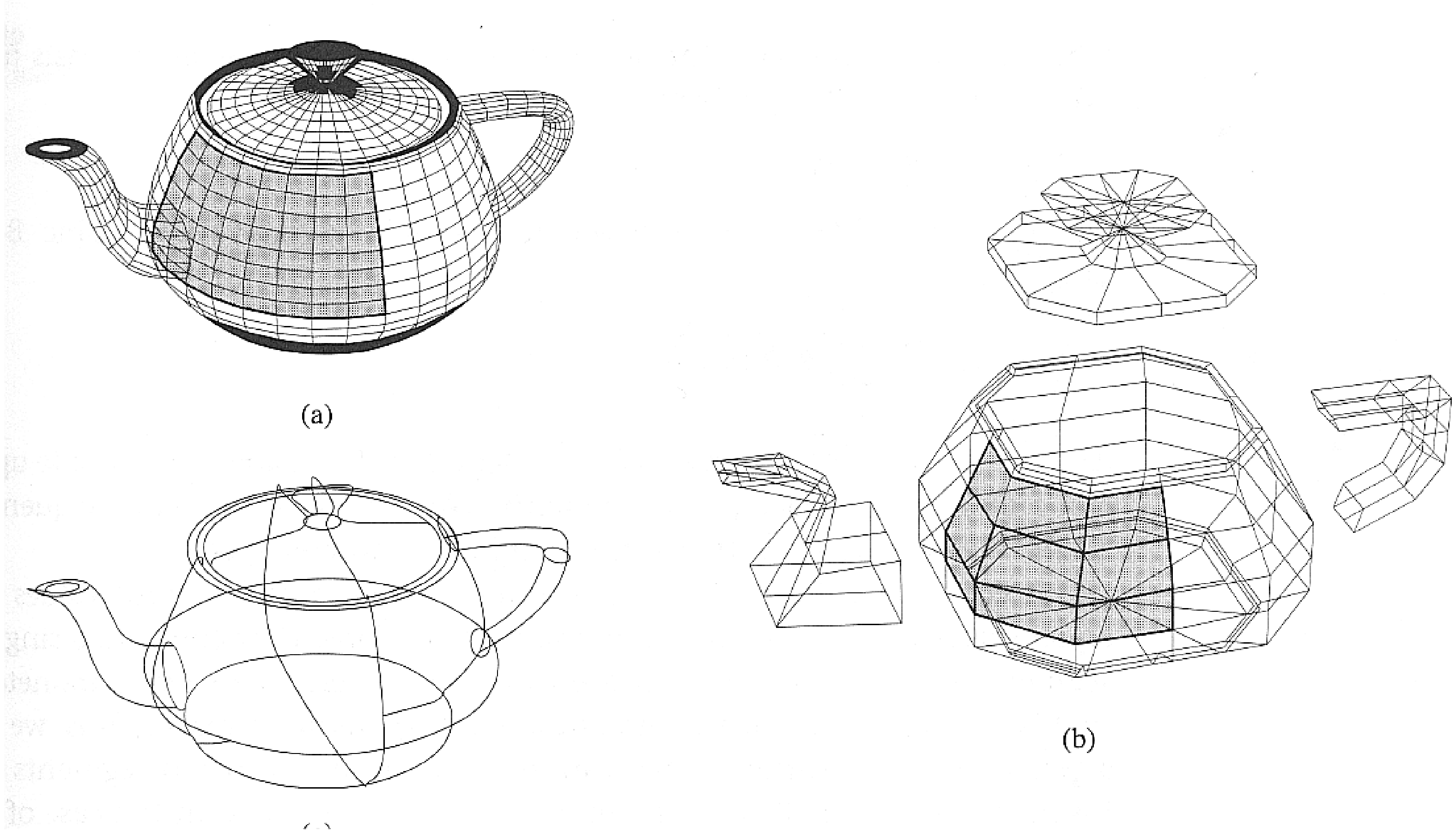
Tensor Product Surfaces

- **Modifying a Bézier Surface**



Tensor Product Surfaces

- **Representing the Utah Teapot as a set continuous Bézier patches**
  - http://www.holmes3d.net/graphics/teapot/

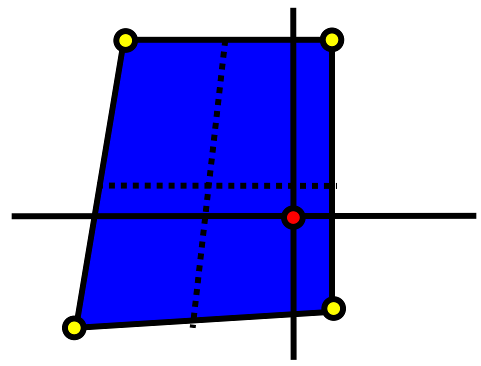


Operations on Surfaces

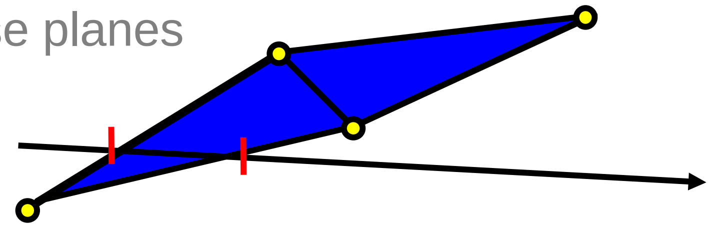
- **deCausteljau/deBoor Algorithm**
  - Once for u in each column
  - Once for v in the resulting row
  - Due to symmetry also in other order
- **Similarly we can derive the related algorithms**
  - Subdivision
  - Extrapolation
  - Display
  - ...

Ray Tracing of Spline Surfaces

- **Several approaches**
  - Tessellate into many triangles (using deCasteljau or deBoor)
    - Often the fasted method
    - May need enormous amounts of memory
  - Recursive subdivision
    - Simply subdivide patch recursively
    - Delete parts that do not intersect ray (Pruning)
    - Fixed depth ensures crack-free surface

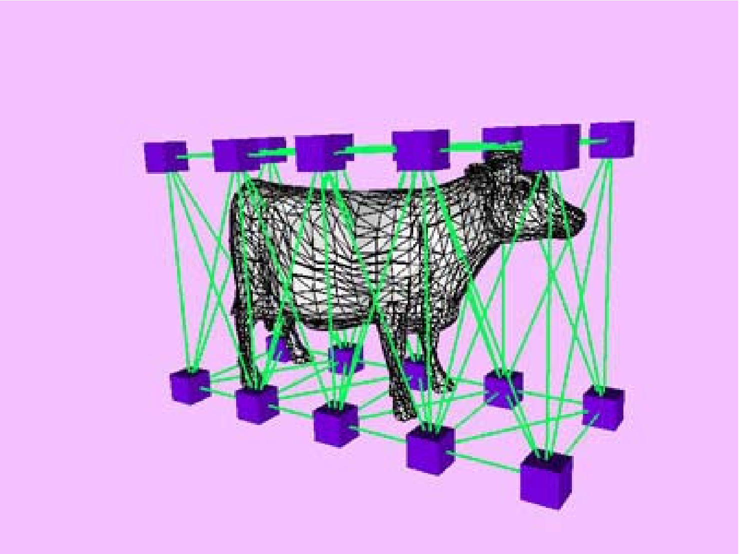


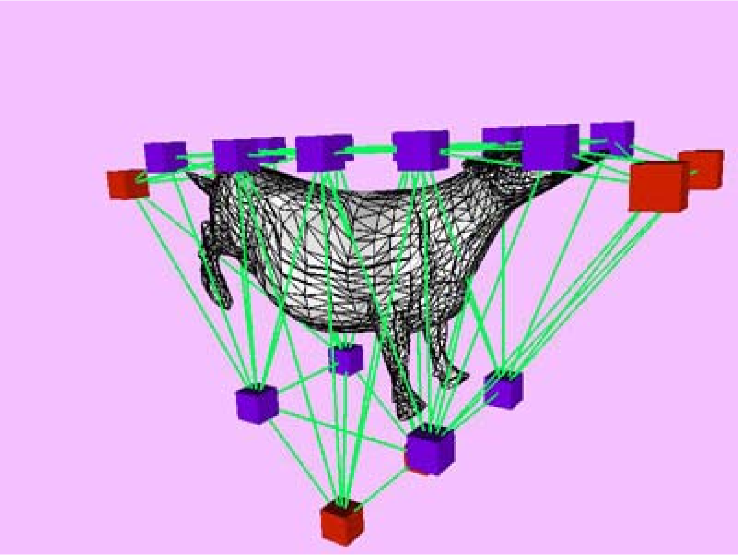
  - Bézier Clipping [Sederberg et al.]
    - Find two orthogonal planes that intersect in the ray
    - Project the surface control points into these planes
    - Intersection must have distance zero
      - ➔ Root finding
      - ➔ Can eliminate parts of the surface where convex hull does not intersect ray
    - Must deal with many special cases – rather slow



Higher Dimensions

- **Volumes**
  - Spline: $R^3 \rightarrow R$
    - Volume density
    - Rarely used
  - Spline: $R^3 \rightarrow R^3$
    - Modifications of points in 3D
    - Displacement mapping
    - Free Form Deformations (FFD)





FFD

Subdivision Surfaces

Modeling

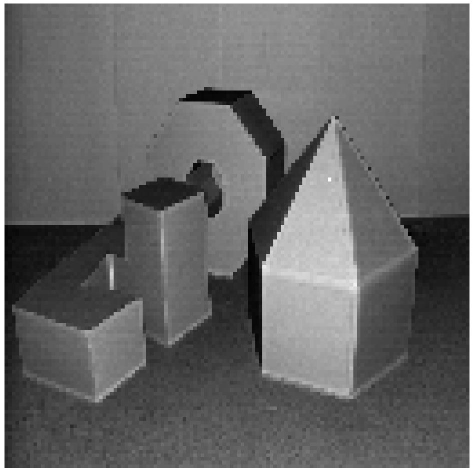
- **How do we ...**
  - Represent 3D objects in a computer?
  - Construct such representations quickly and/or automatically with a computer?
  - Manipulate 3D objects with a computer?
- **3D Representations provide the foundations for**
  - Computer Graphics
  - Computer-Aided Geometric Design
  - Visualization
  - Robotics, …
- **Different methods for different object representations**

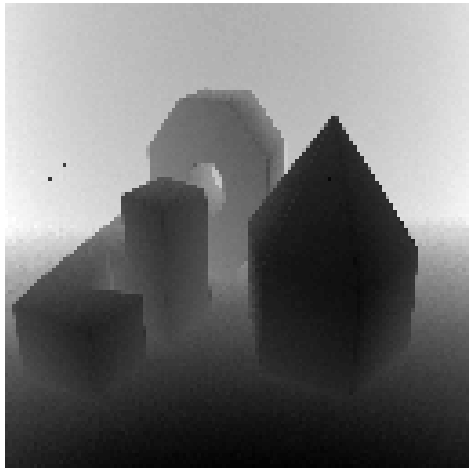
3D Object Representations

- **Raw data**
  - Range image
  - Point cloud
  - Polygon soup
- **Surfaces**
  - Mesh
  - Subdivision
  - Parametric
  - Implicit
- **Solids**
  - Voxels
  - BSP tree
  - CSG

Range Image

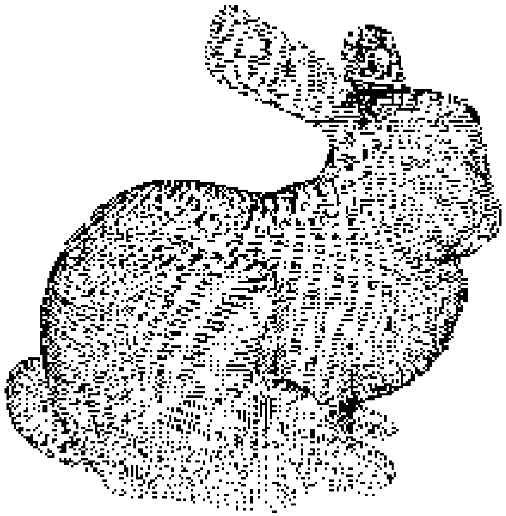
- **Range image**
  - Acquired from range scanner
    - E.g. laser range scanner, structured light, phase shift approach
  - Structured point cloud
    - Grid of depth values with calibrated camera
    - 2-1/2D: 2D plus depth





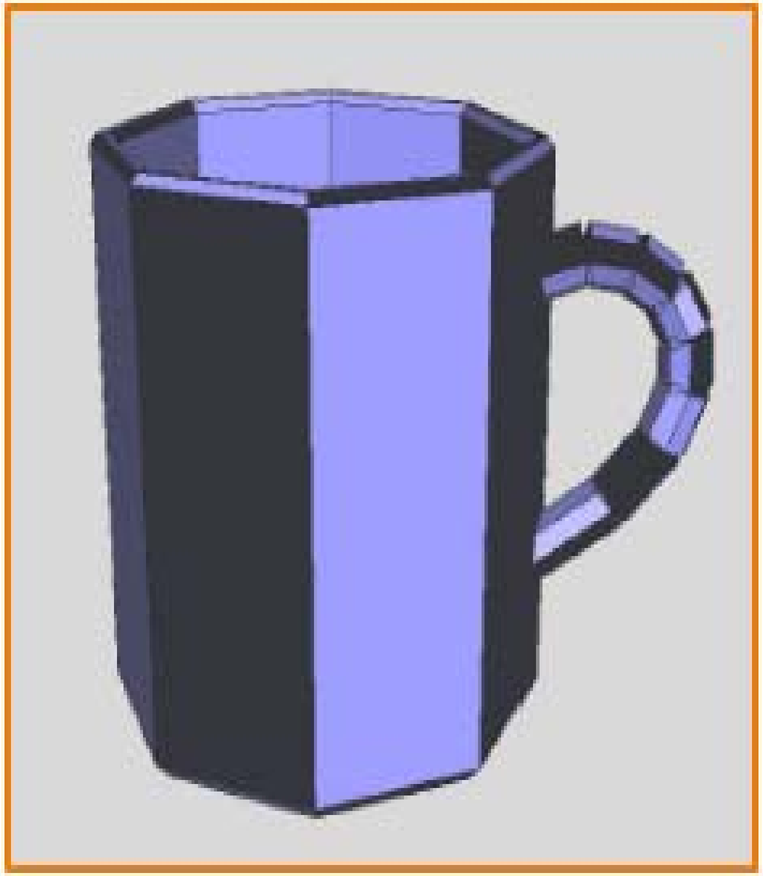
Point Cloud

- **Unstructured set of 3D point samples**
  - Often constructed from many range images



Polygon Soup

- **Unstructured set of polygons**

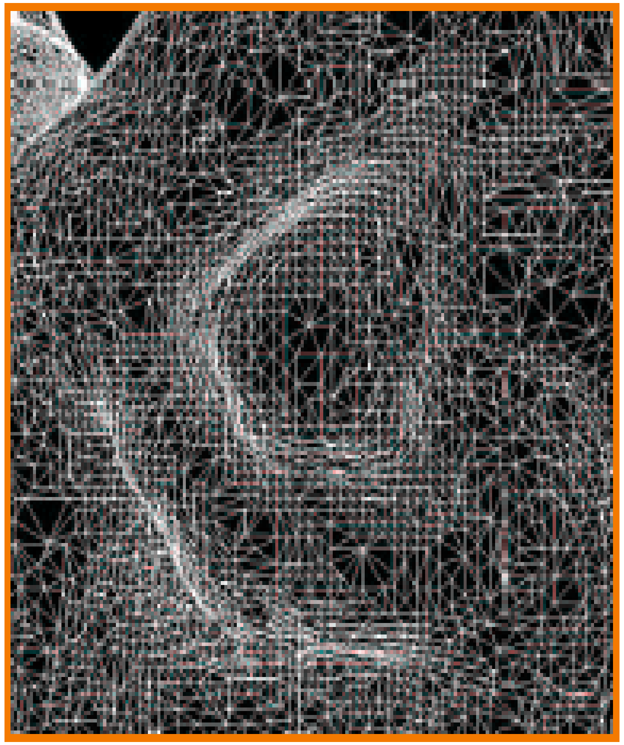


3D Object Representations

- **Raw data**
  - Point cloud
  - Range image
  - Polygon soup
- **Surfaces**
  - Mesh
  - Subdivision
  - Parametric
  - Implicit
- **Solids**
  - Voxels
  - BSP tree
  - CSG

