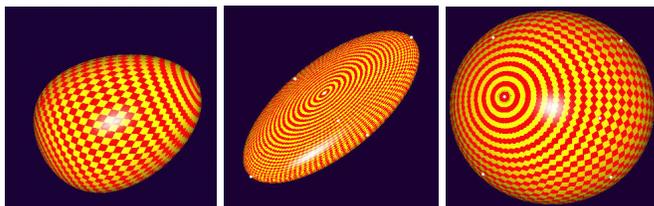


A Primer on Shapes: Curves and Surfaces

Ardy Goshtasby
Alyn Rockwood
Demetri Terzopoulos



SIGGRAPH
2001 EXPLORE INTERACTION
AND DIGITAL IMAGES

A Primer on Shapes: Curves and Surfaces

SIGGRAPH 2001 Course No. 7
Organizer: Ardeshir Goshtasby

August 12, 2001

Description

Shape representations employed for design should enable easy shape specification and modification, while those used for recovery should readily accommodate unstructured data. This course discusses curve and surface representations that are effective for shape design and shape recovery—including Bezier, B-spline, rational Bezier, non-uniform rational B-spline (NURBS), dynamic NURBS, thin-plate spline, and rational Gaussian representations—as well as methods for grouping and parametrizing scattered points. The tutorial will convey technical concepts through imagery rather than formal mathematics; students will be directed to the relevant literature for further study.

Table of Contents/Schedule

1. Introduction – Rockwood (10:00)
2. Curves and Surfaces for Shape Design
 - 2.1. Bezier – Rockwood (10:05)
 - 2.2. B-spline – Rockwood (10:15)
 - 2.3. NURBS – Rockwood (10:25)
 - 2.4. D-NURBS – Terzopoulos (10:45)
 - 2.5. Triangular D-NURBS – Terzopoulos (10:55)
3. Curves and Surfaces for Shape Recovery
 - 3.1. Thin-Plate Splines – Terzopoulos (11:10)
 - 3.2. Rational Gaussian Representation – Goshtasby (11:25)
4. Grouping and Parametrizing Scattered Points – Goshtasby (11:40)
5. Summary – Goshtasby (11:55)

Speaker Biographies

Ardeshir Goshtasby is a professor (9/1/01) in the Department of Computer Science and Engineering at Wright State University. For nearly a decade he has been working on curves and surfaces for design of geometric models as well as curves and surfaces for recovery of free-form shapes from scattered points. The RaG formulation developed by Goshtasby unifies shape design and shape recovery. With the RaG formulation, not only well-known shapes such as circles, spheres, cylinders, and cones can be defined, but complex free-form shapes from scattered points can be represented.

Alyn Rockwood received a Ph.D. from the Department of Applied Mathematics and Theoretical Physics, Cambridge University. He has spent 25 years in industrial and academic research, including positions at SGI, where he developed the NURBS rendering methods for GL/OpenGL; at Evans and Sutherland, where he worked on the first hardware textured graphics system (used in flight simulation); at Shape Data Ltd., where he developed the first commercial automatic blending methods in CAD/CAM; and more recently at Arizona State University, where he was a faculty member and project co-director for a major research project in brain imaging. He has authored several books and 50 articles on computer graphics and also served recently as the 1999 SIGGRAPH paper's chair.

Demetri Terzopoulos holds the Lucy and Henry Moses Professorship in the Sciences at New York University and is Professor of Computer Science and Mathematics at NYU's Courant Institute. He is currently on leave from the University of Toronto where he is Professor of Computer Science and Professor of Electrical and Computer Engineering. He graduated from McGill and received the PhD degree from MIT. He was elected a Fellow of the IEEE, a Killam Research Fellow of the Canada Council for the Arts, an EWR Steacie Memorial Fellow of the Natural Sciences and Engineering Research Council of Canada, and an AI and Robotics Fellow of the Canadian Institute for Advanced Research. Among his many awards are computer graphics honors from Ars Electronica and the International Digital Media Foundation for his work on artificial animals and from NICOGRAPH for his work on facial modeling and animation.



A Primer on Shapes: Curves and Surfaces

Alyn Rockwood
Demetri Terzopoulos
Ardeshir Goshtasby

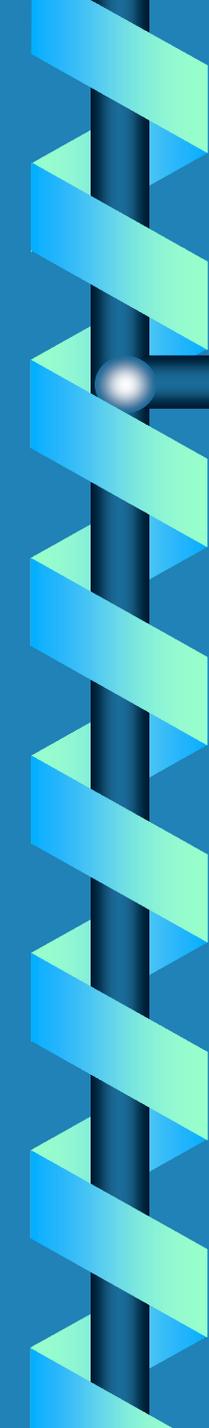


Who is Alyn Rockwood?

- Ph.D. Cambridge University
- Industrial Research: ES, SGI
- Stints in Academics: BYU, ASU

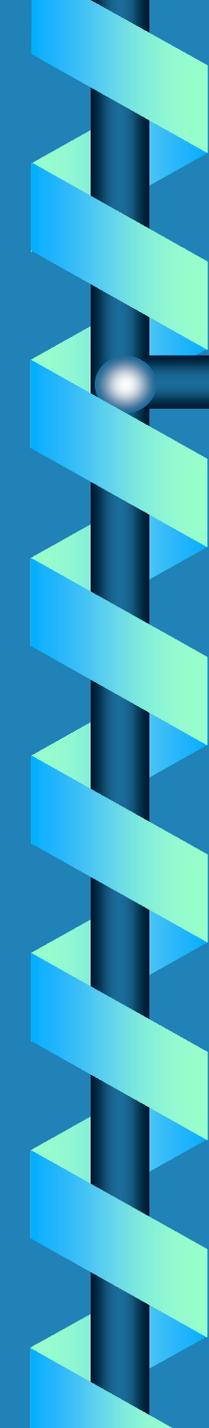
Who is Demetri Terzopoulos?

- Ph.D. MIT
- Lucy and Henry Moses Professor in Sciences and Professor of Computer Science and Mathematics at NYU
- Professor of Computer Science and Professor of Electrical and Computer Engineering at U. Toronto



Who is Ardy Goshtasby?