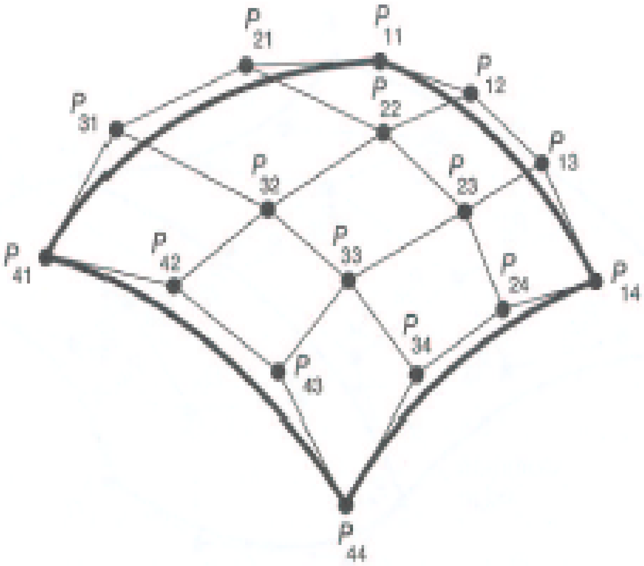
Mesh

- **Connected set of polygons (usually triangles)**

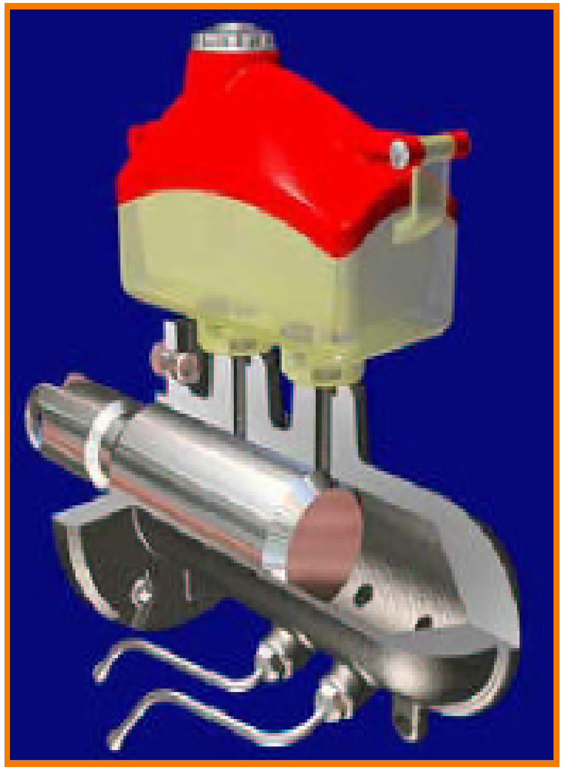


Parametric Surface

- **Tensor product spline patches**
  - Careful constraints to maintain continuity

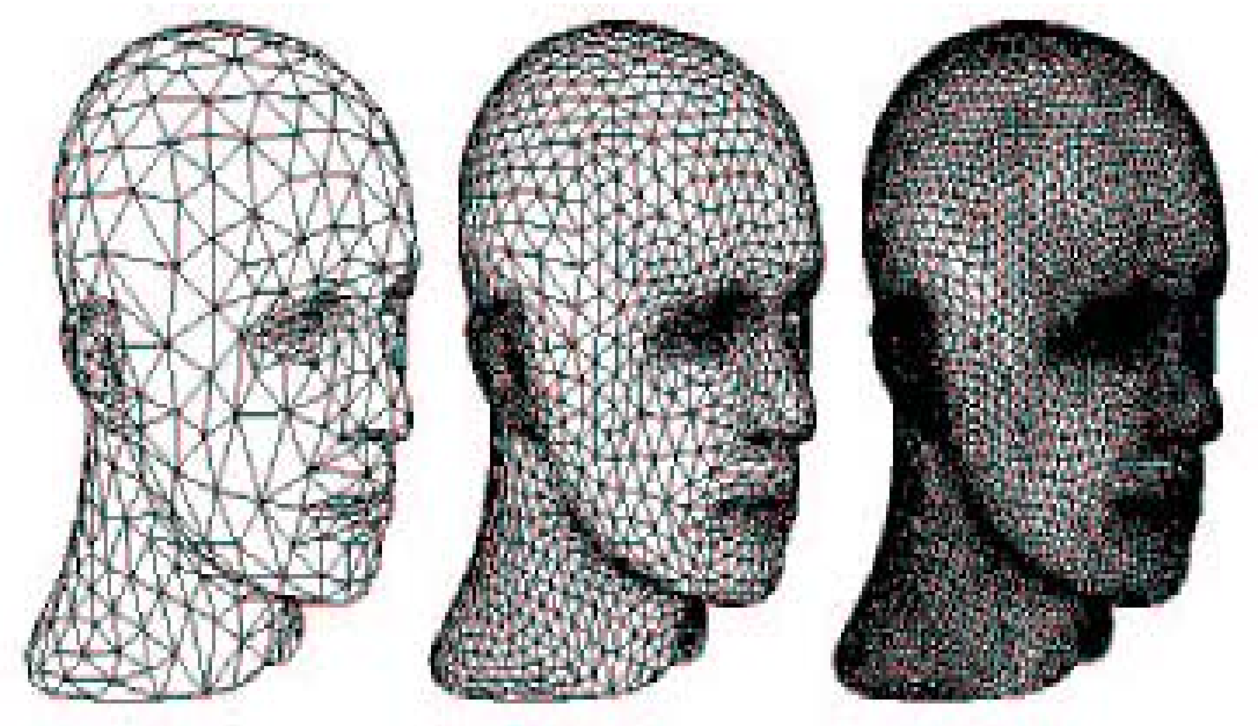


FvDFH Figure 11.44



Subdivision Surface

- **Coarse mesh & subdivision rule**
  - Define smooth surface as limit of sequence of refinements



Implicit Surface

- **Points satisfying: F(x,y,z) = 0**

Polygonal Model

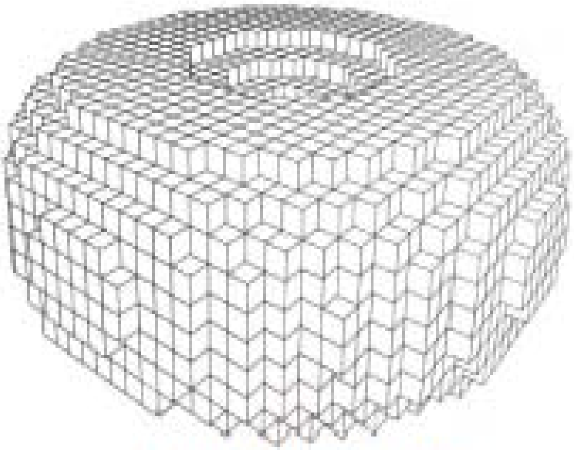
Implicit Model

3D Object Representations

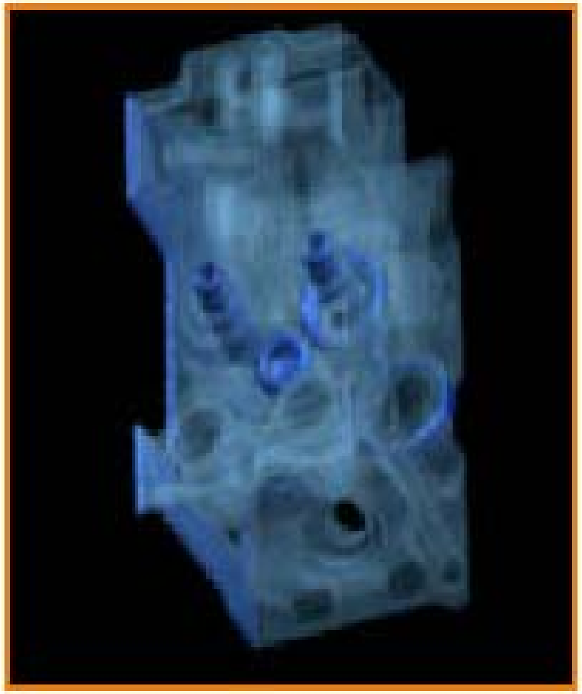
- **Raw data**
  - Point cloud
  - Range image
  - Polygon soup
- **Surfaces**
  - Mesh
  - Subdivision
  - Parametric
  - Implicit
- **Solids**
  - Voxels
  - BSP tree
  - CSG

Voxels

- **Uniform grid of volumetric samples**
  - Acquired from CAT, MRI, etc.



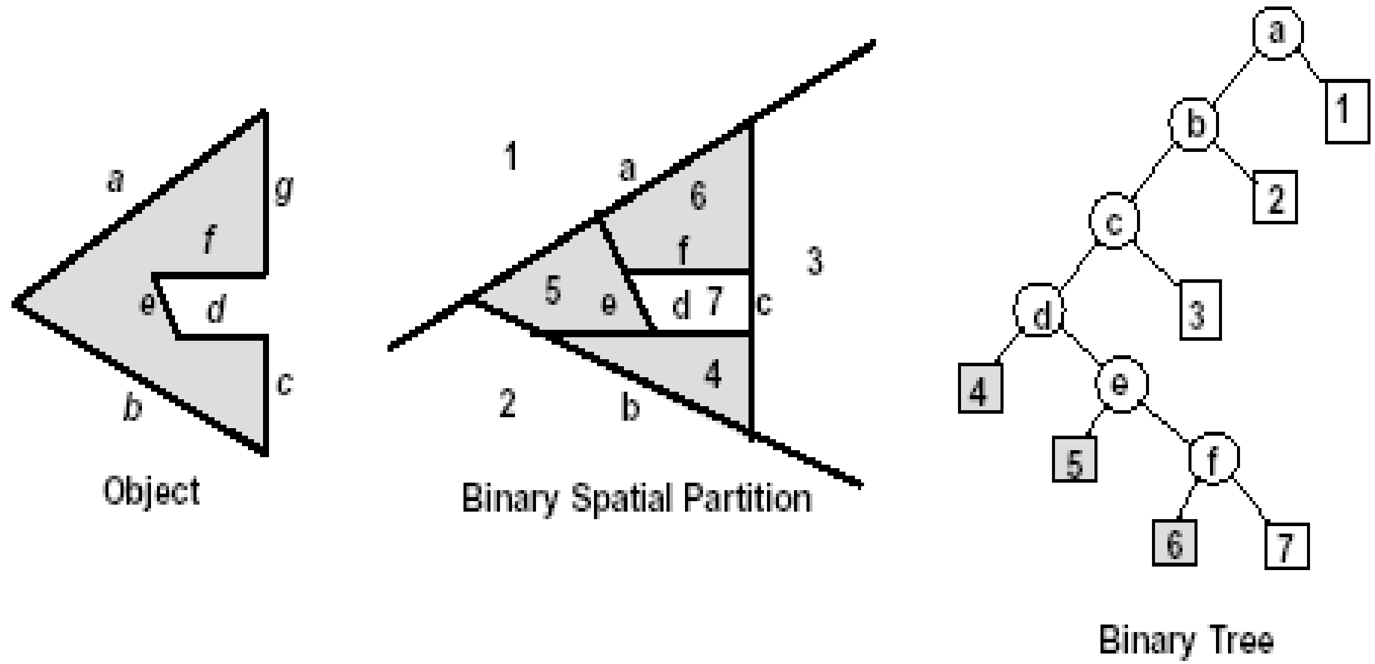
FvDFH Figure 12.20



Stanford Graphics Laboratory

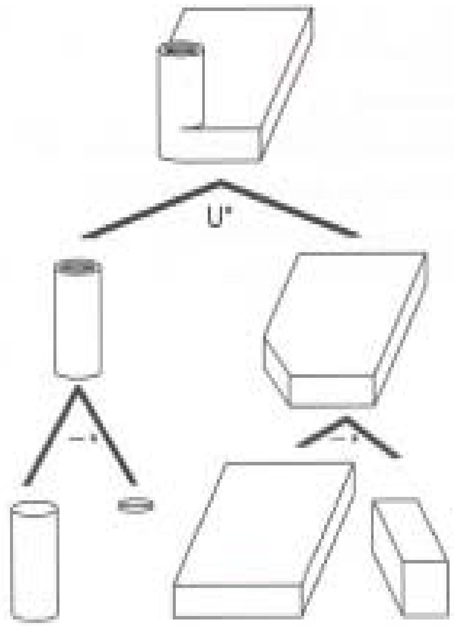
BSP Tree

- **Binary space partition with solid cells labeled**
  - Constructed from polygonal representations

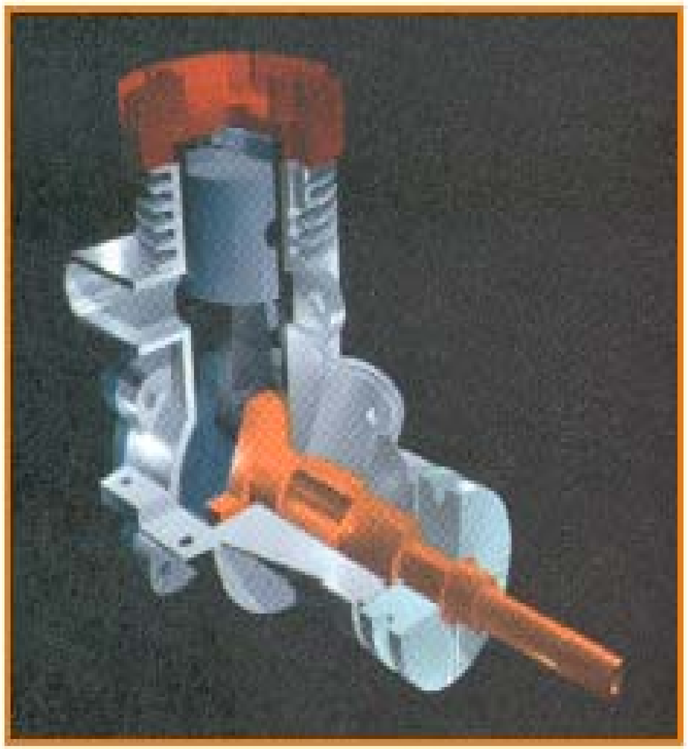


CSG – Constructive Solid Geometry

- **Hierarchy of boolean set operations (union, difference, intersect) applied to simple shapes**



FvDFH Figure 12.27



H&B Figure 9.9

Motivation

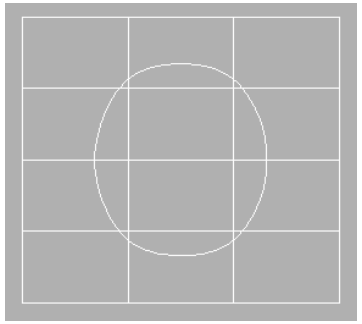
- **Splines**
  - Traditionally spline patches (NURBS) have been used in production for character animation.
- **Difficult to stitch together**
  - Maintaining continuity is hard
- **Difficult to model objects with complex topology**

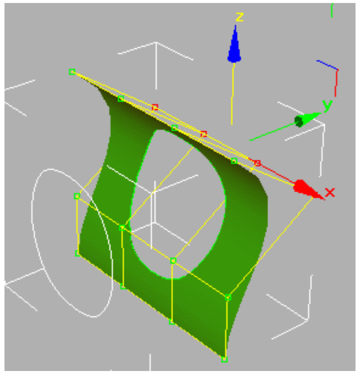
Subdivision in Character Animation
Tony Derose, Michael Kass, Tien Troung
(SIGGRAPH ’98)

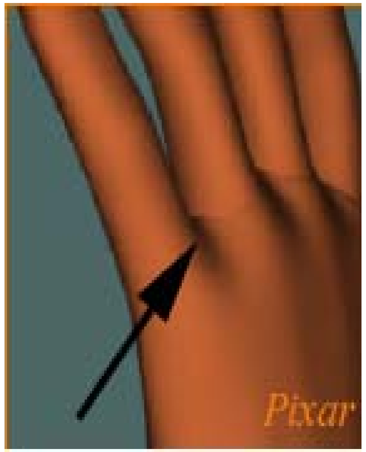
(Geri’s Game, Pixar 1998)

Motivation

- **Splines (Bézier, NURBS, …)**
  - Easy and commonly used in CAD systems
  - Most surfaces are not made of quadrilateral patches
    - Need to trim surface: Cut of parts
  - Trimming NURBS is expensive and often has numerical errors
  - Very difficult to stich together separate surfaces
  - Very hard to hide seams

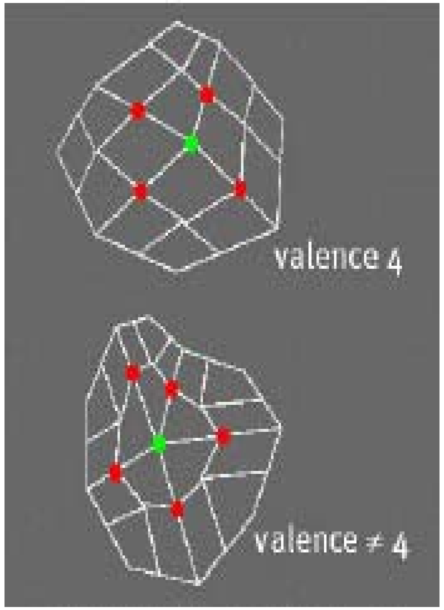


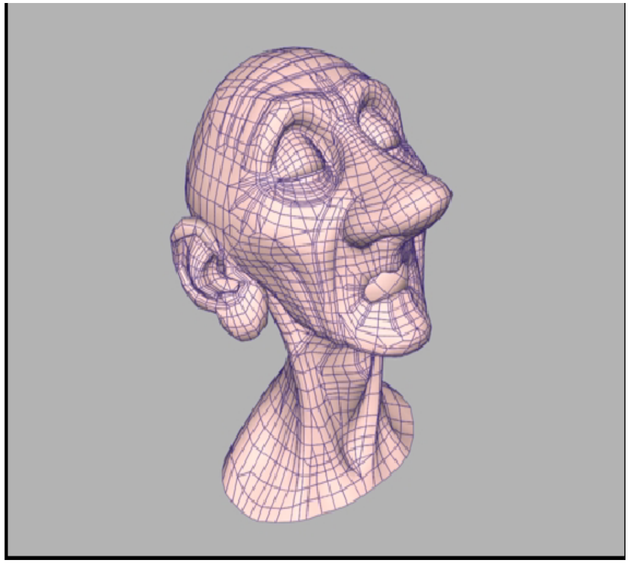




Why Subdivision Surfaces?

- **Subdivision methods have a series of interesting properties:**
  - Applicable to meshes of arbitrary topology (non-manifold meshes).
  - No trimming needed
  - Scalability, level-of-detail.
  - Numerical stability.
  - Simple implementation.
  - Compact support.
  - Affine invariance.
  - Continuity
  - Still less tools in CAD systems (but improving quickly)





Types of Subdivision

- **Interpolating Schemes**
  - Limit Surfaces/Curve will pass through original set of data points.
- **Approximating Schemes**
  - Limit Surface will not necessarily pass through the original set of data points.

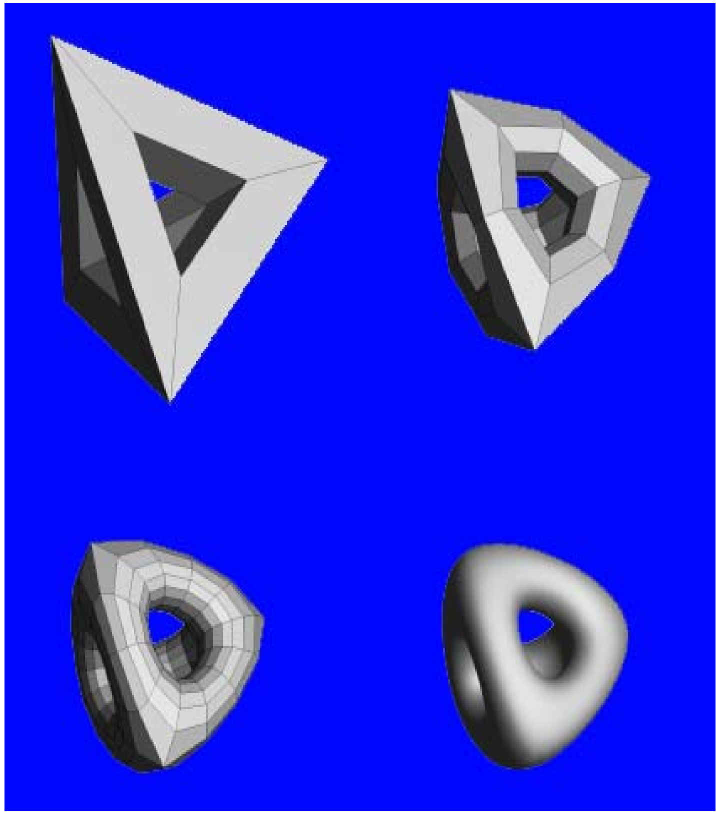
Example: Geri's Game

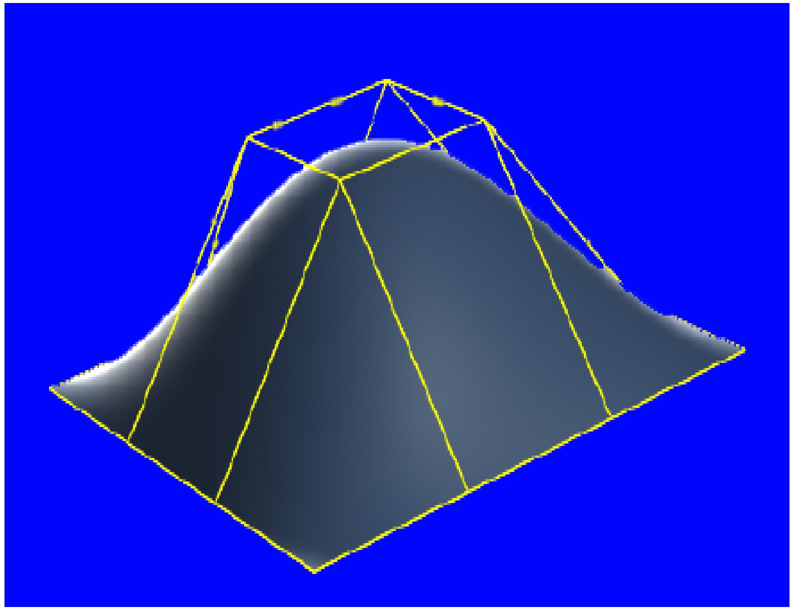
- **Subdivision surfaces are used for:**
  - Geri's hands and head
  - Clothes: Jacket, Pants, Shirt
  - Tie and Shoes

(Geri's Game, Pixar 1998)

Subdivision

- **Construct a surface from an arbitrary polyhedron**
  – Subdivide each face of the polyhedron
- **The limit will be a smooth surface**



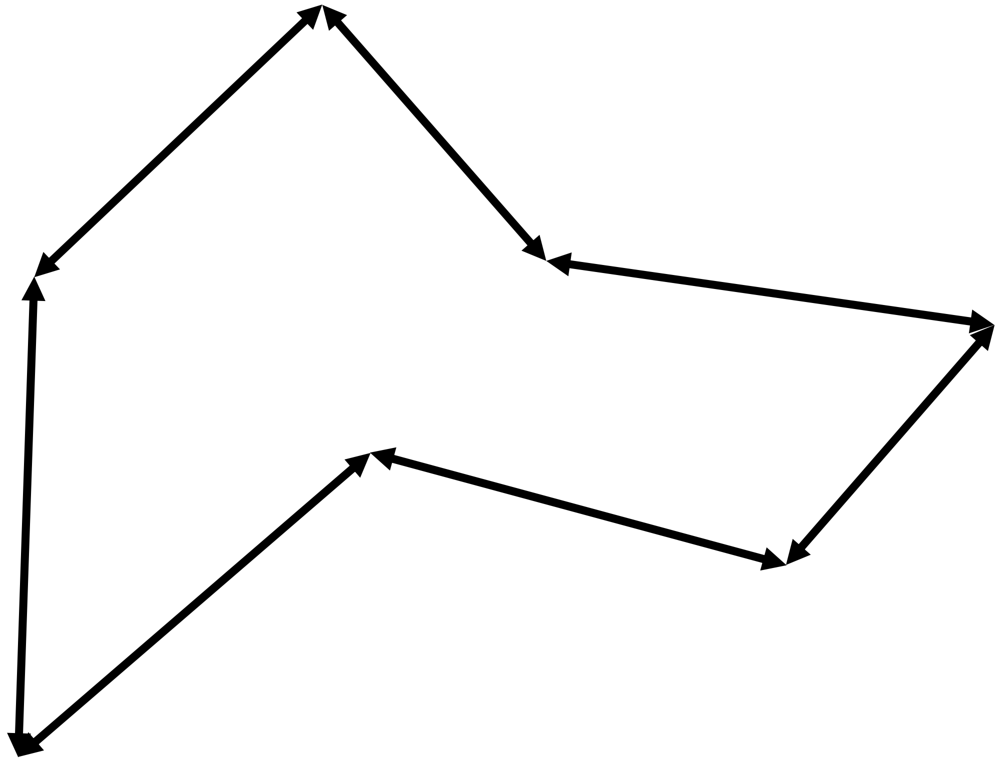


Subdivision Curves and Surfaces

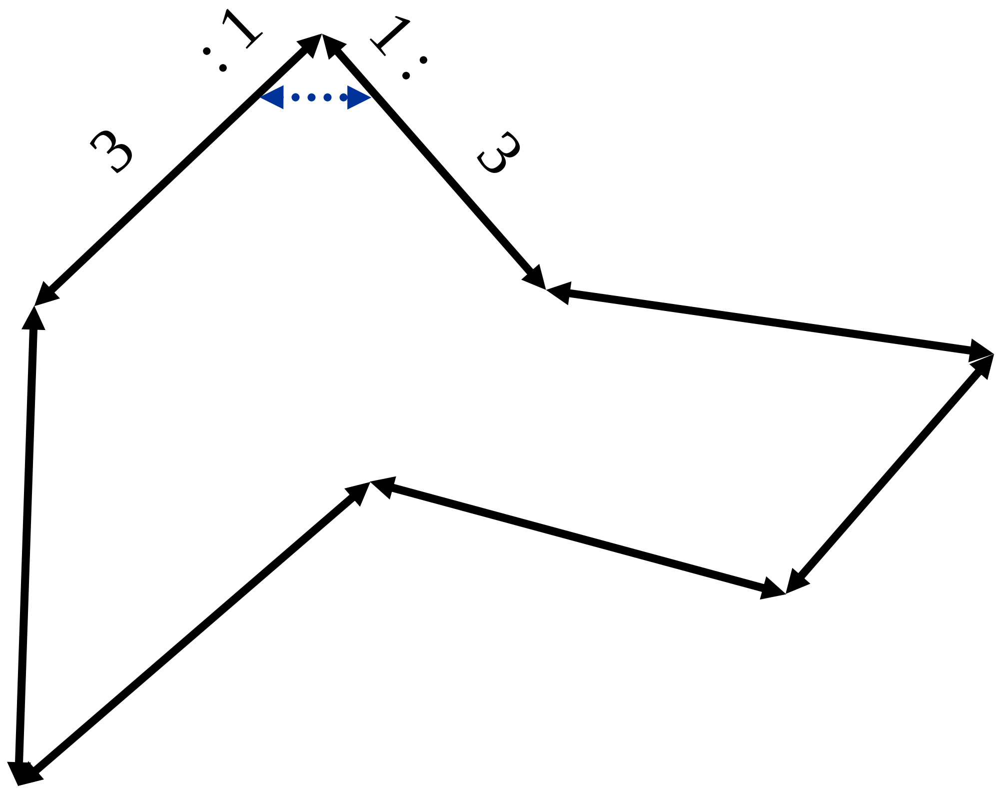
- **Subdivision curves**
  – The basic concepts of subdivision.
- **Subdivision surfaces**
  – Important known methods.
  – Discussion: subdivision vs. parametric surfaces.

Based on slides Courtesy of Adi Levin, Tel-Aviv U.