- Ph.D. Michigan State University
- Professor of Computer Science and Engineering (9/1/01), Wright State U.
- Developed the RaG Curve and Surface Formulation



Course Contents

- **Curves and Surfaces for Shape Design**
 - Bezier, B-Spline, NURBS
 - Design Issues
 - D-NURBS, Triangular D-NURBS
- **Curves and Surfaces for Shape Recovery**
 - Thin-Plate Splines
 - Rational Gaussian (RaG) Representations
- **Grouping and Parametrizing Irregularly Spaced Points**

Visual Vocabulary

Terms

Use



For Example:

Turbo Charged
Engine



Don't need to be
a mechanical
engineer to drive
a car

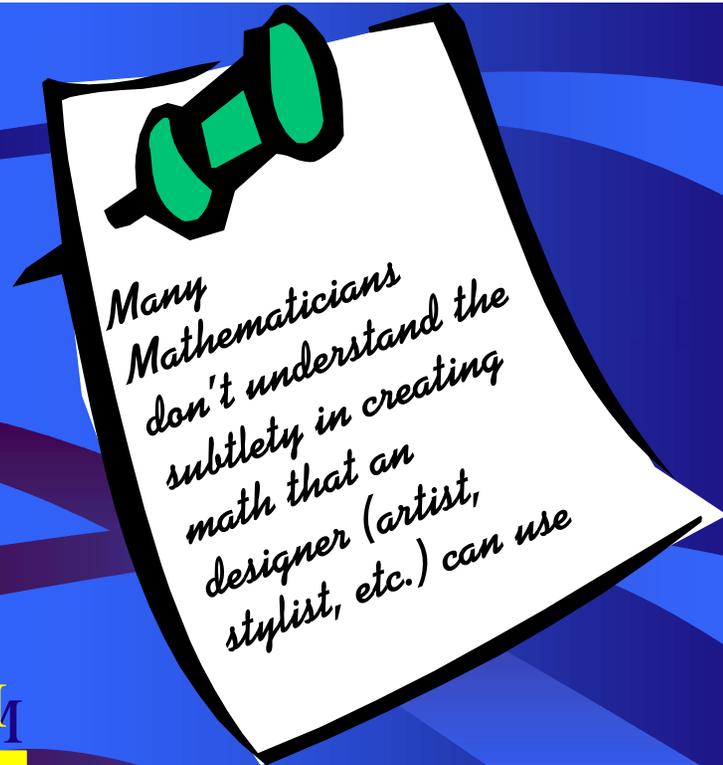
Obstacle



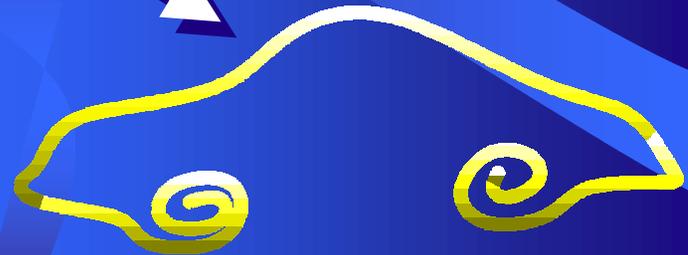
Designers

$$B(t) = \sum_{i=0}^M d_i N_i^n(t)$$

Math

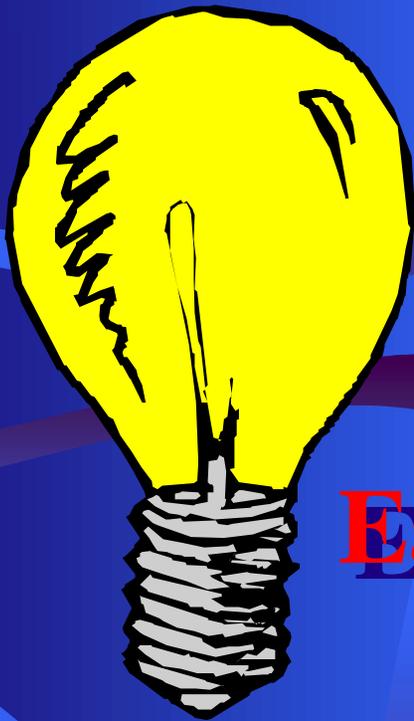


Many
Mathematicians
don't understand the
subtlety in creating
math that an
designer (artist,
stylist, etc.) can use



Curves

Design

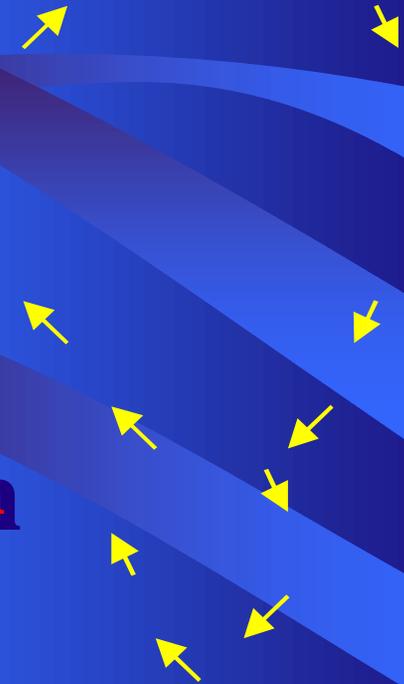


Idea

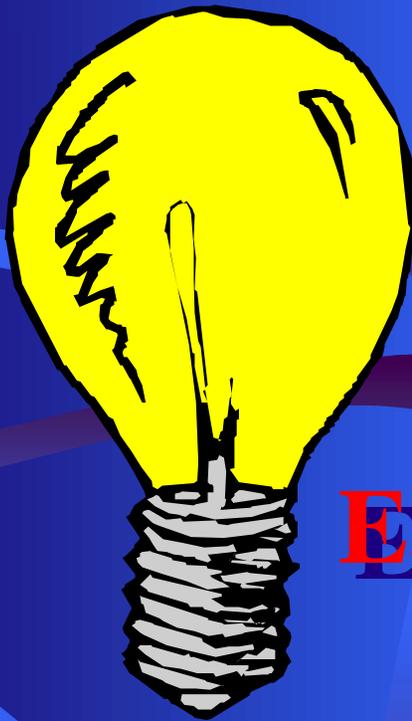


Easy Modification
(no Math)