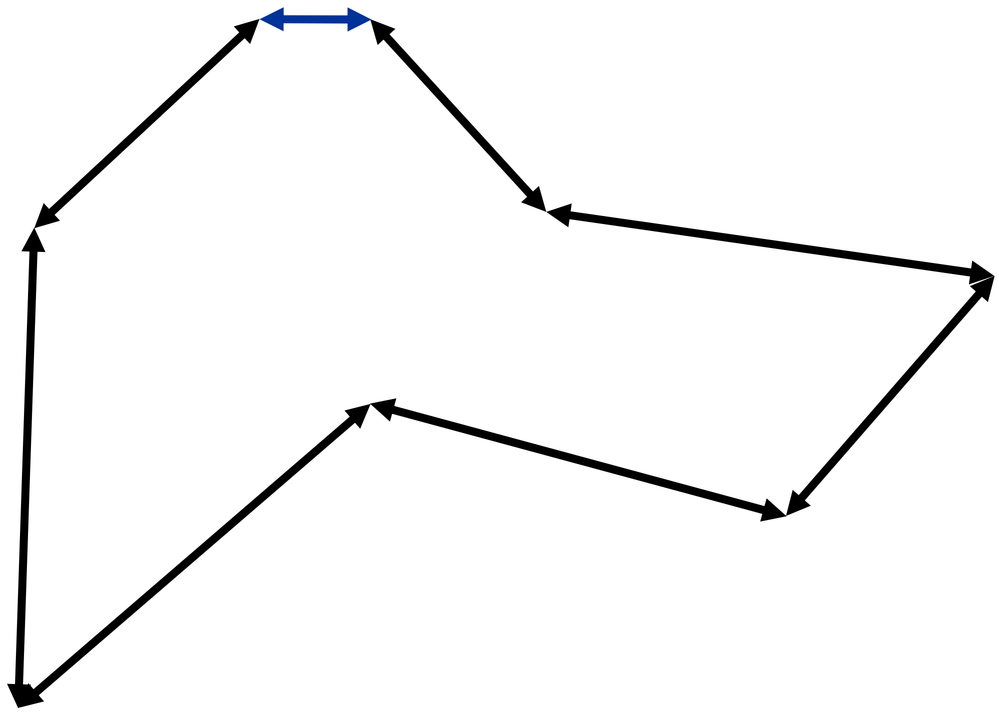
Curves: Corner Cutting



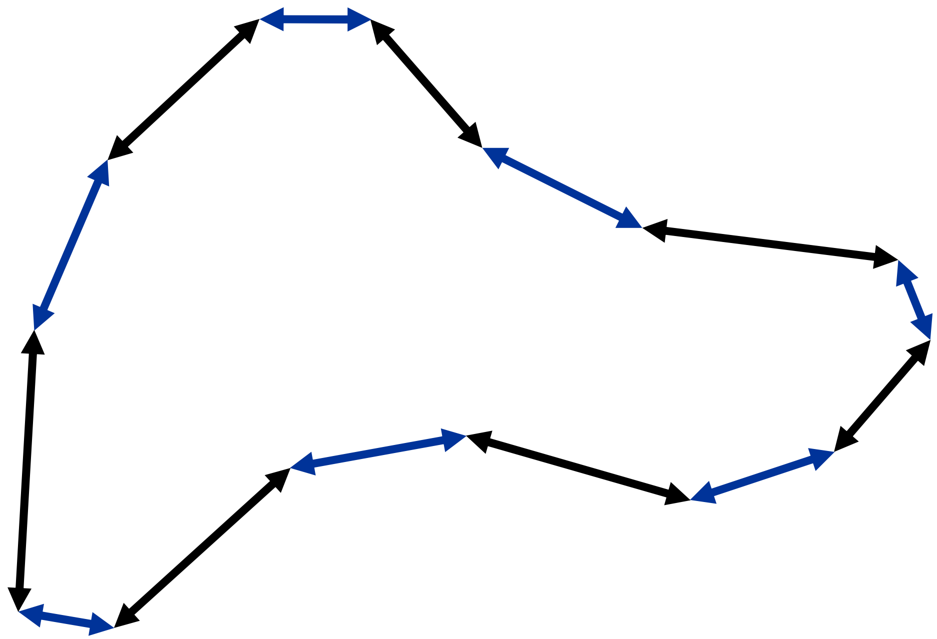
Corner Cutting



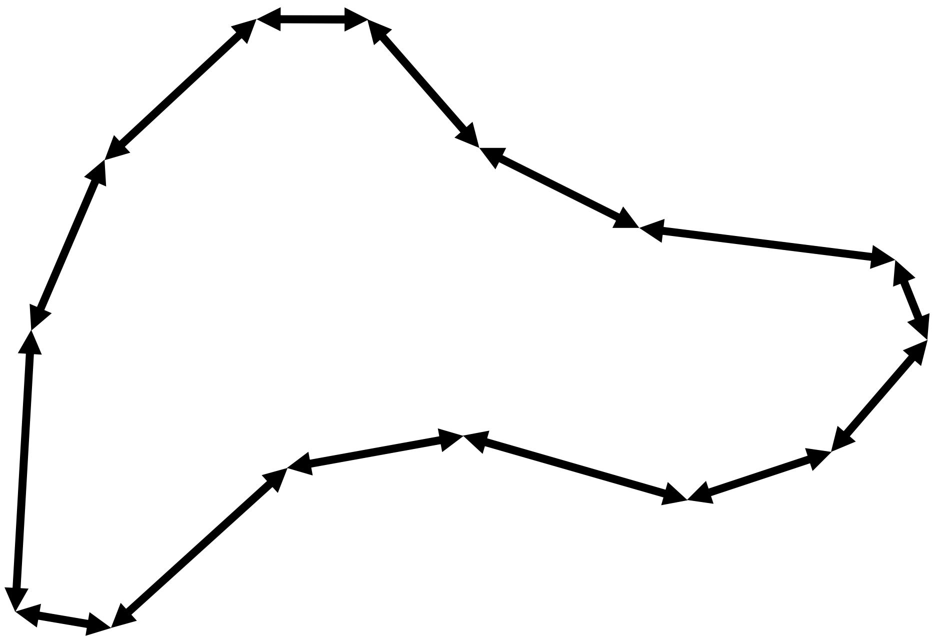
Corner Cutting



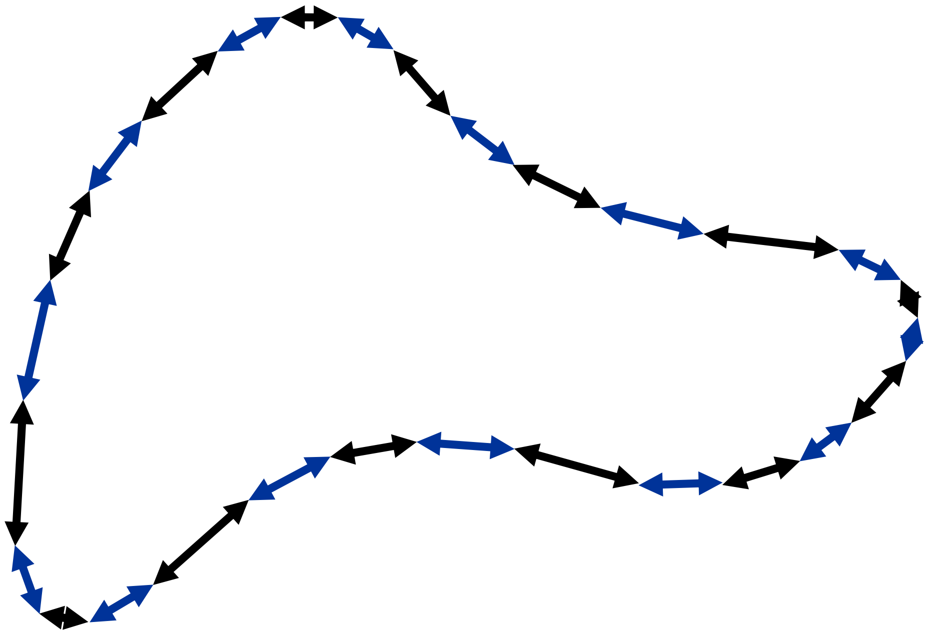
Corner Cutting



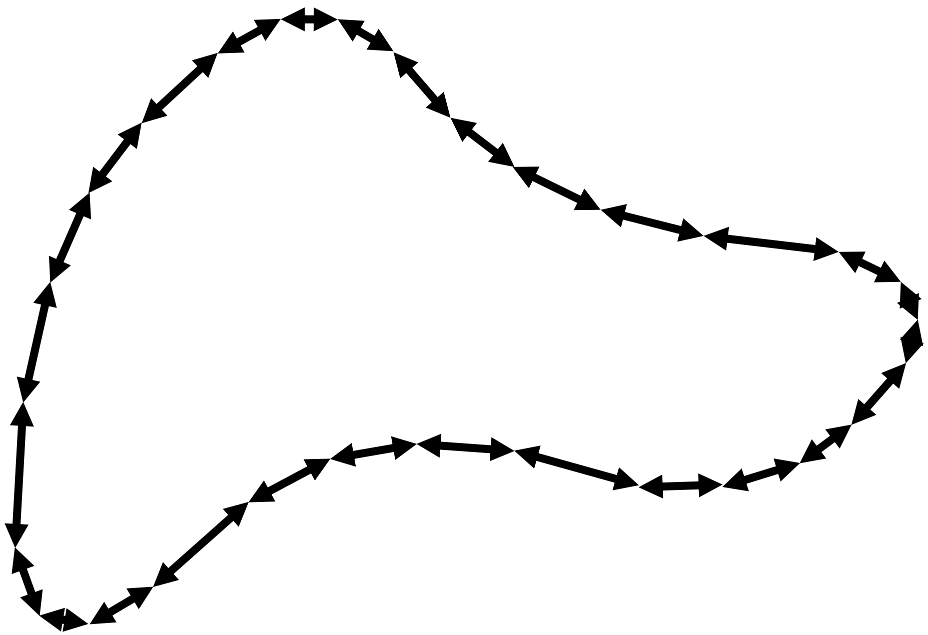
Corner Cutting



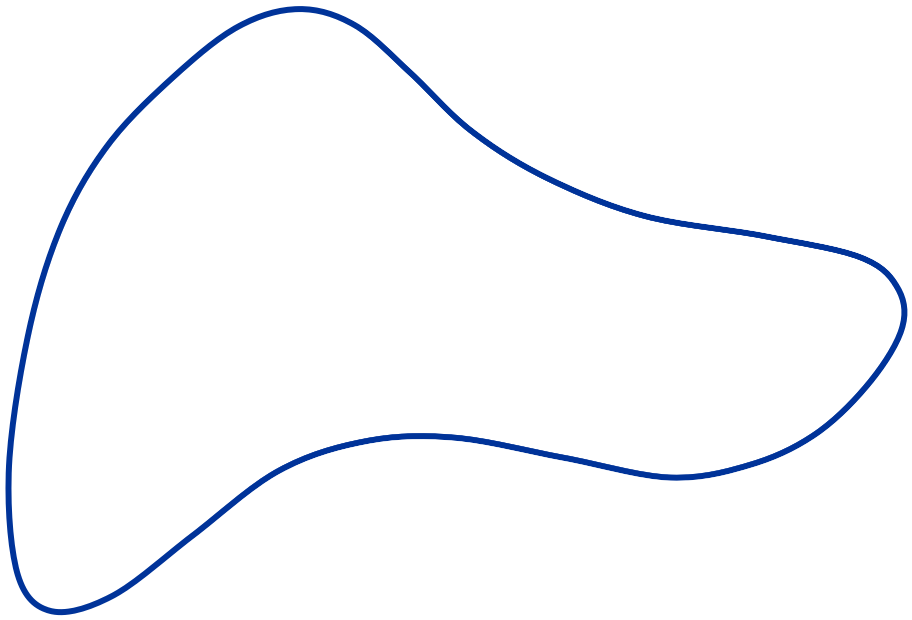
Corner Cutting



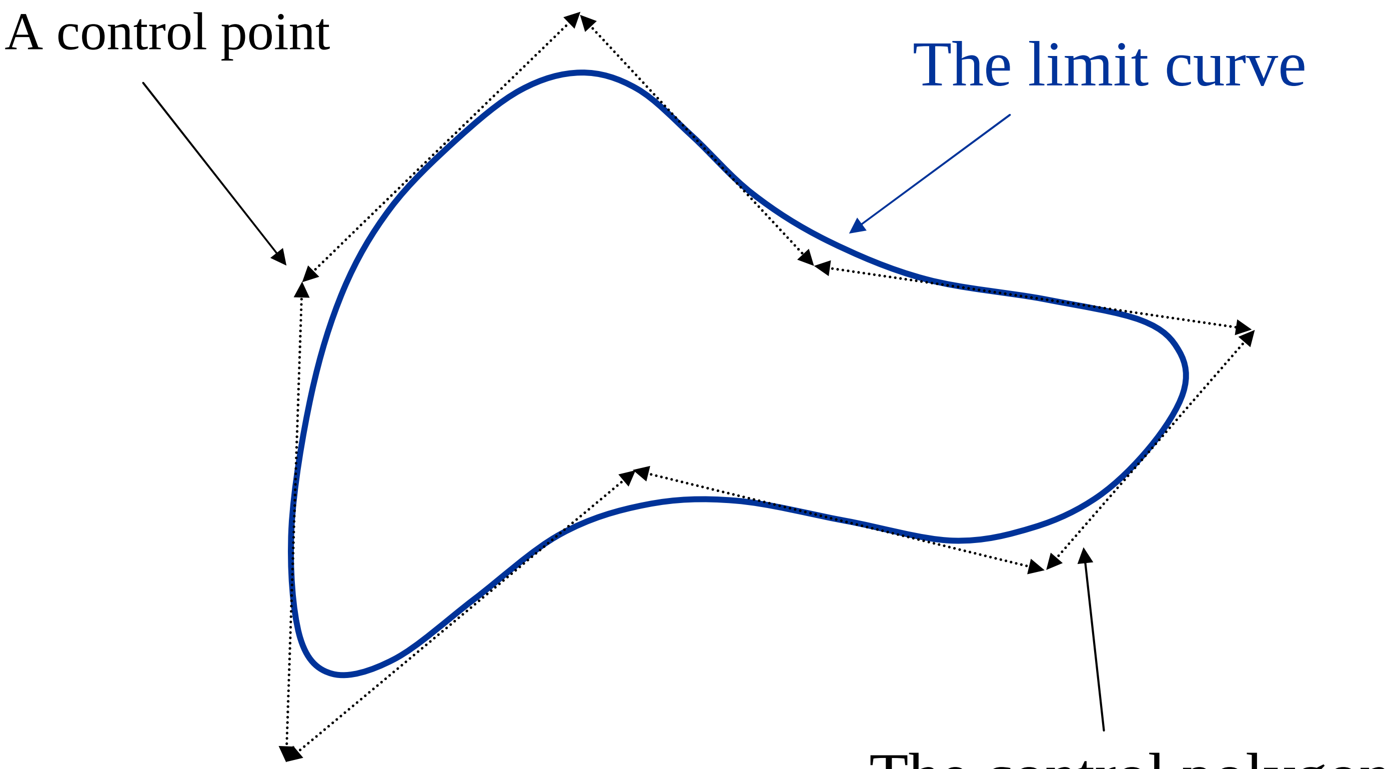
Corner Cutting



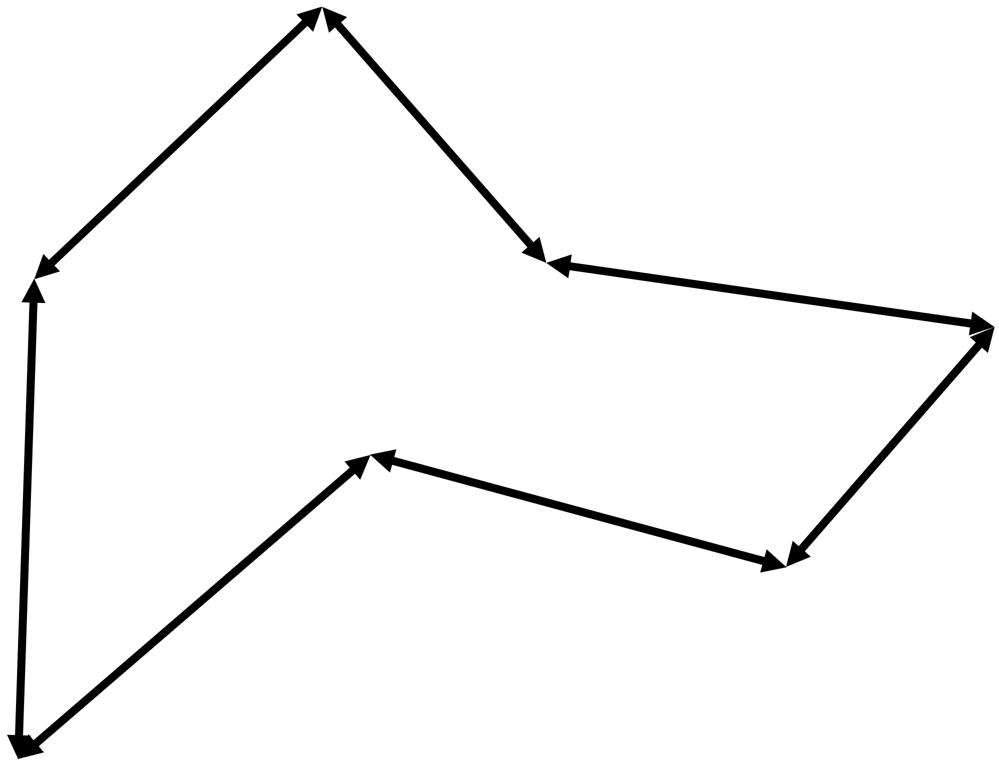
Corner Cutting



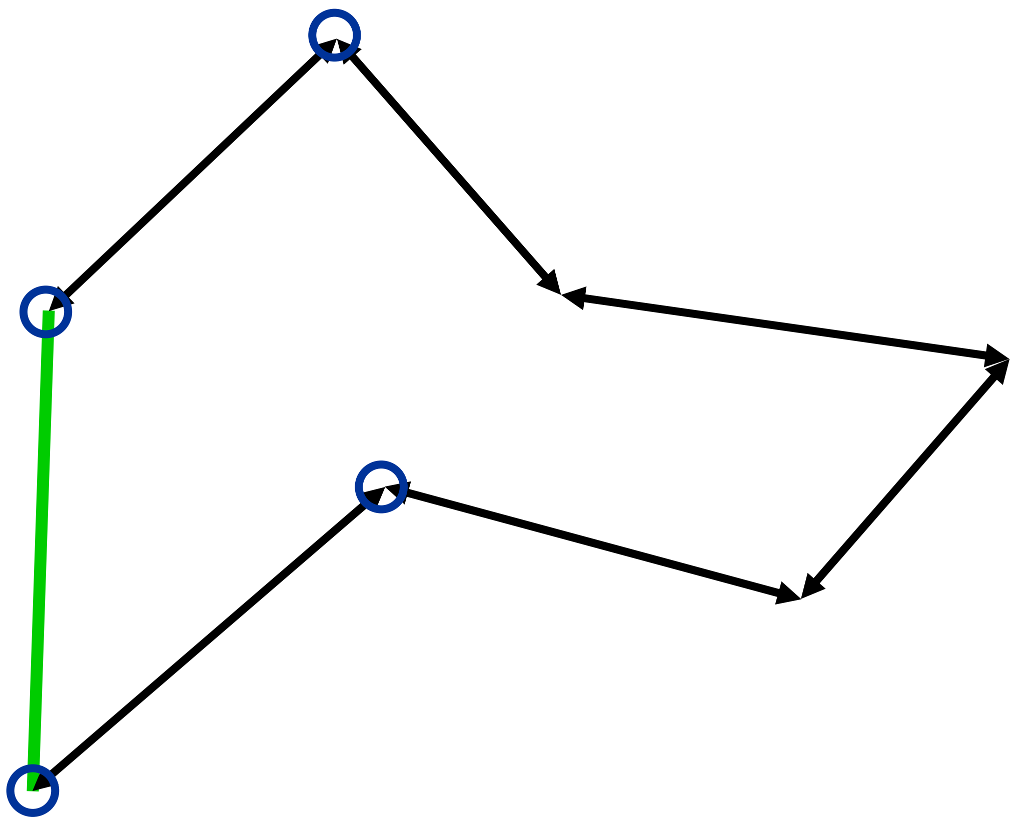
Corner Cutting



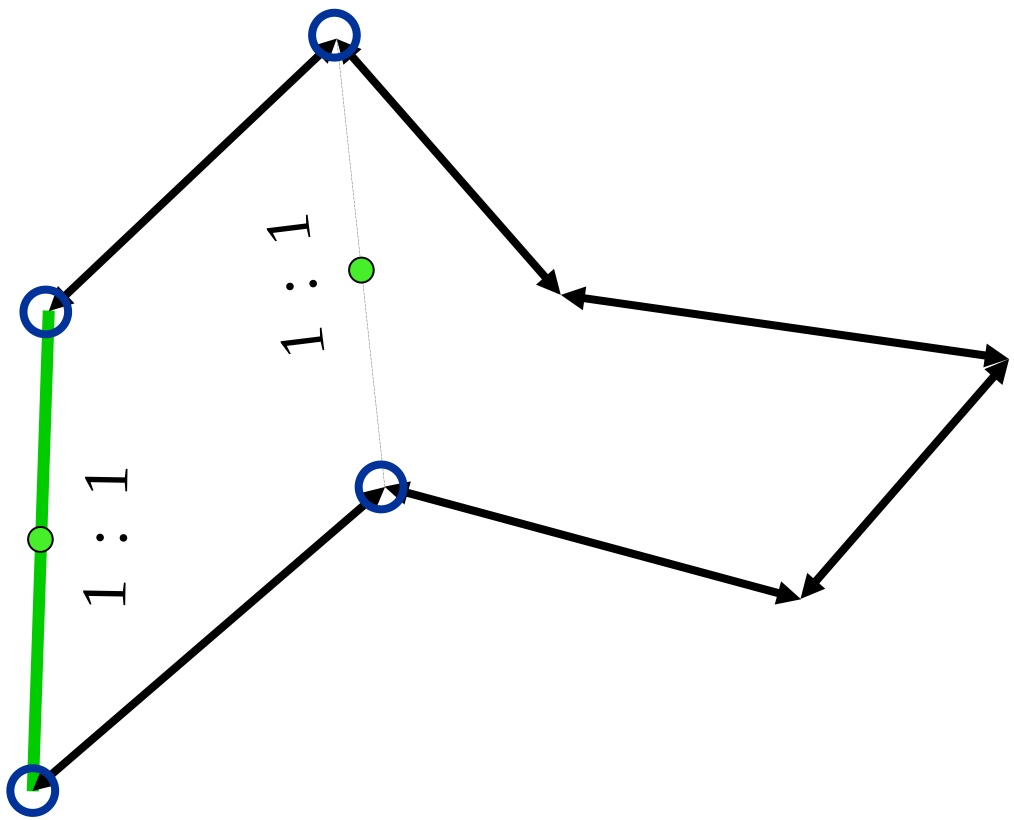
The 4-Point Scheme



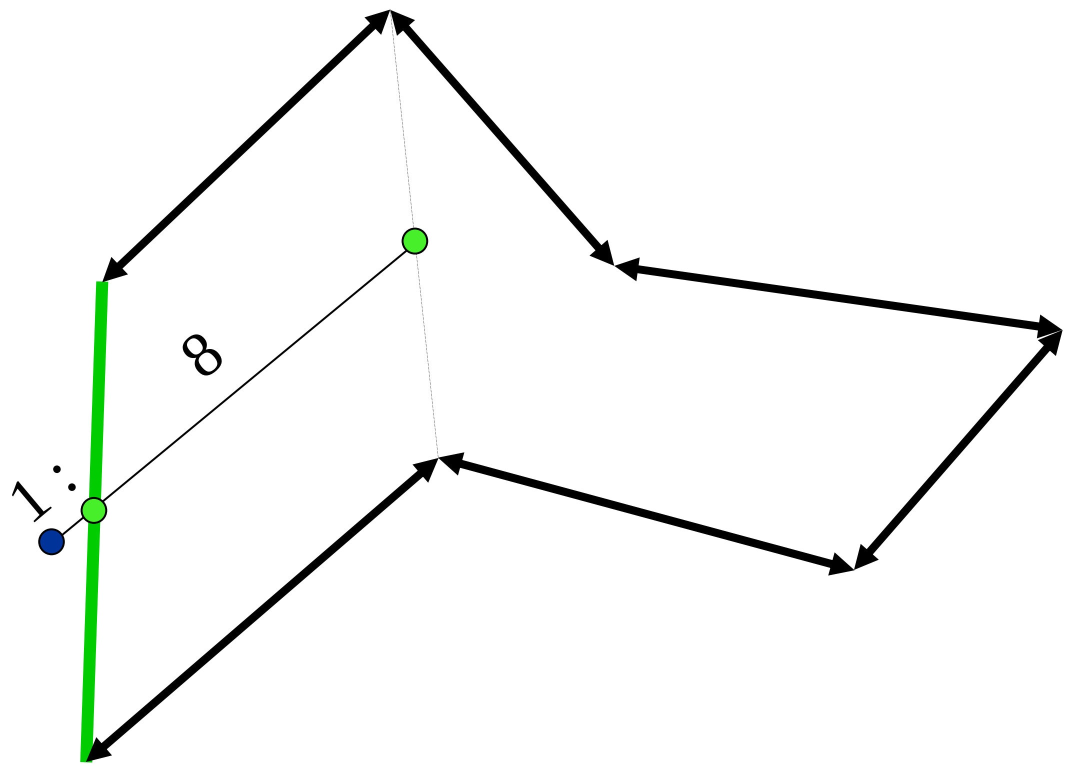
The 4-Point Scheme



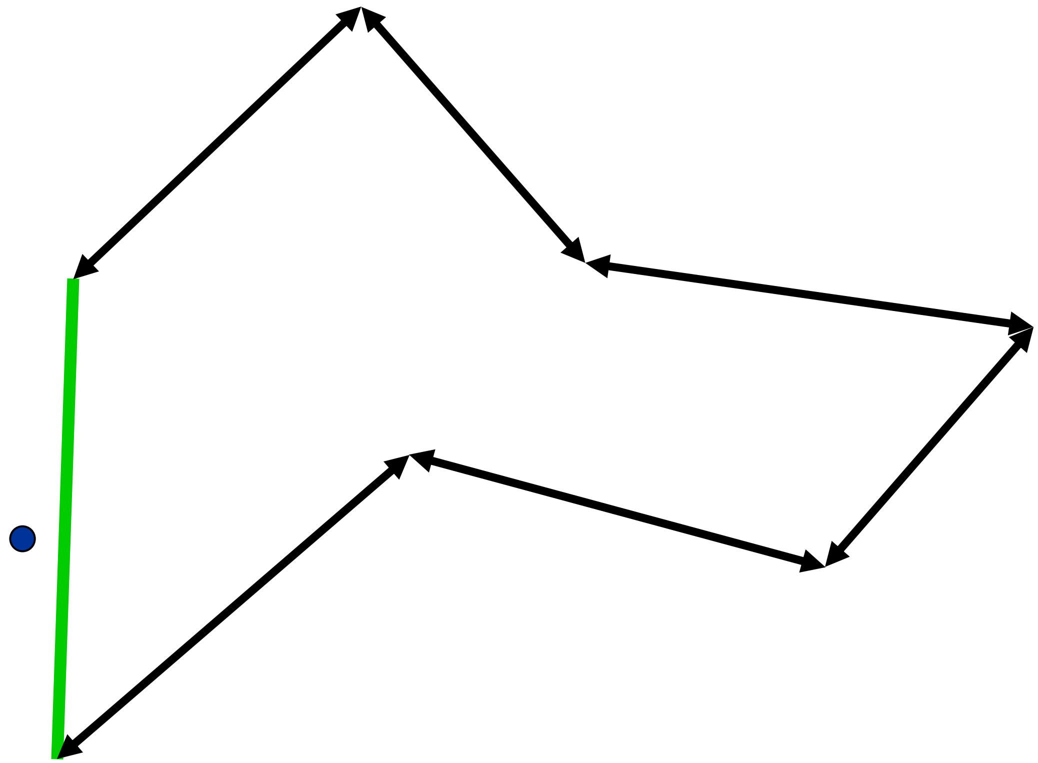
The 4-Point Scheme



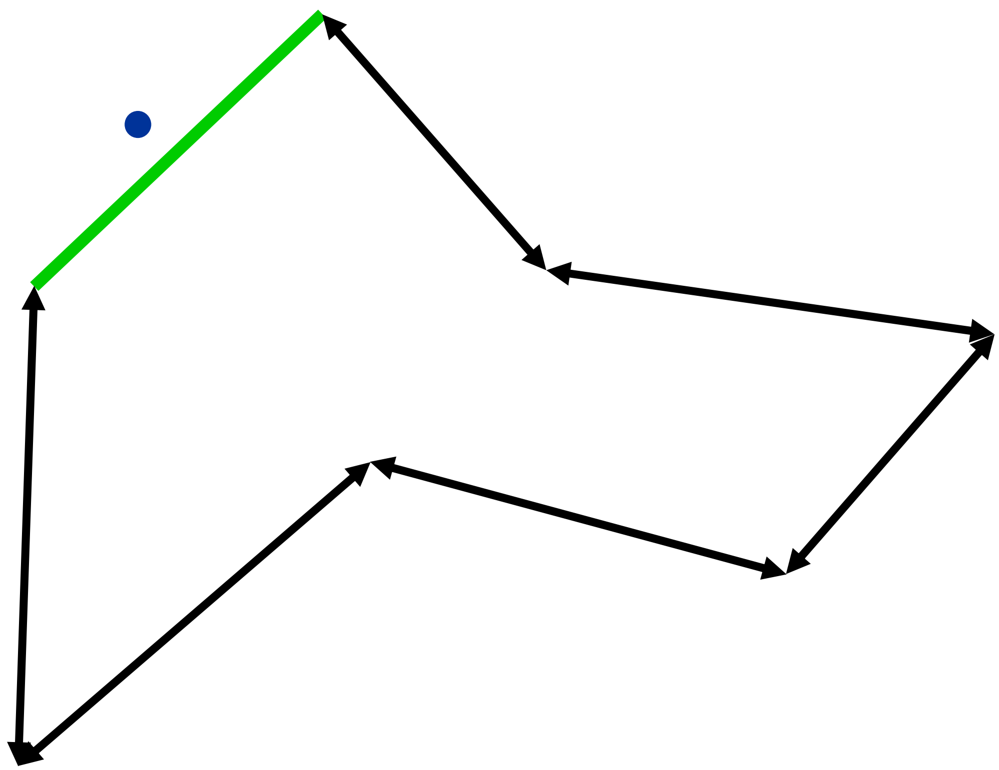
The 4-Point Scheme



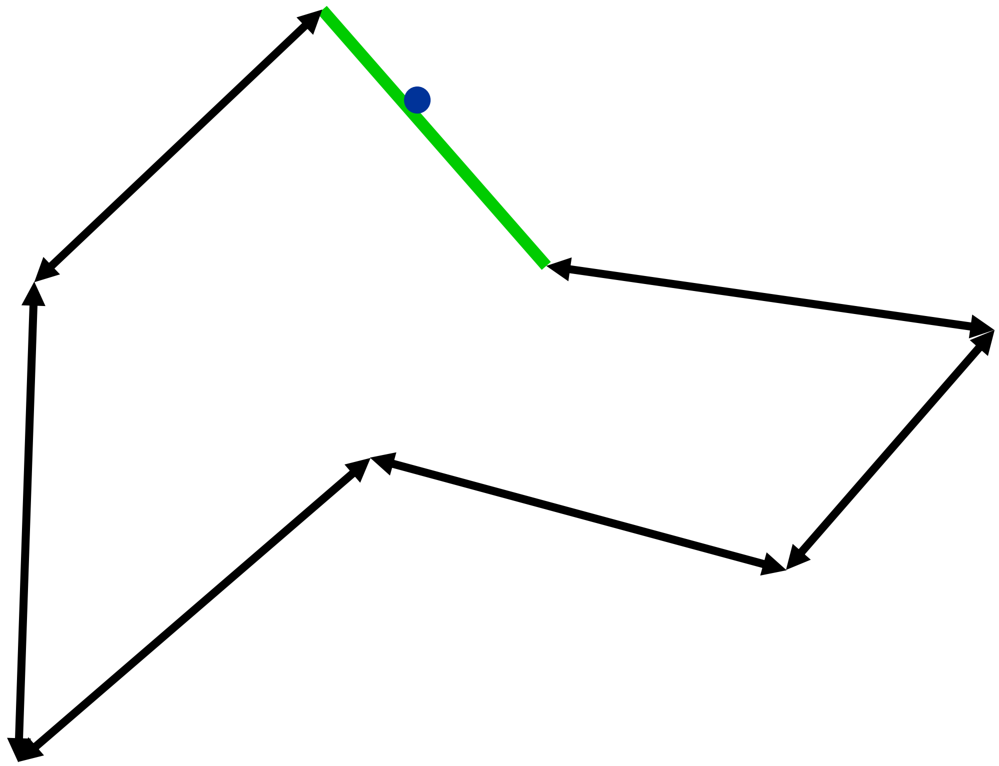
The 4-Point Scheme



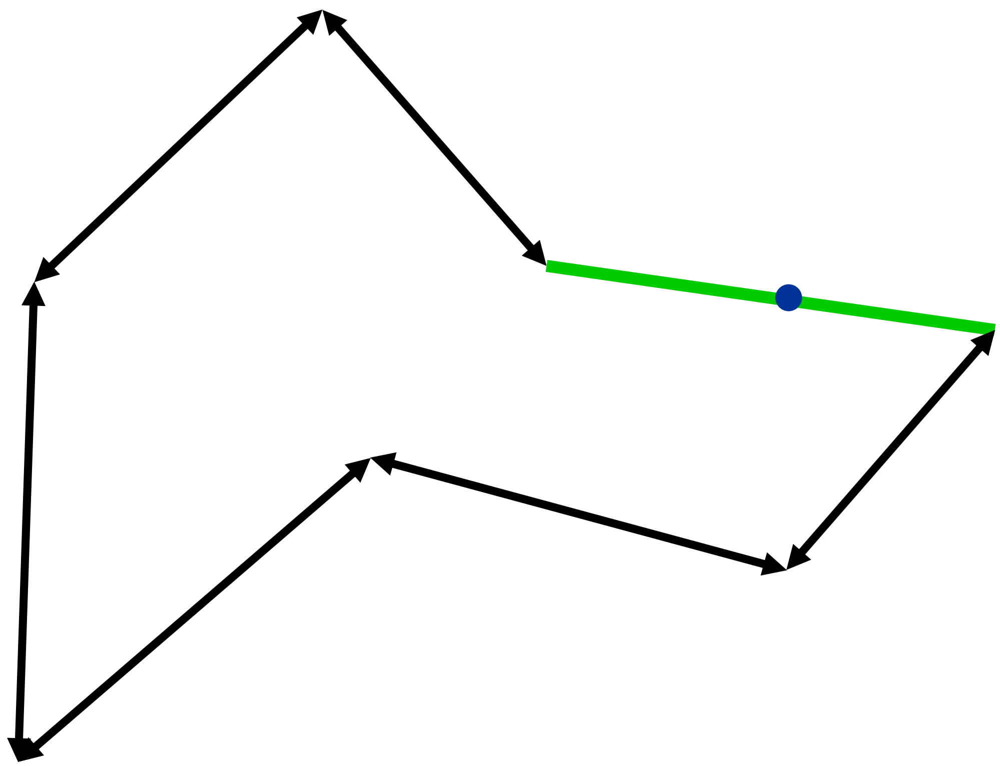
The 4-Point Scheme



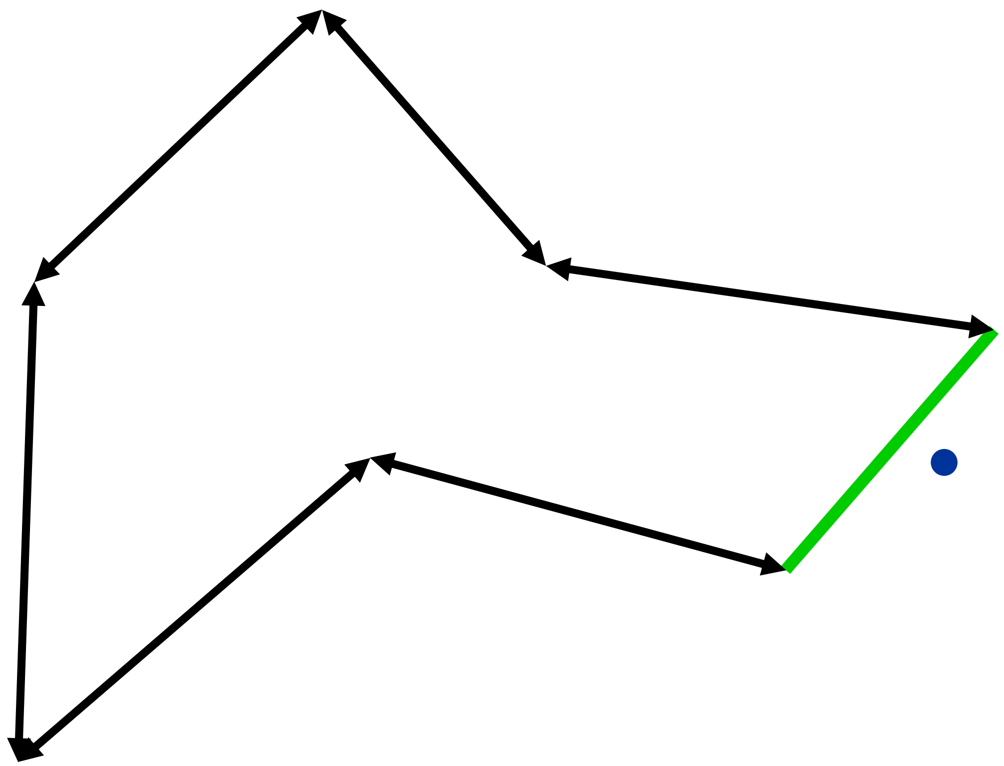
The 4-Point Scheme



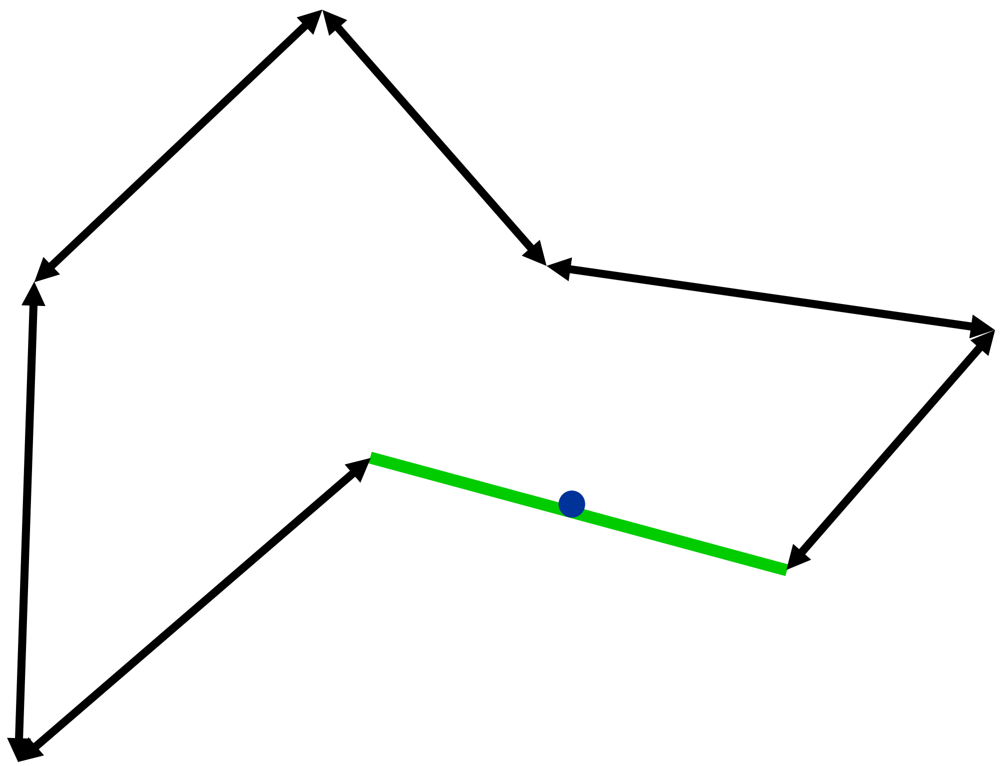
The 4-Point Scheme



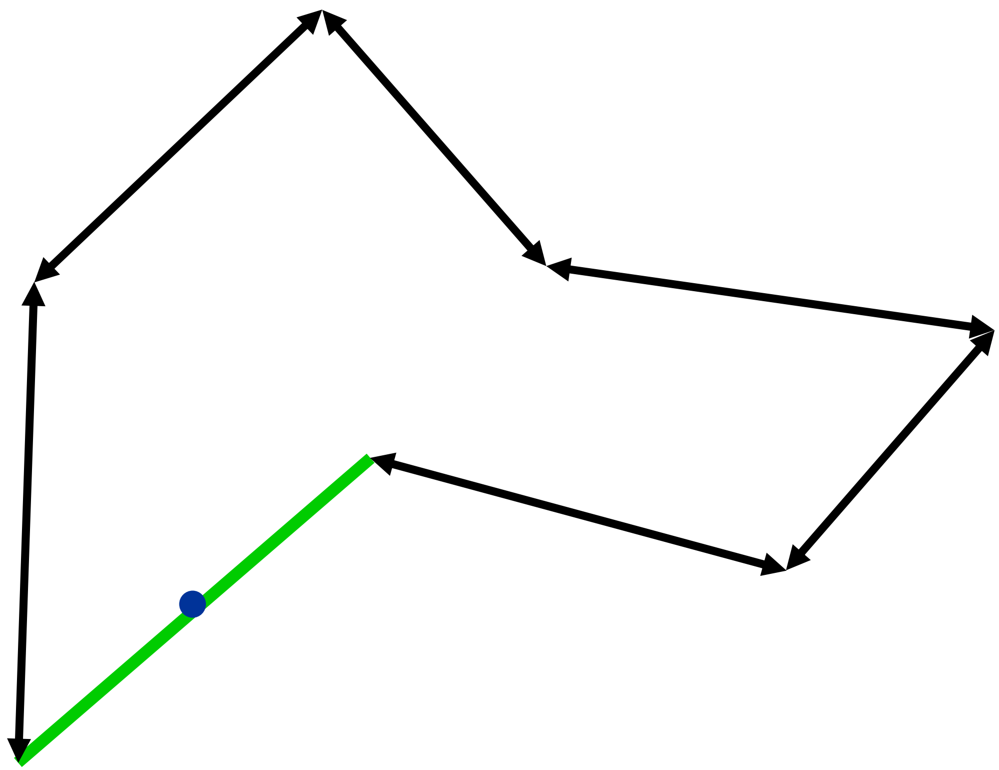
The 4-Point Scheme



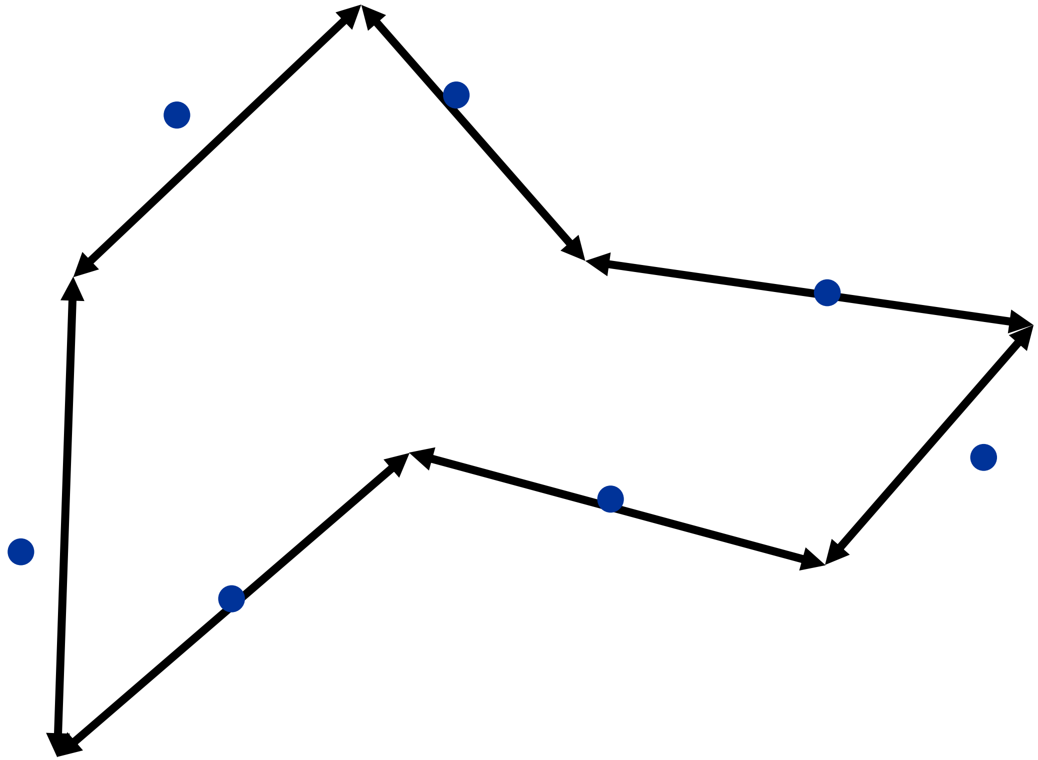
The 4-Point Scheme



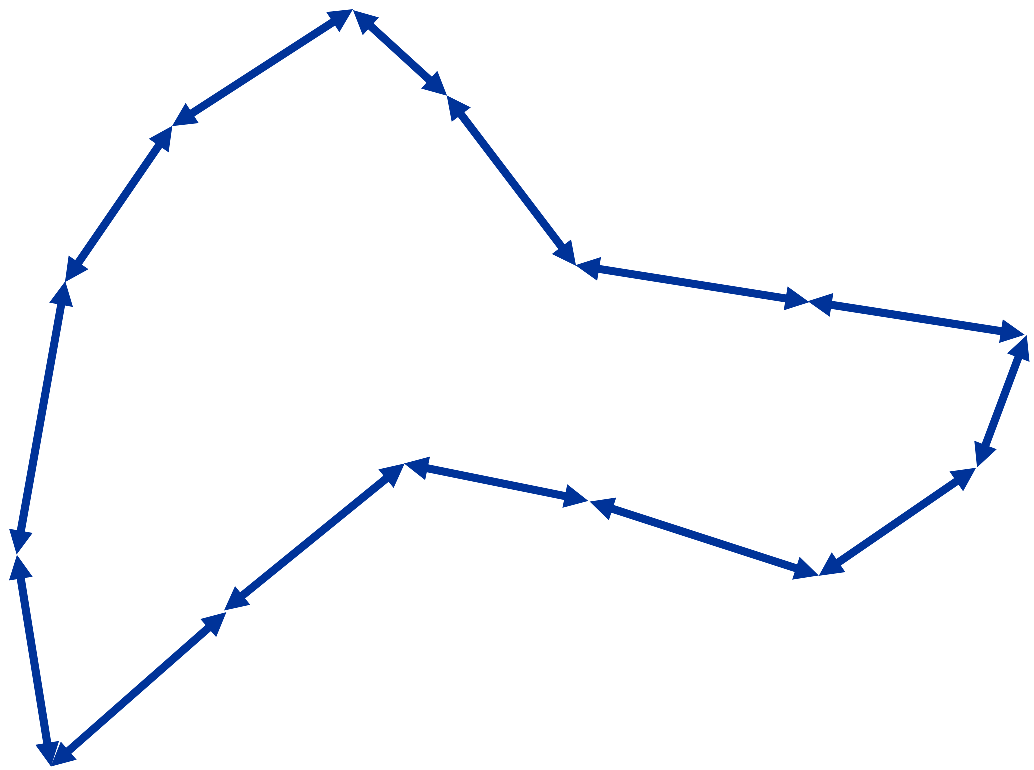
The 4-Point Scheme



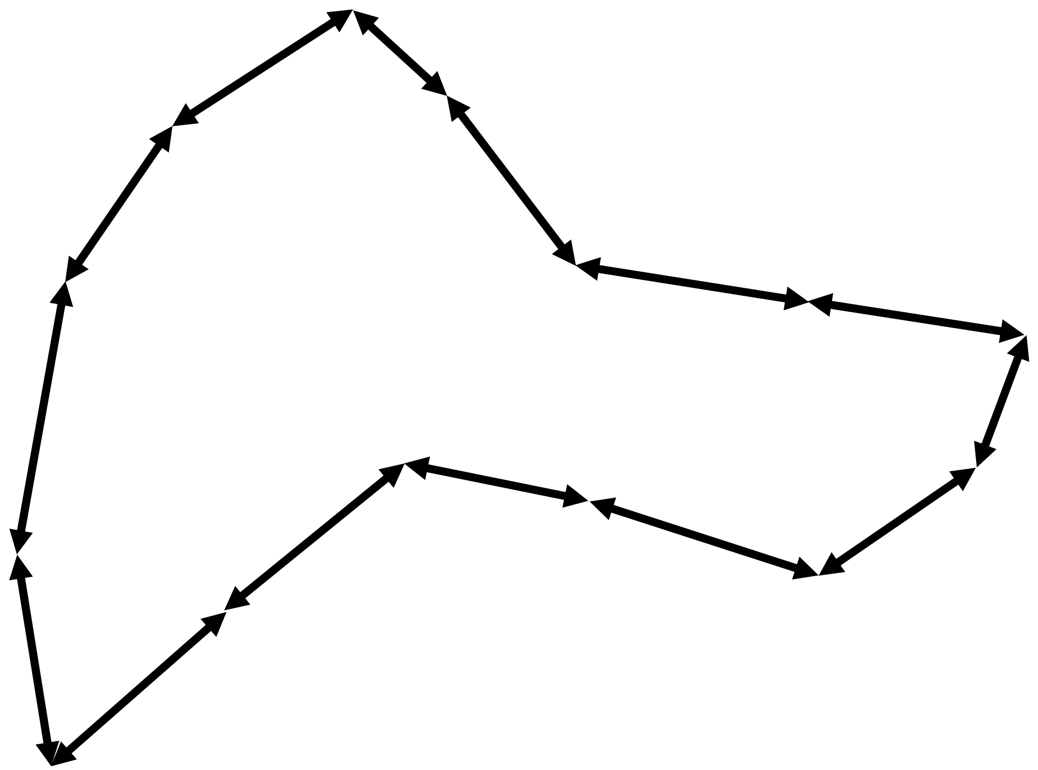
The 4-Point Scheme



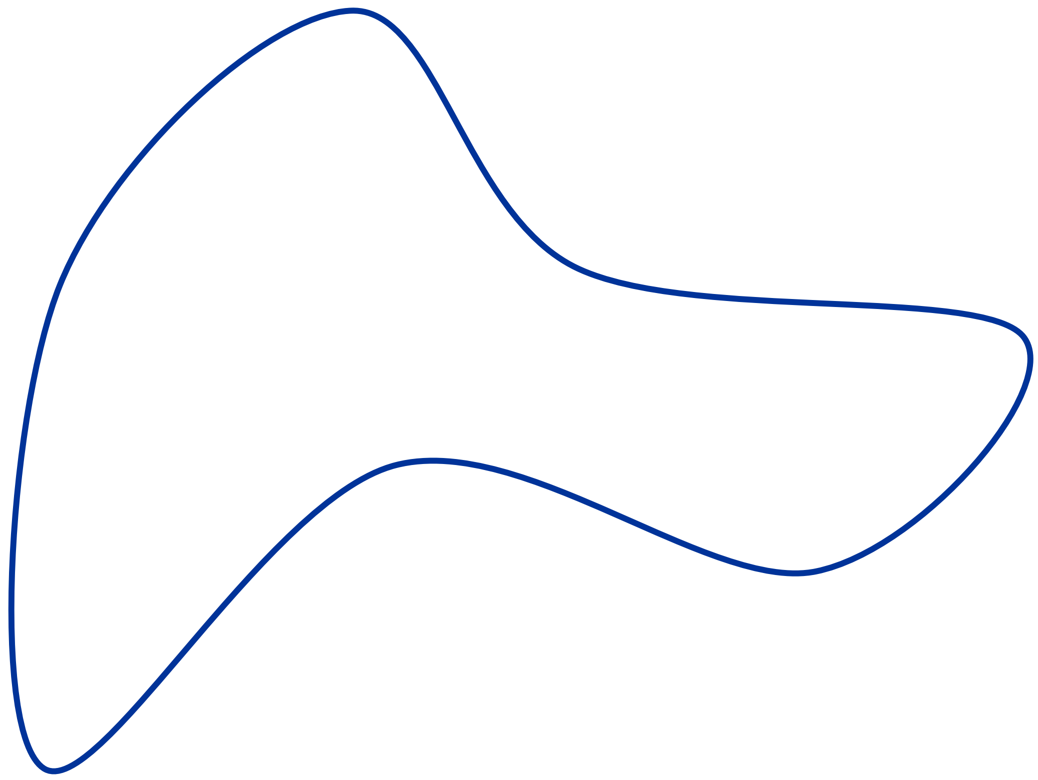
The 4-Point Scheme



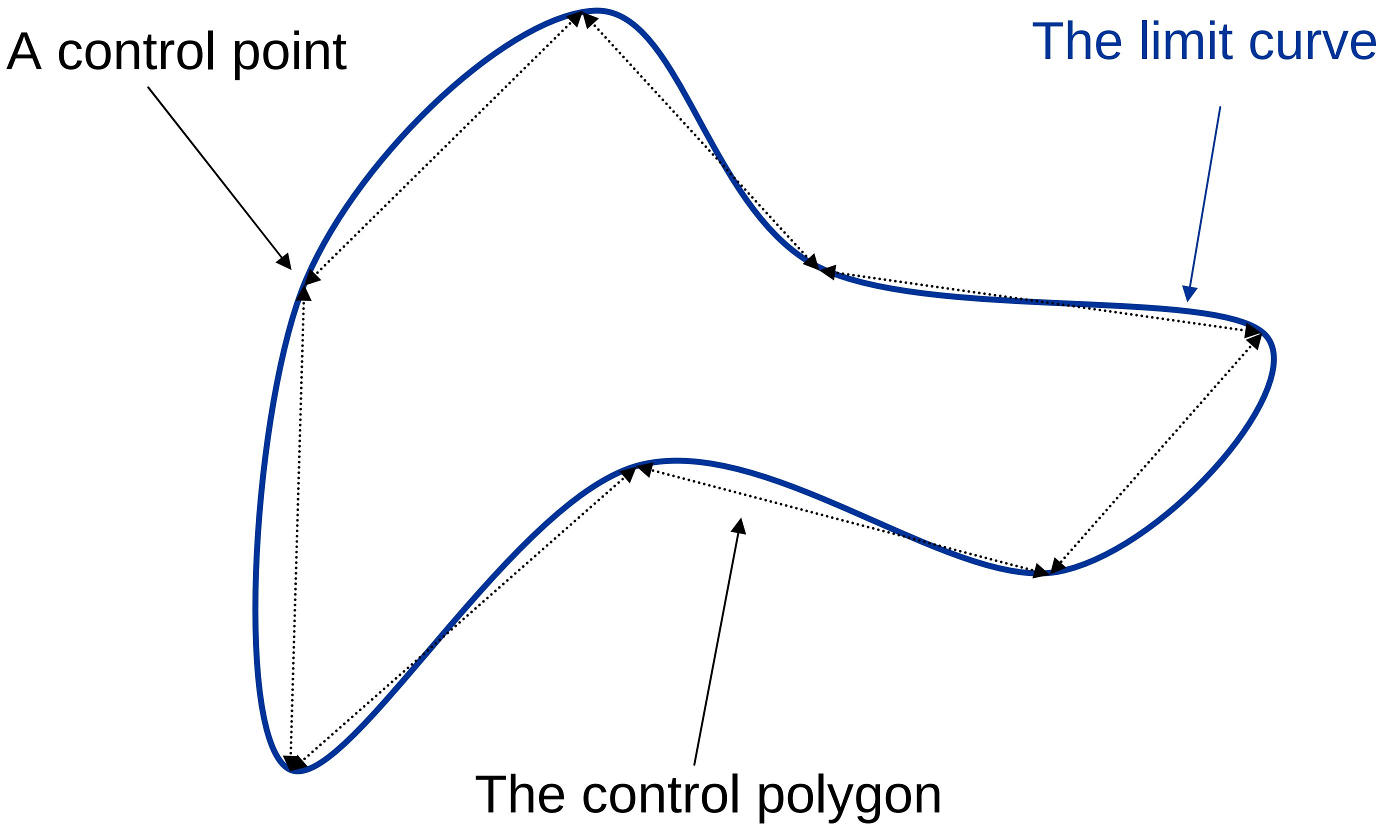