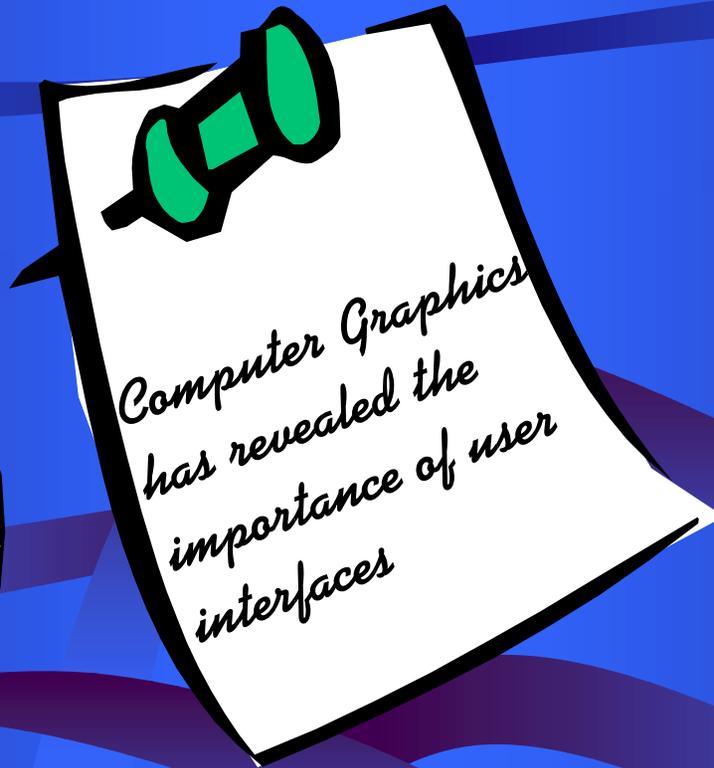
Simple inputs



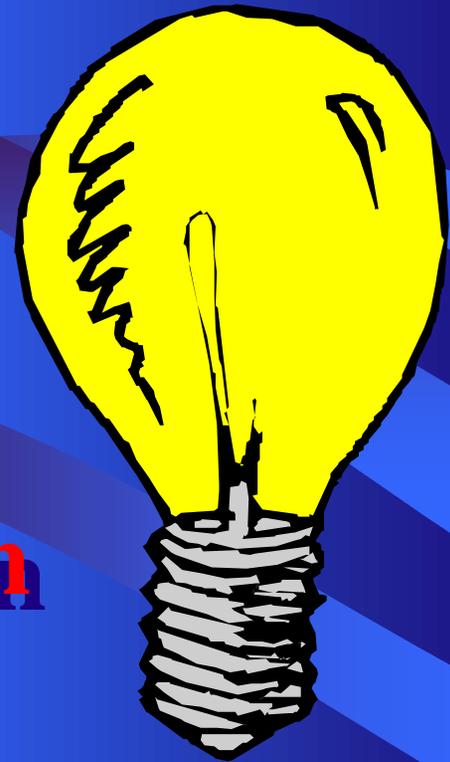
Design



Idea



Easy Modification
(no Math)



Designed Object

Curve Types

∞ Parametric



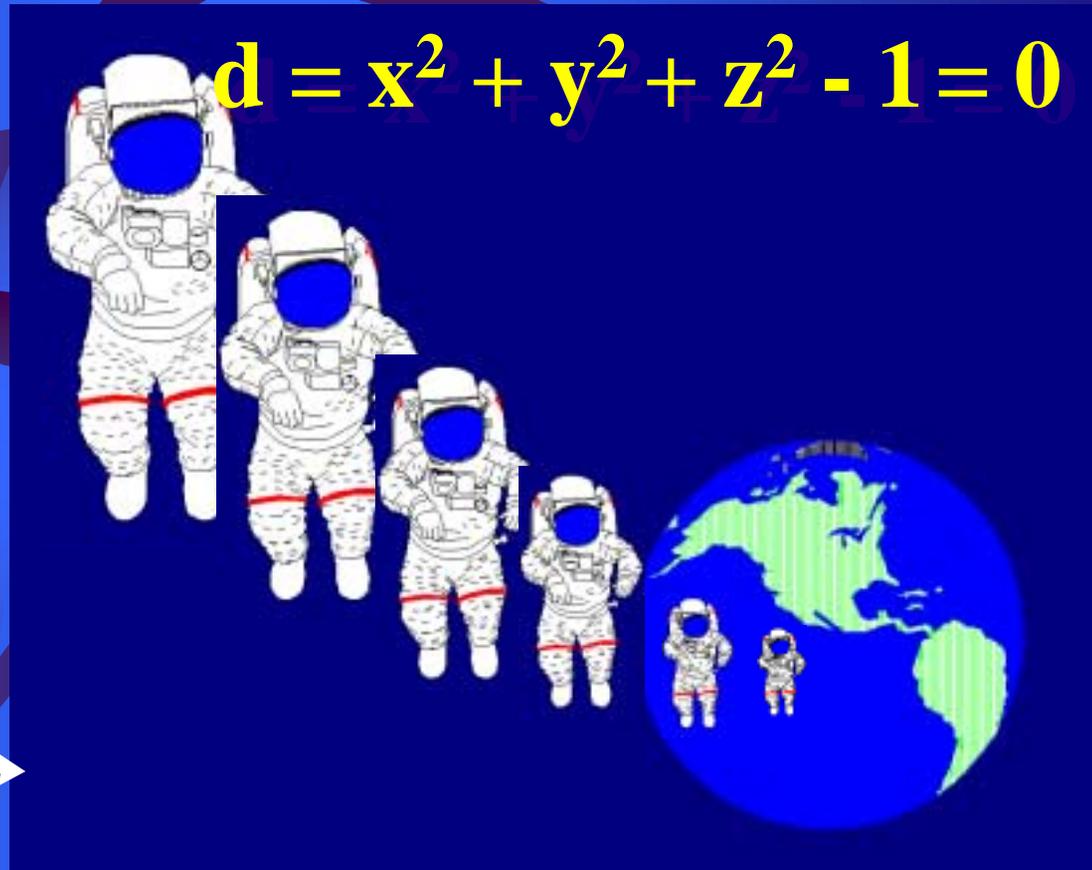
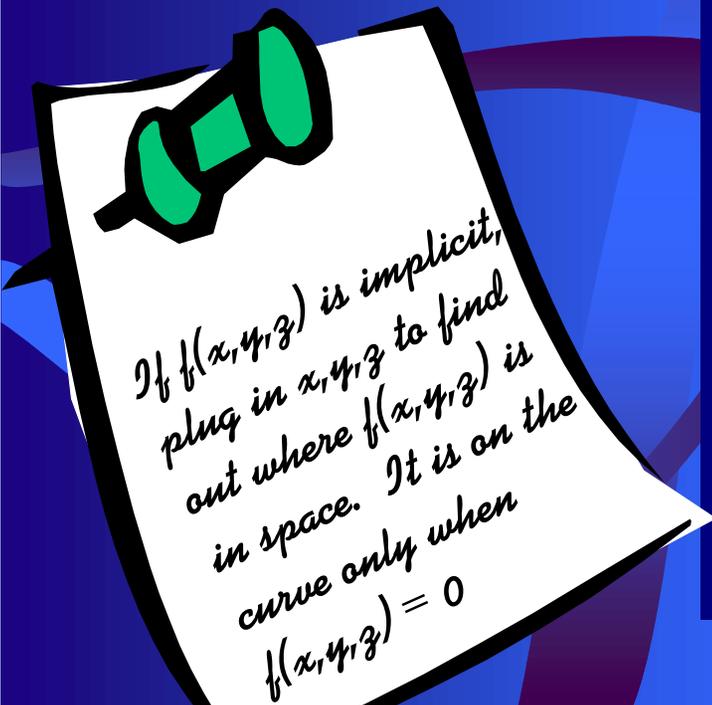
If $f(t)$ is a "parametric", plug in t to find out where $f(t)$ is on the curve

Where is the fly at time t ?
 $t = \hat{a}$ and
(t is a parameter)