The 4-Point Scheme



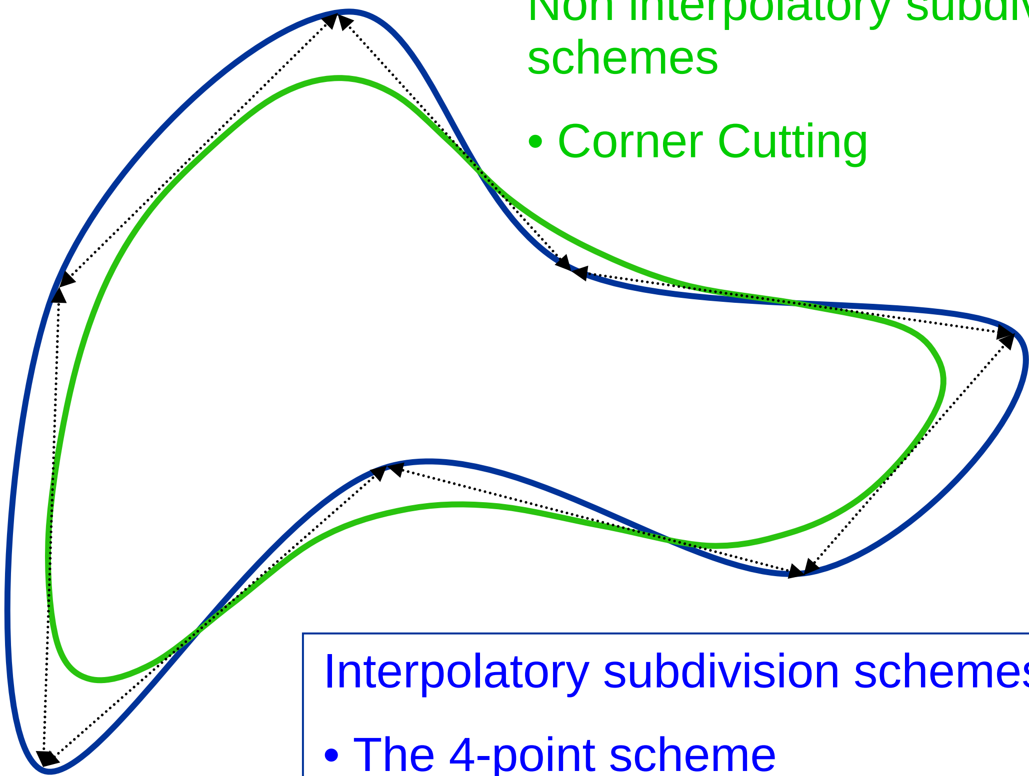
The 4-Point Scheme



The 4-Point Scheme



Subdivision Curves



Non interpolatory subdivision schemes

- Corner Cutting

Interpolatory subdivision schemes

- The 4-point scheme

Basic Concepts of Subdivision

- **Definition**
  - A subdivision curve is generated by repeatedly applying a subdivision operator to a given polygon (called the control polygon).

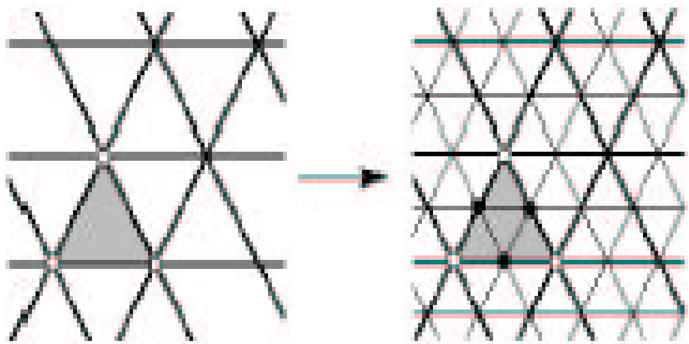
- **The central theoretical questions:**
  - Convergence:
    Given a subdivision operator and a control polygon, does the subdivision process converge?
  - Smoothness:
    Does the subdivision process converge to a smooth curve?

Surfaces Subdivision Schemes

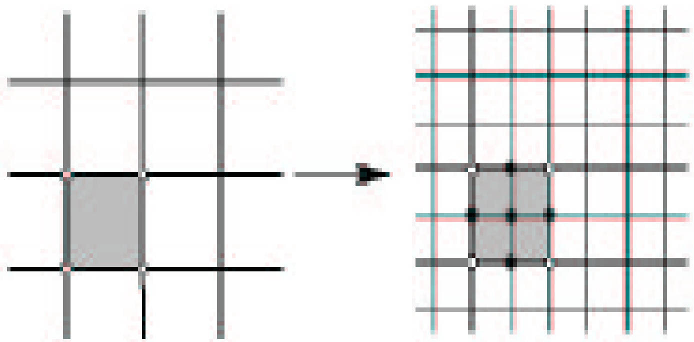
- **A *control net* consists of vertices, edges, and faces.**
- **Refinement**
  - In each iteration, the subdivision operator refines the control net, increasing the number of vertices (approximately) by a factor of 4.
- **Limit Surface**
  - In the limit the vertices of the control net converge to a limit surface.
- **Topology and Geometry**
  - Every subdivision method has a method to generate the topology of the refined net, and rules to calculate the location of the new vertices.

Subdivision Schemes

- **There are different subdivision schemes**
  - Different methods for refining topology
- **Different rules for positioning vertices**
  - Interpolating versus approximating



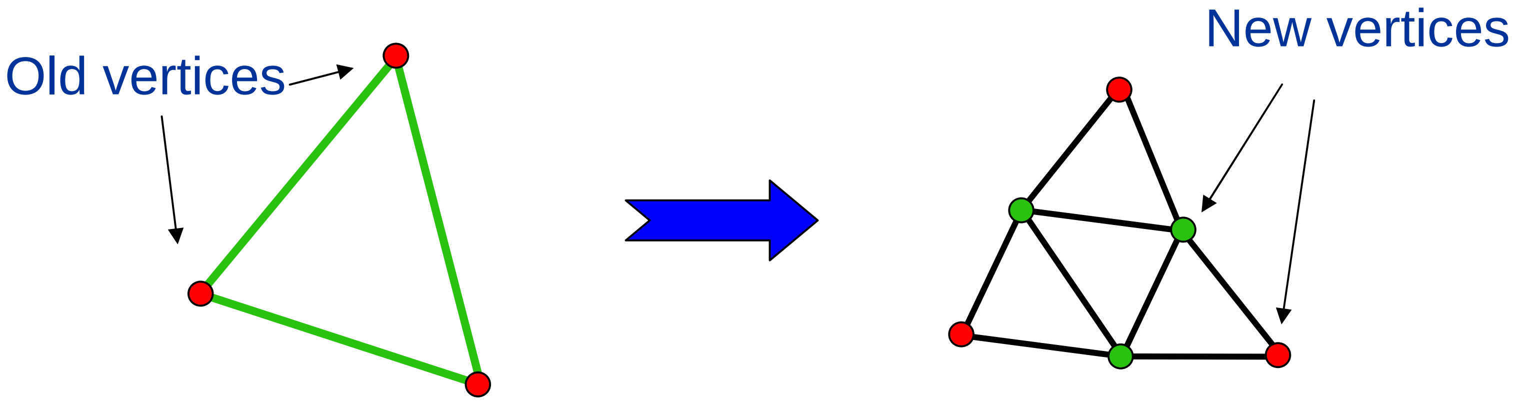
Face split for triangles



Face split for quads

Triangular Subdivision

- For control nets whose faces are triangular.



Every face is replaced by 4 new triangular faces.

The are two kinds of new vertices:

- Green vertices are associated with old edges
- Red vertices are associated with old vertices.

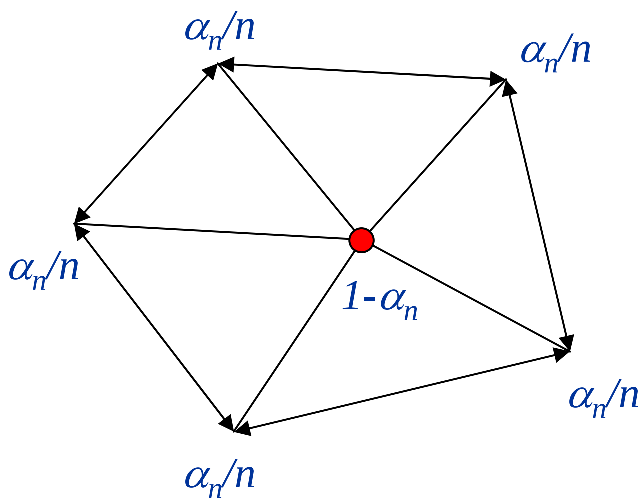
Loop Subdivision Scheme

- **Works on triangular meshes**
- **Is an Approximating Scheme**
- **Guaranteed to be smooth everywhere except at** *extraordinary* **vertices.**

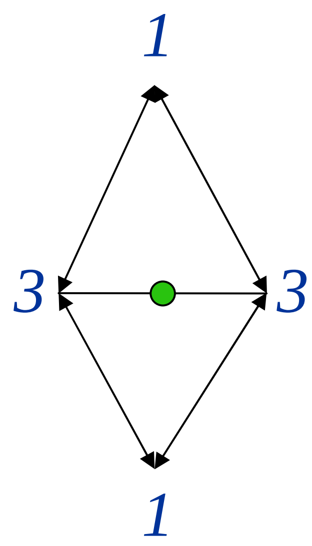
Loop's Scheme

- **Location of New Vertices**
  - Every new vertex is a weighted average of the old vertices. The list of weights is called the subdivision mask or the stencil

A rule for new red vertices



A rule for new green vertices



$$\alpha_n = \frac{1}{64}\left(40 - \left(3 + 2\cos\left(\frac{2\pi}{n}\right)\right)^2\right)$$

Original

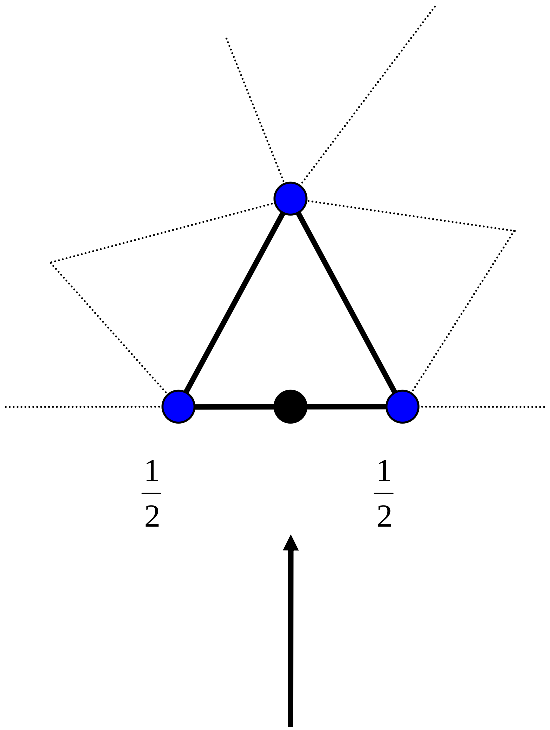
$$\alpha_n = \begin{cases} \frac{3}{8} & n > 3 \\ \frac{3}{16} & n = 3 \end{cases}$$

Warren

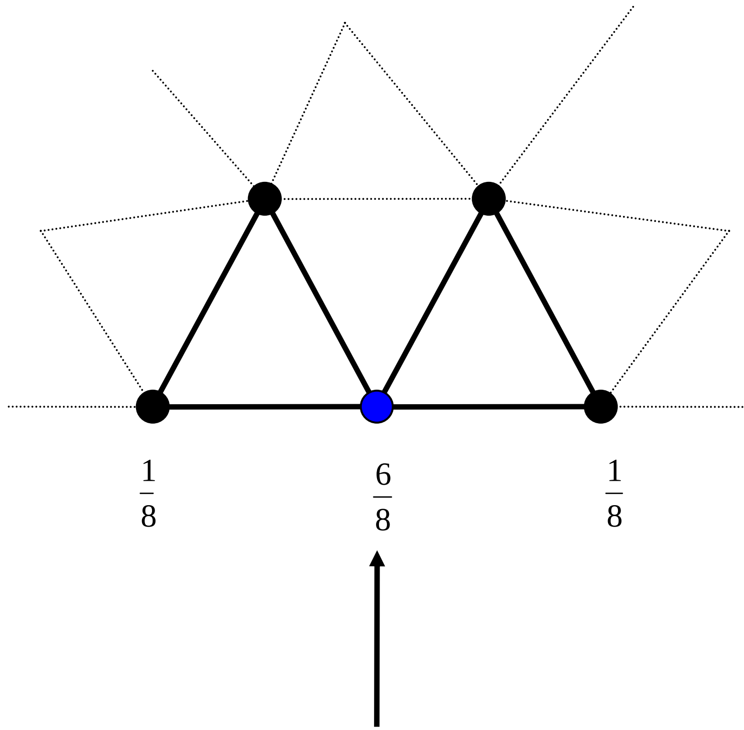
n - the vertex valence

Loop Subdivision Boundaries

- **Subdivision Mask for Boundary Conditions**



Edge Rule



Vertex Rule

Subdivision as Matrices

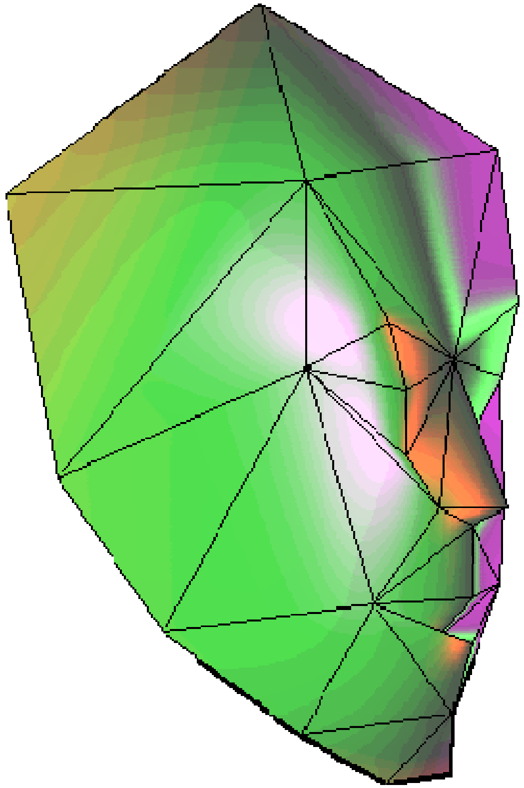
- **Subdivision can be expressed as a matrix *S_{mask}* of weights *w*.**
  - *S_{mask} is very sparse*
  - *Never Implement this way!*
  - Allows for analysis
    - Curvature
    - Limit Surface

$$S_{mask} P = \hat{P}$$

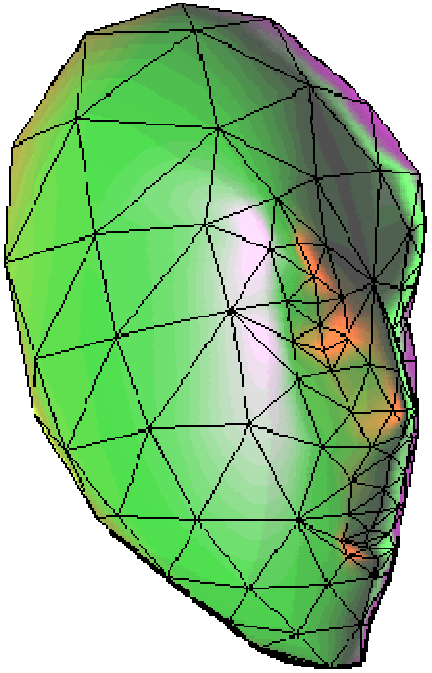
$$\begin{bmatrix} w_{00} & w_{01} & \cdots & 0 \\ w_{10} & w_{11} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & w_{nj} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} \hat{p}_0 \\ \hat{p}_1 \\ \hat{p}_2 \\ \vdots \\ \hat{p}_0 \end{bmatrix}$$

S_{mask} Weights — Old Control Points — New Points

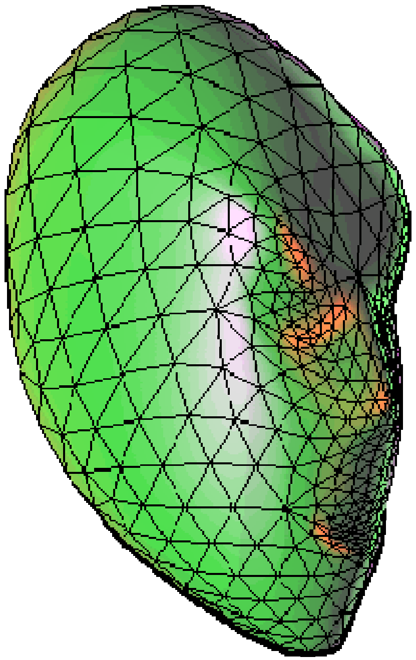
The Original Control Net



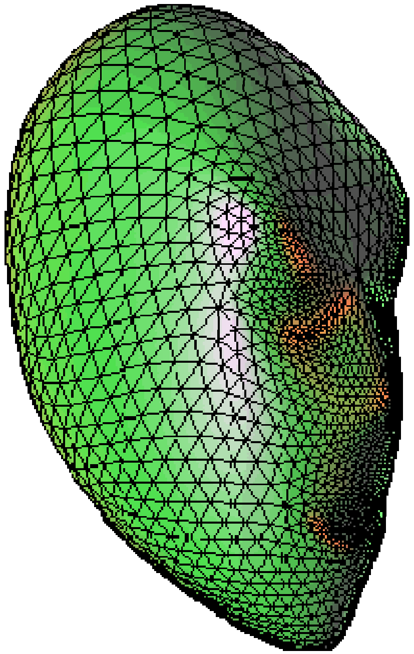
After 1st Iteration



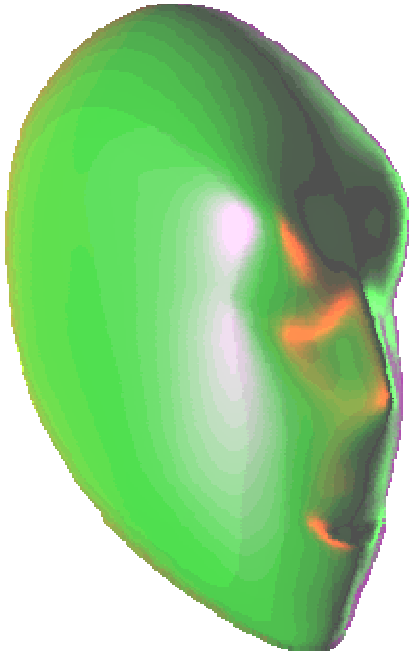
After 2nd Iteration



After 3rd Iteration



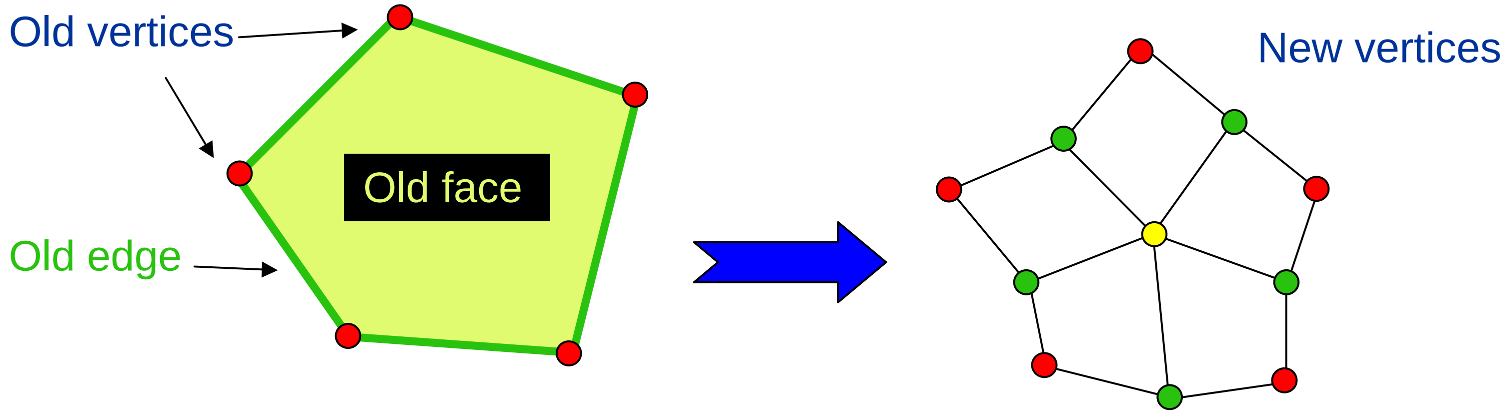
The Limit Surface



The limit surfaces of Loop's subdivision have continuous curvature almost everywhere

Quadrilateral Subdivision

- **Works for control nets of arbitrary topology**
  – After one iteration, all the faces are quadrilateral.



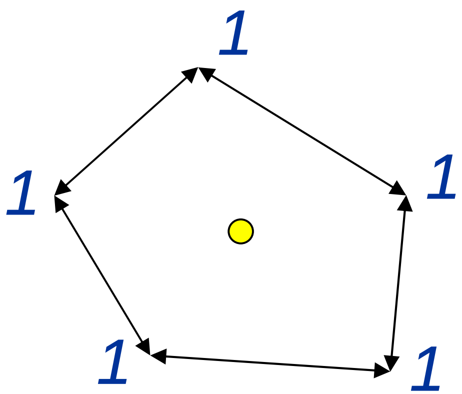
Every face is replaced by quadrilateral faces.
The are three kinds of new vertices:

- Yellow vertices are associated with old faces
- Green vertices are associated with old edges
- Red vertices are associated with old vertices.