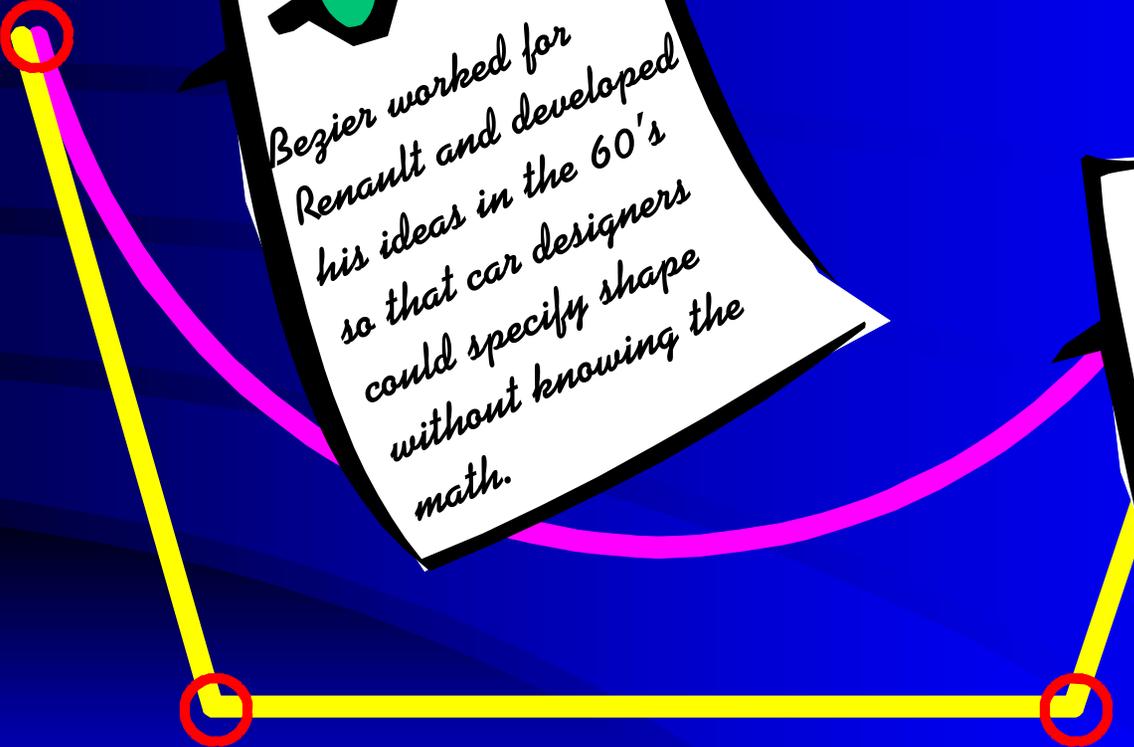
Curve Types

∞ Implicit

How Far?



Monsieur Bezier's Curve

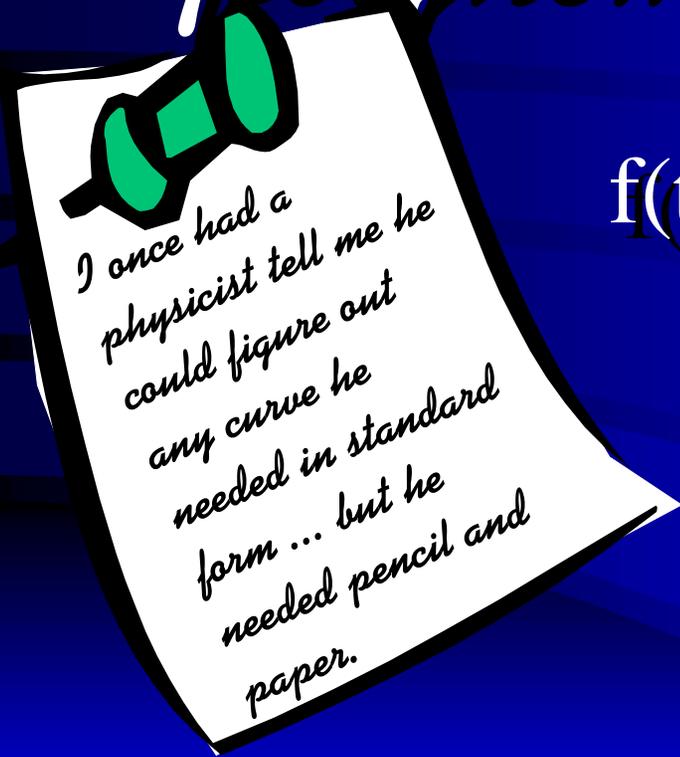


Bezier worked for Renault and developed his ideas in the 60's so that car designers could specify shape without knowing the math.

Quiz:

What are the characteristics that make this curve valuable for design?

What's Wrong with Standard polynomials for Design?

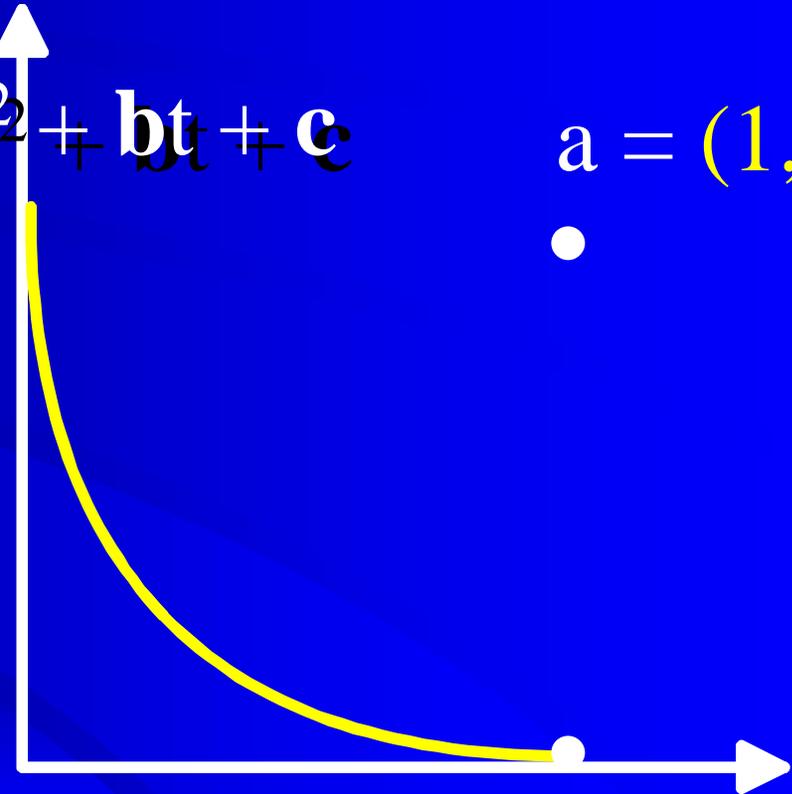


$$f(t) = at^2 + bt + c$$

$$a = (1,1)$$

$$b = (-2,0)$$

$$c = (1,0)$$



What's Wrong with Standard polynomials for Design?

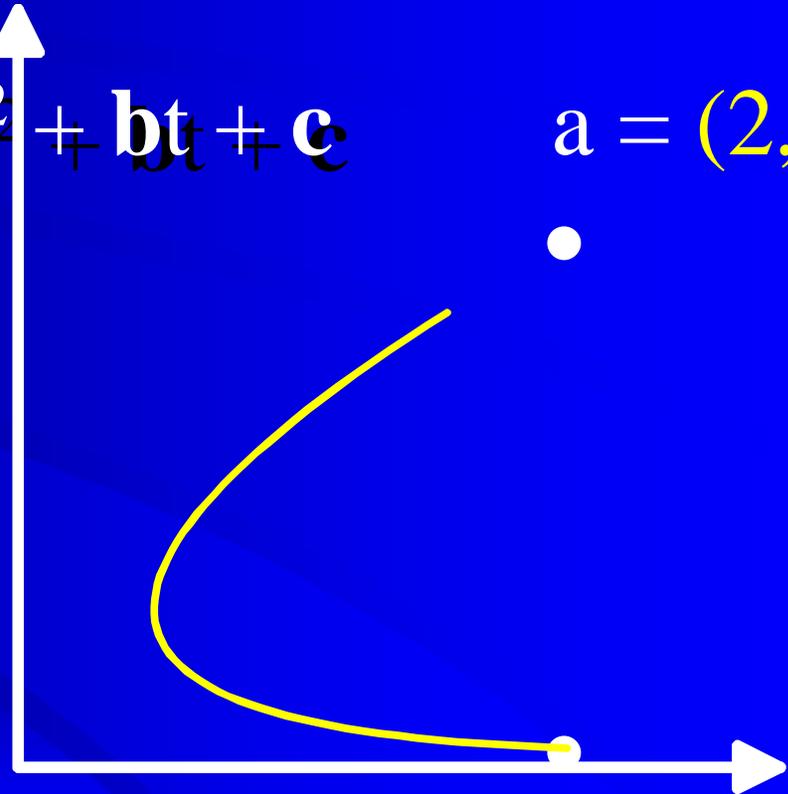
$$f(t) = at^2 + bt + c$$

$$a = (2, 1)$$

- $b = (-2, -2)$

What are a, b, and c now?

$$c = (1, 0)$$



Monsieur Bezier's Solution

$$f(t) = \begin{pmatrix} 1 \\ 1 \end{pmatrix} t^2 + \begin{pmatrix} -2 \\ 0 \end{pmatrix} t + \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

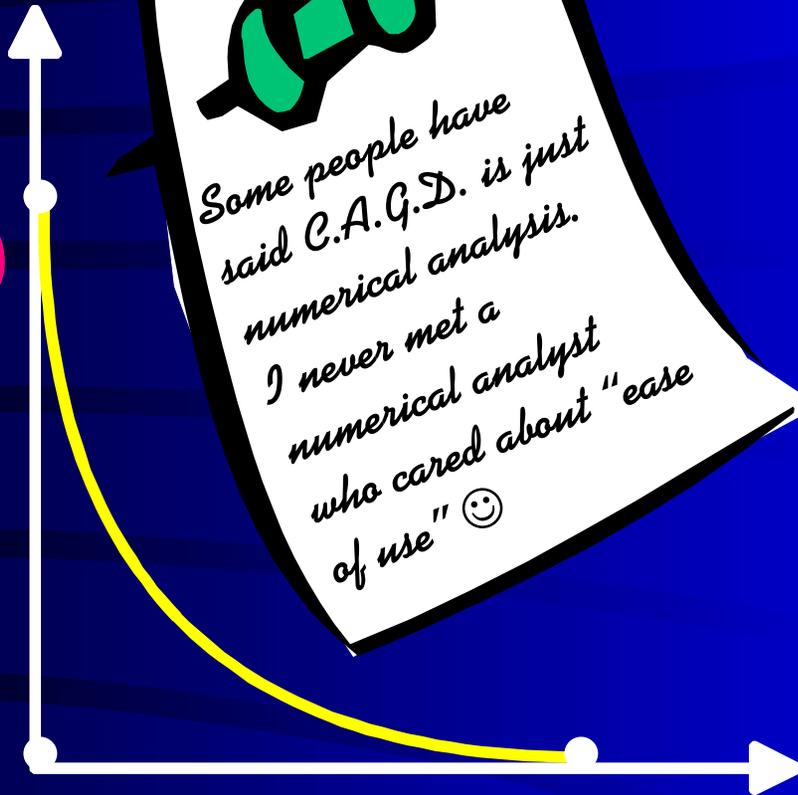
$$= \begin{pmatrix} 1 - 2t + t^2 \\ t^2 \end{pmatrix}$$

$$= \begin{pmatrix} 1 - t^2 \\ t^2 \end{pmatrix}$$

$$= \begin{pmatrix} 1 \\ 0 \end{pmatrix} 1 - t^2 + \begin{pmatrix} 0 \\ 0 \end{pmatrix} 2t(1 - t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} t^2$$

Only Math

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$



$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Some people have said C.A.G.D. is just numerical analysis. I never met a numerical analyst who cared about "ease of use" 😊

What does CAGD stand for?

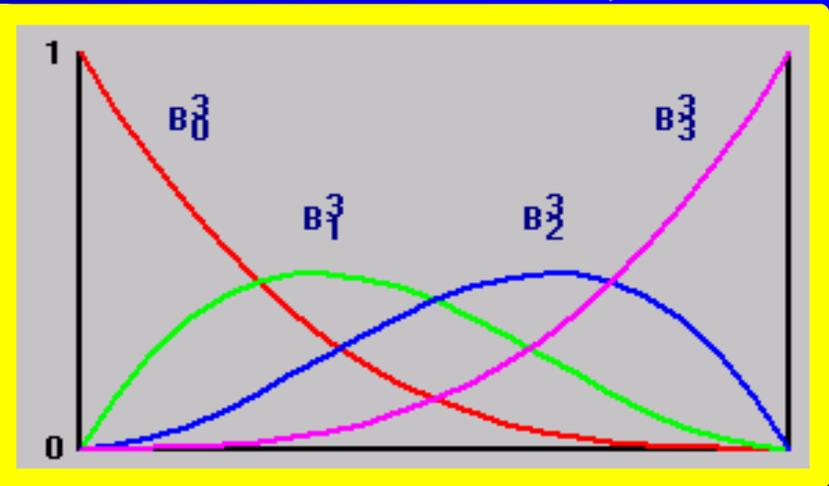
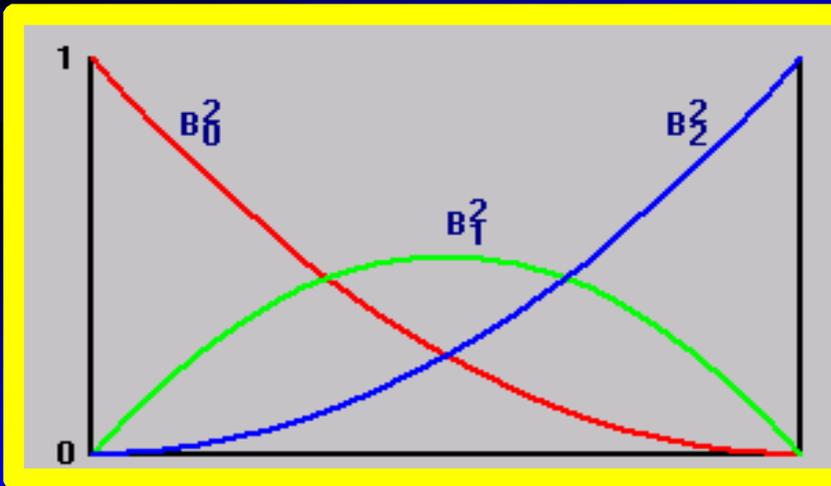
*Coefficients mean something,
What?*