Catmull Clark's Scheme

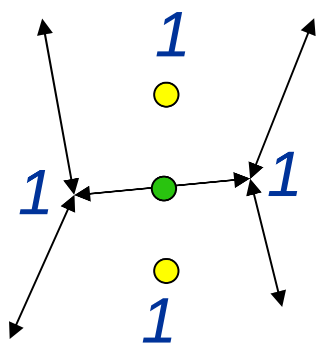
Step 1

First, all the yellow vertices are calculated



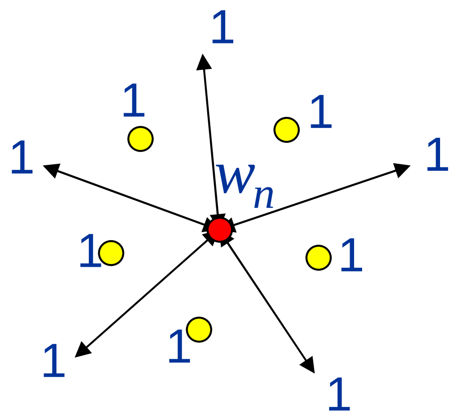
Step 2

Then the green vertices are calculated using the values of the yellow vertices



Step 3

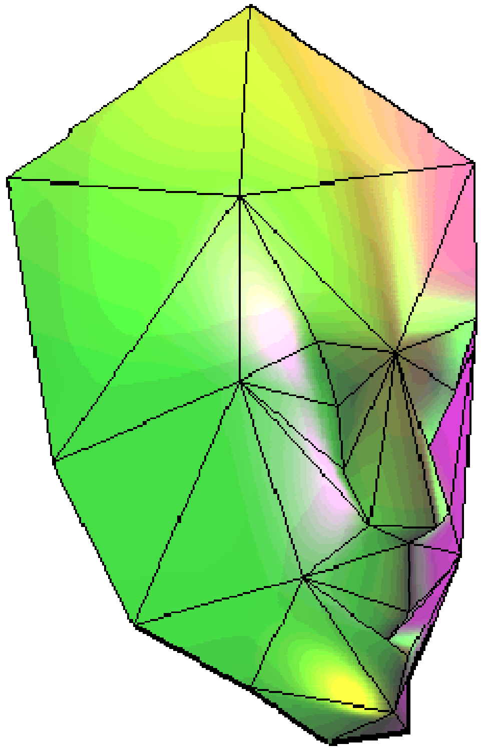
Finally, the red vertices are calculated using the values of the yellow vertices



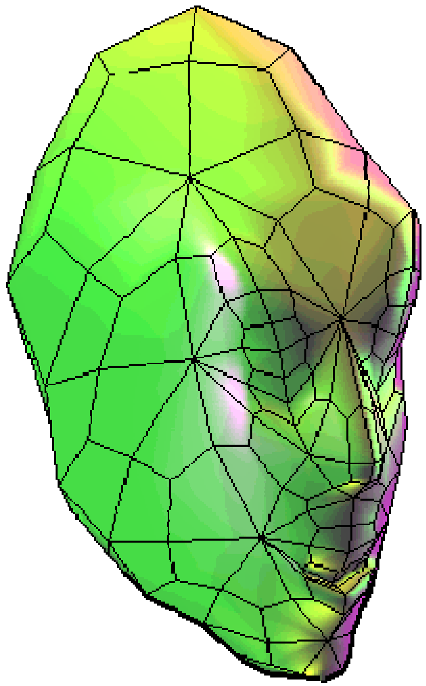
n - the vertex valence

$$w_n = n(n-2)$$

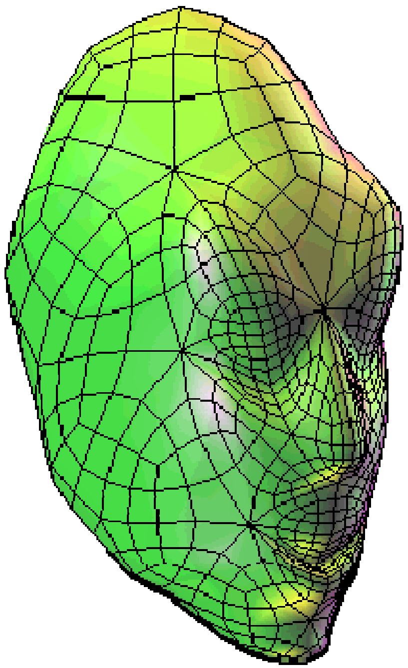
The Original Control Net



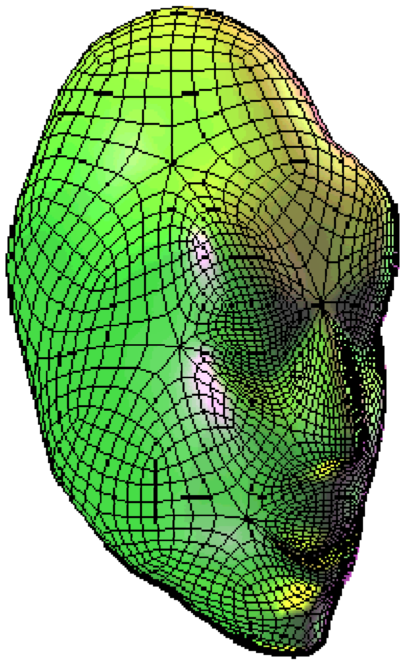
After 1st Iteration



After 2nd Iteration



After 3rd Iteration



The Limit Surface

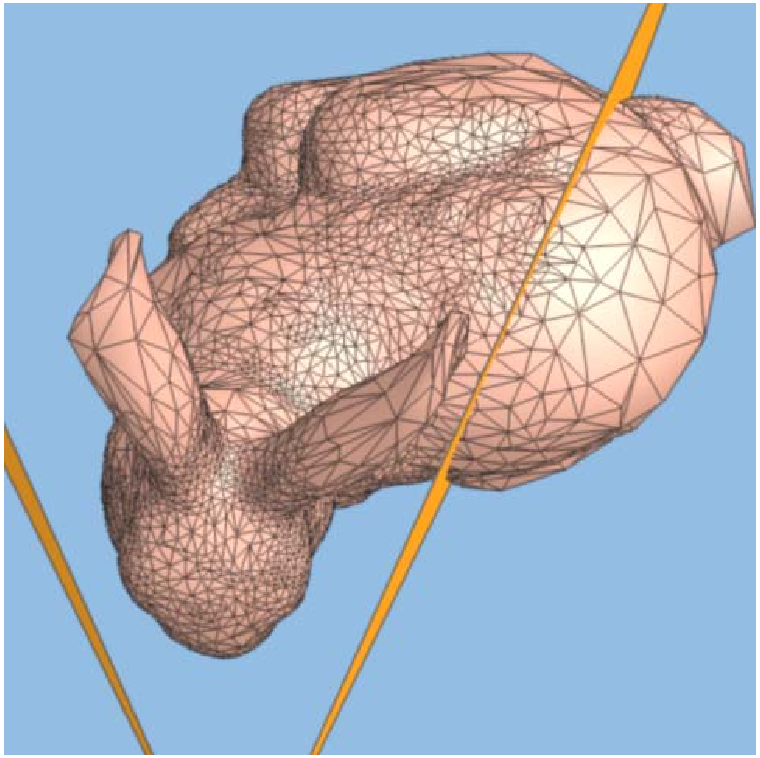
The limit surfaces of Catmull-Clarks's subdivision
have continuous curvature almost everywhere

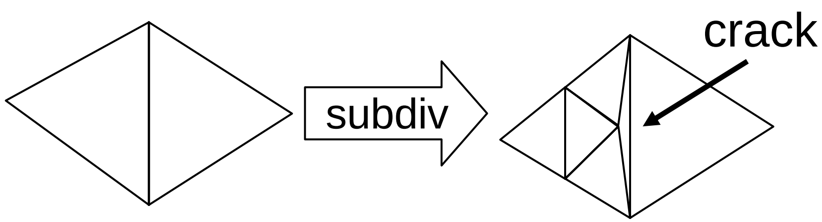
Edges and Creases

- **Most surface are not smooth everywhere**
  - Edges & creases
  - Can be marked in model
    - Weighting is changed to preserve edge or crease
- **Generalization to semi-sharp creases (Pixar)**
  - Controllable sharpness
  - Sharpness (s) = 0, smooth
  - Sharpness (s) = inf, sharp
  - Achievable through hybrid subdivision step
    - Subdivision iff s==0
    - Otherwise parameter is decremented

Adaptive Subdivision

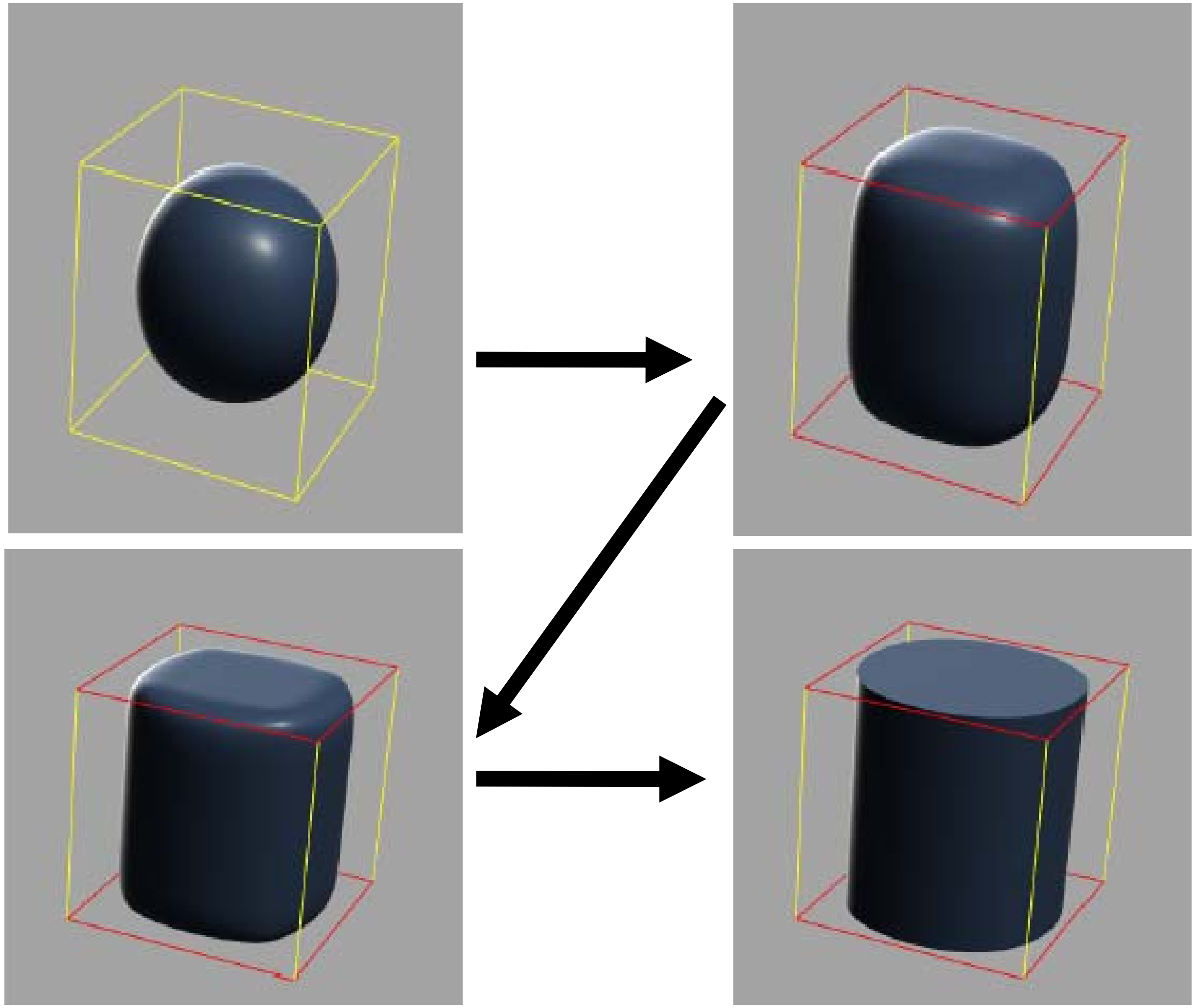
- **Not all regions of a model need to be subdivided.**
- **Idea: Use some criteria and adaptively subdivide mesh where needed.**
  - Curvature
  - Screen size
    - Make triangles < size of pixel
  - View dependence
    - Distance from viewer
    - Silhouettes
    - In view frustum
  - Careful!
    - Must avoid “cracks”





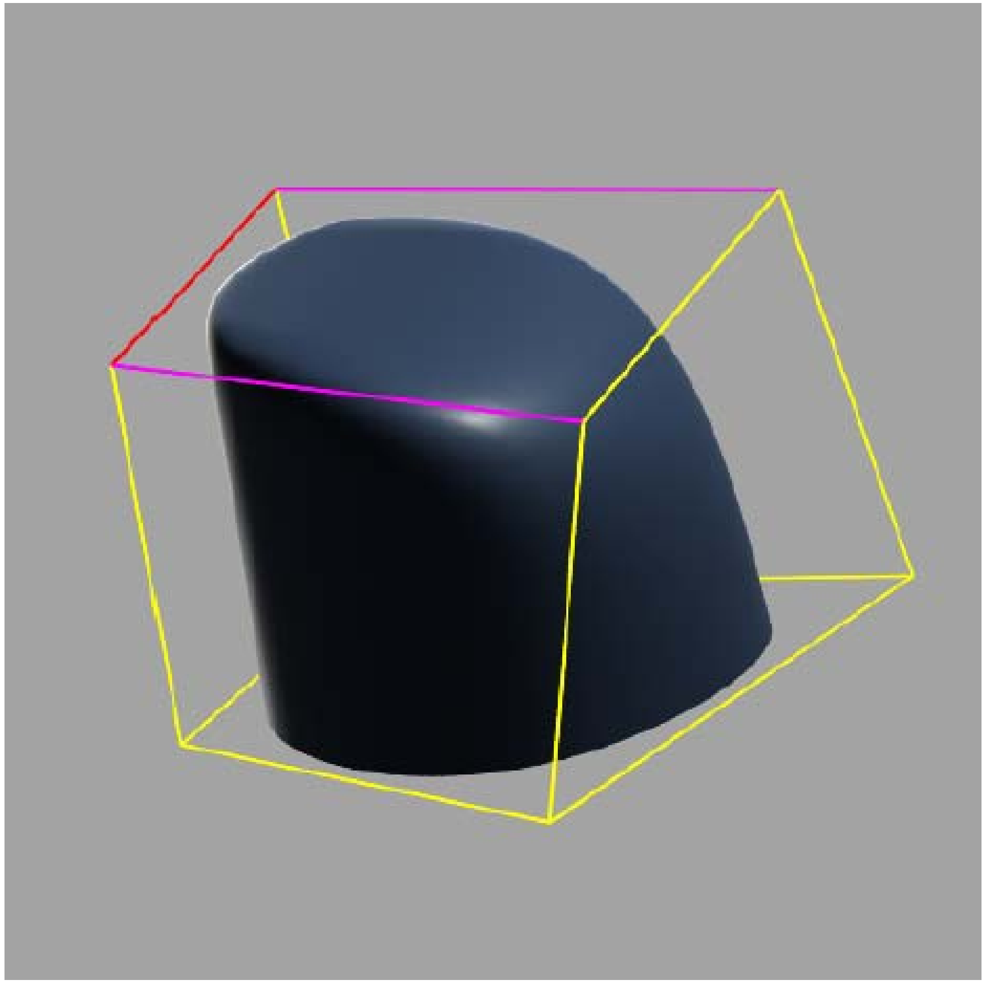
Edges and Creases

- **Increasing sharpness of edges**



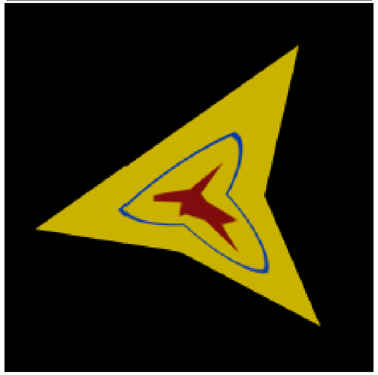
Edges and Creases

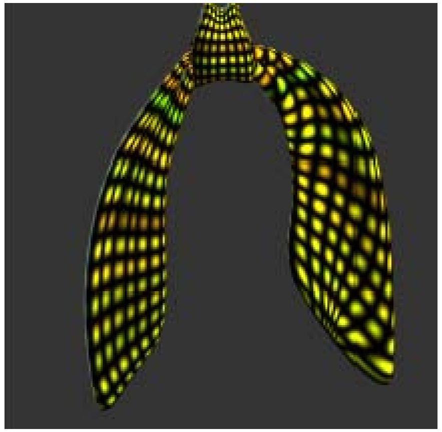
- **Can be changed on a edge by edge basis**



Texture mapping

- **Solid color painting is easy, already defined**
- **Texturing is not so easy**
  - Using polygonal methods can result in distortion
- **Solution**
  - Assign texture coordinates to each original vertex
  - Subdivide them just like geometric coordinates
- **Introduces a smooth scalar field**
  - Used for texturing in Geri's jacket, ears, nostrils





Advanced Topics

- **Hierarchical Modeling**
  - Store offsets to vertices at different levels
  - Offsets performed in normal direction
  - Can change shape at different resolutions while rest stays the same
- **Surface Smoothing**
  - Can perform filtering operations on meshes
    - E.g. (Weigthed) averaging of neighbors
- **Level-of-Detail**
  - Can easily adjust maximum depth for rendering

Wrapup: Subdivision Surfaces

- **Advantages**
  - Simple method for describing complex surfaces
  - Relatively easy to implement
  - Arbitrary topology
  - Local support
  - Guaranteed continuity
  - Multi-resolution
- **Difficulties**
  - Intuitive specification
  - Parameterization
  - Intersections