Special Form

“Bernstein” Polynomials

$$\{(1 - t^2), 2t(1 - t), t^2\}$$

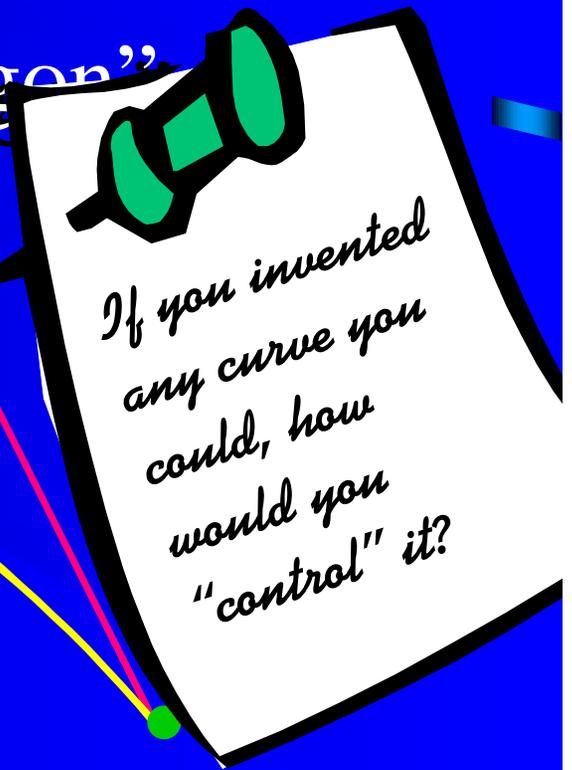
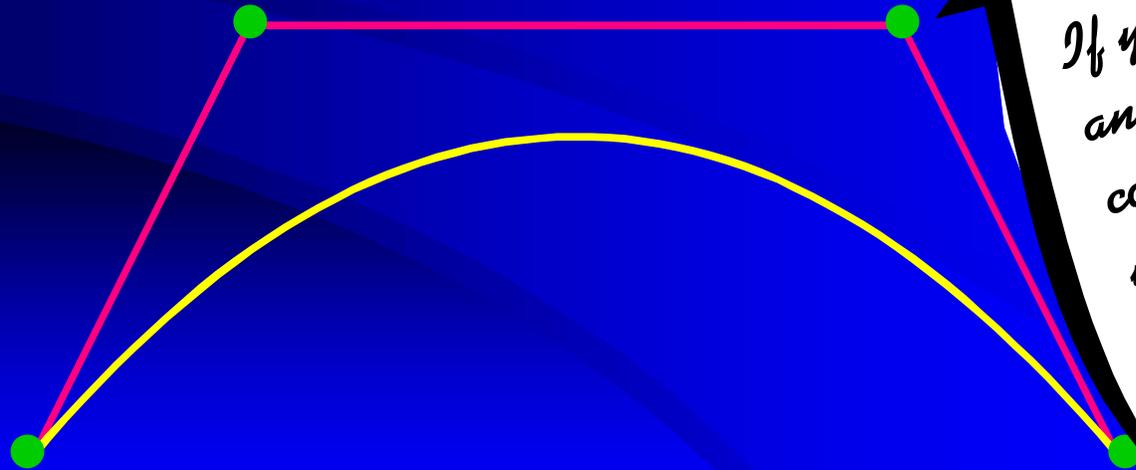
Also called “basis” functions, because they are the mathematical basis from which properties come, e.g. When $t=0$ (left side all functions are 0 except $1 \Rightarrow$ endpoint interpolation



Great with Properties

Coefficients \equiv "Control Points"

Polygon \equiv "Control Polygon"



What does a Bezier curve do?

The "Convex Hull" is the tightest polygon that contains the points. The curve never strays outside the Hull

Imagine if the curve wiggled an undetermined number of times. It doesn't. It is limited by wiggles in control polygon

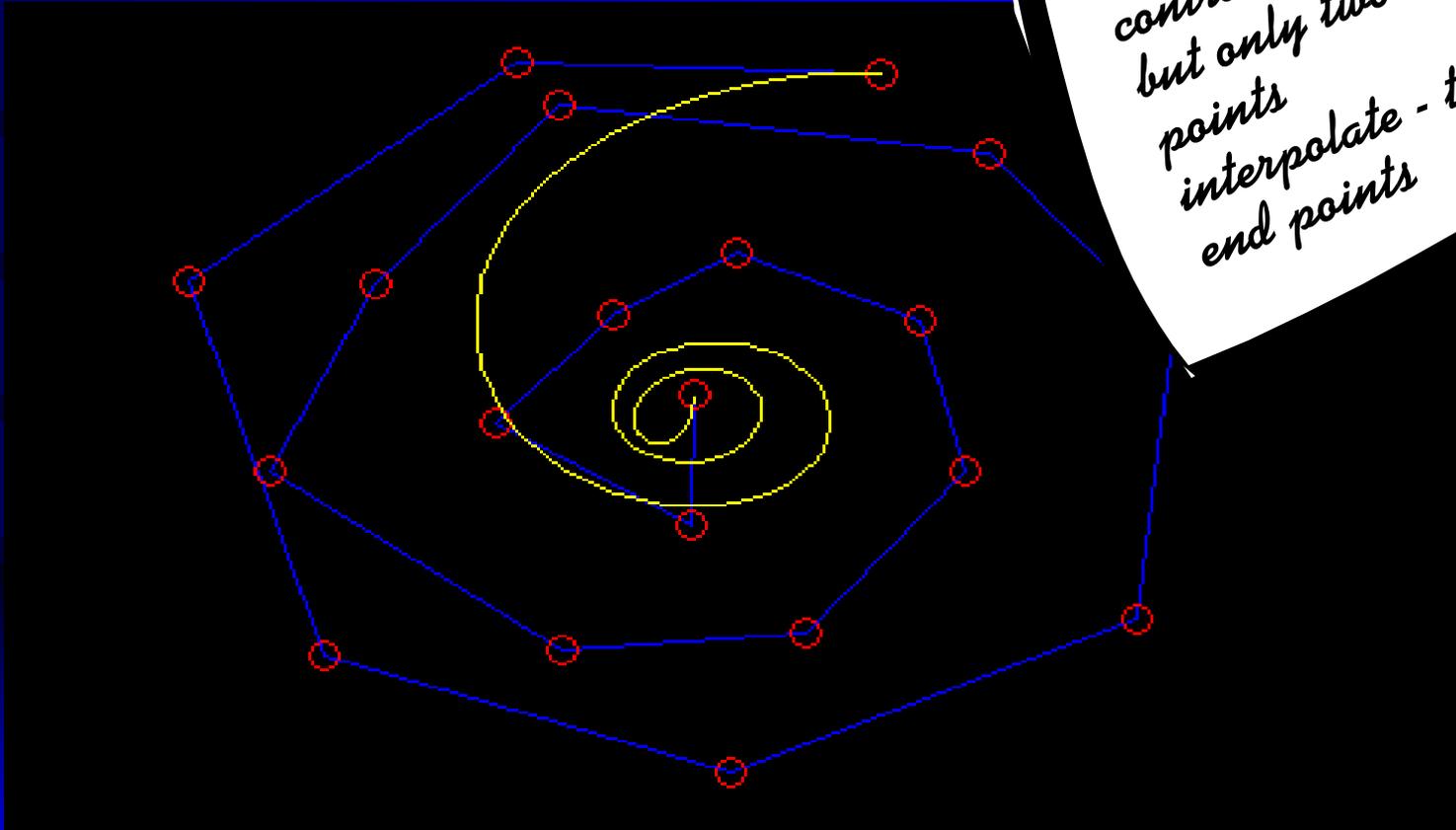
Affine invariance means one only need to map control points (translate, scale, rotate) and the resulting curve is the same as mapping the whole curve

If all control points lie in a line, then the curve does too

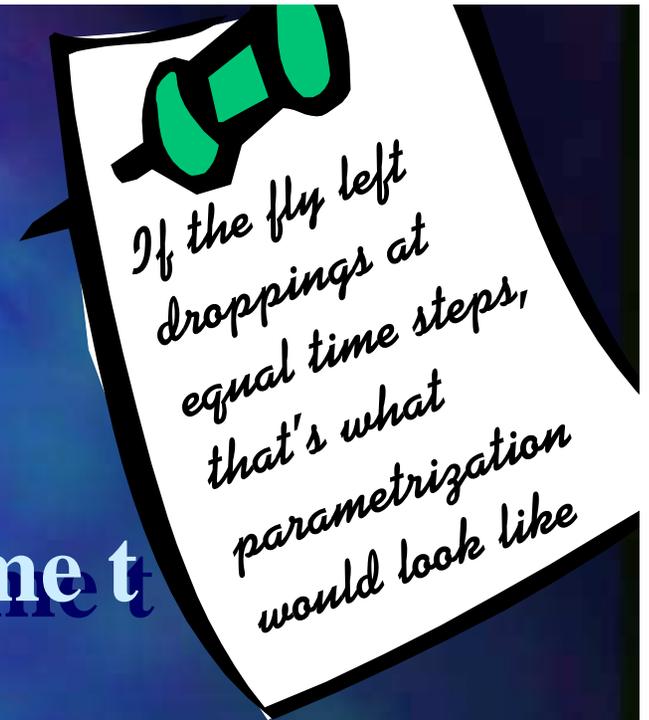
Higher degree



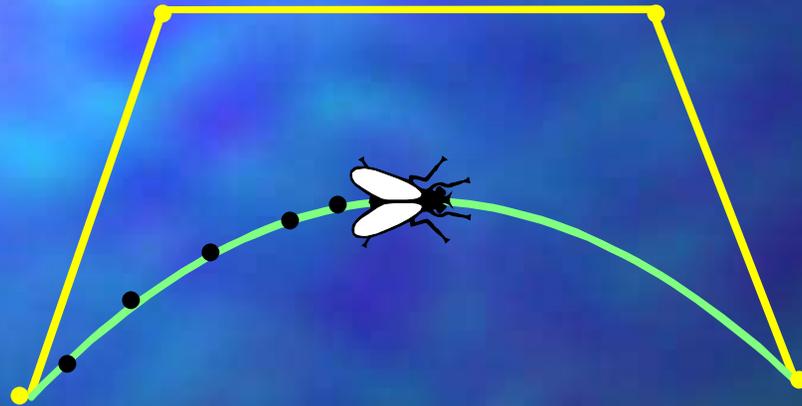
Higher degree
means more
control points,
but only two
points
interpolate - the
end points



Evaluation

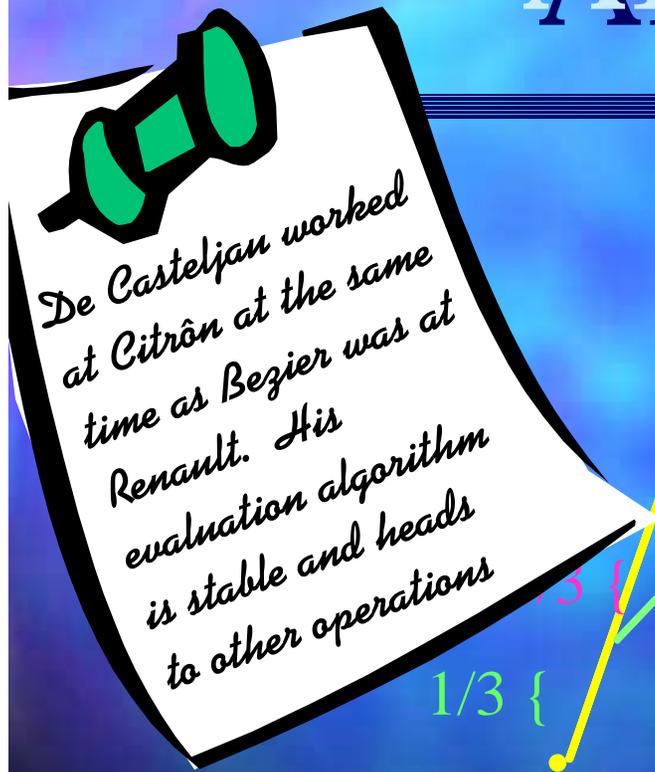


Where is the fly at time t

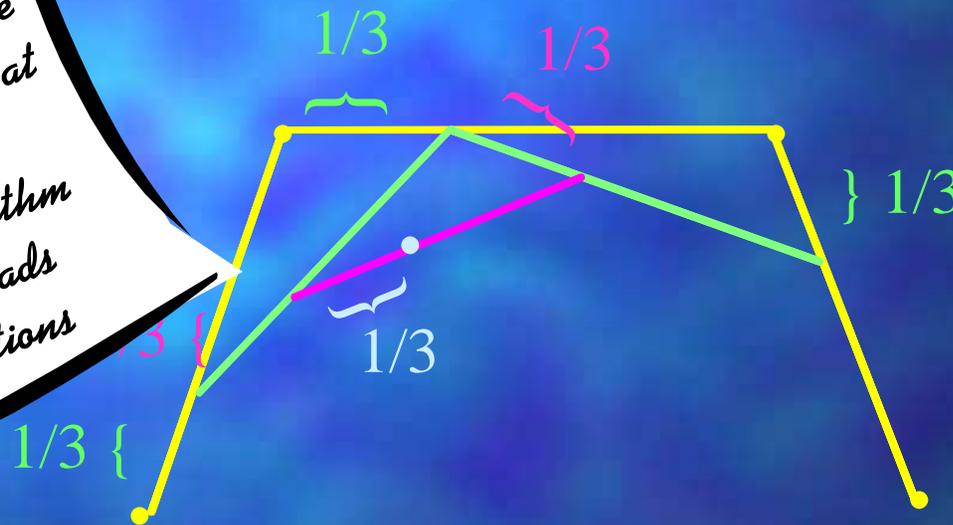


Also called “parametrization”!

Monsieur de Casteljau's Algorithm



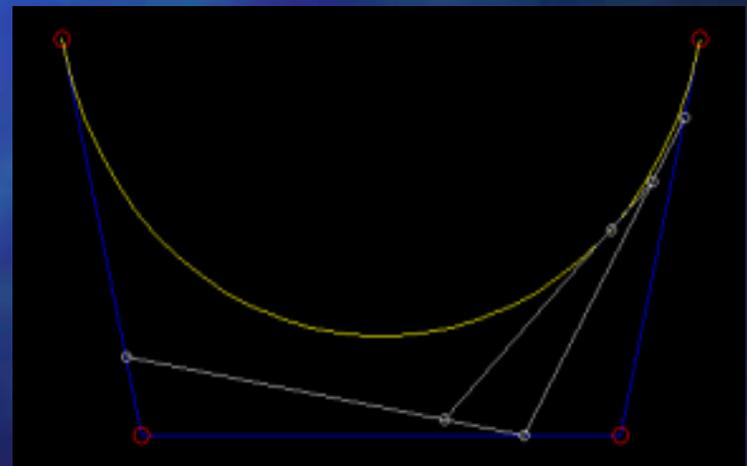
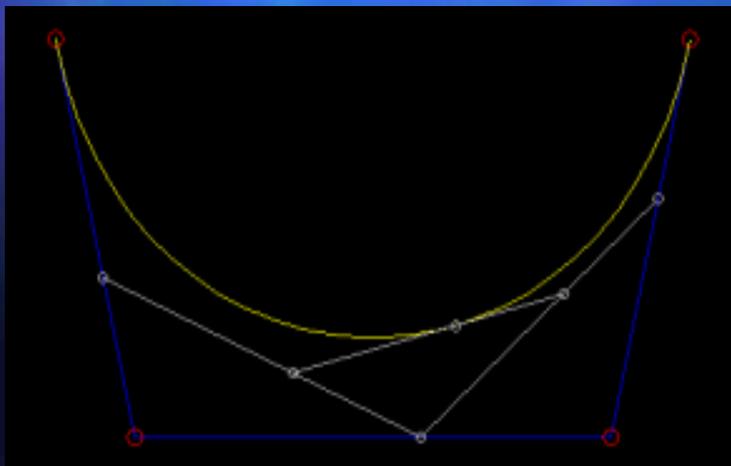
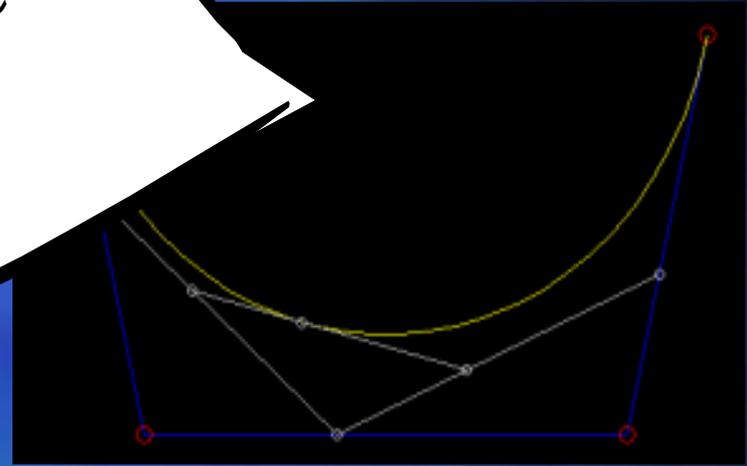
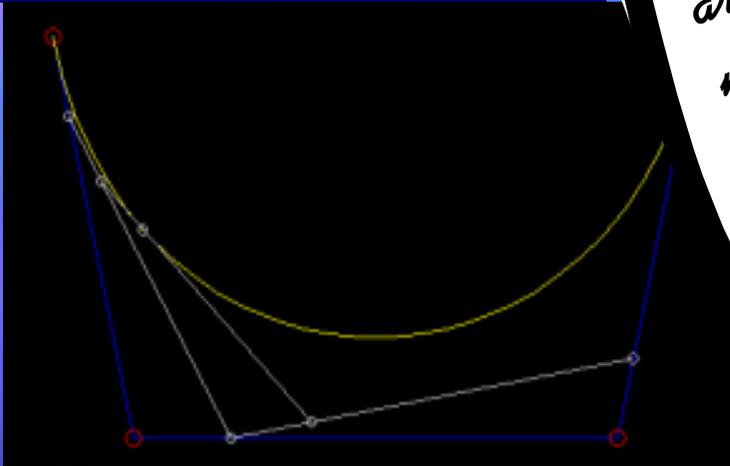
Pick t , say $t = 1/3$



Control Polygon

Final point is on the curve

It is like string art (sort of) or mechanical sliders



Quadratic

$$t = 1/2$$



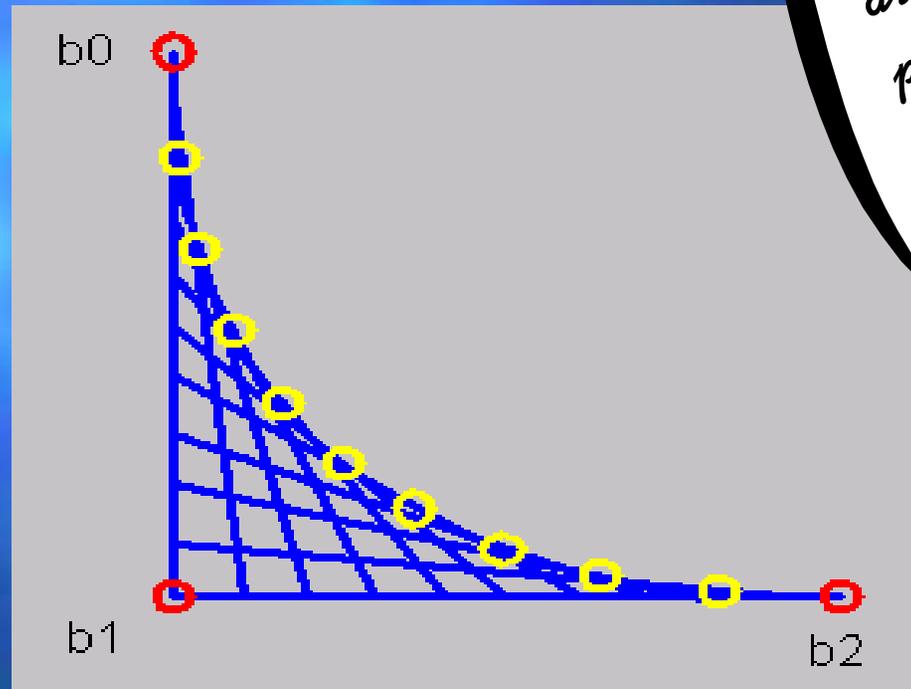
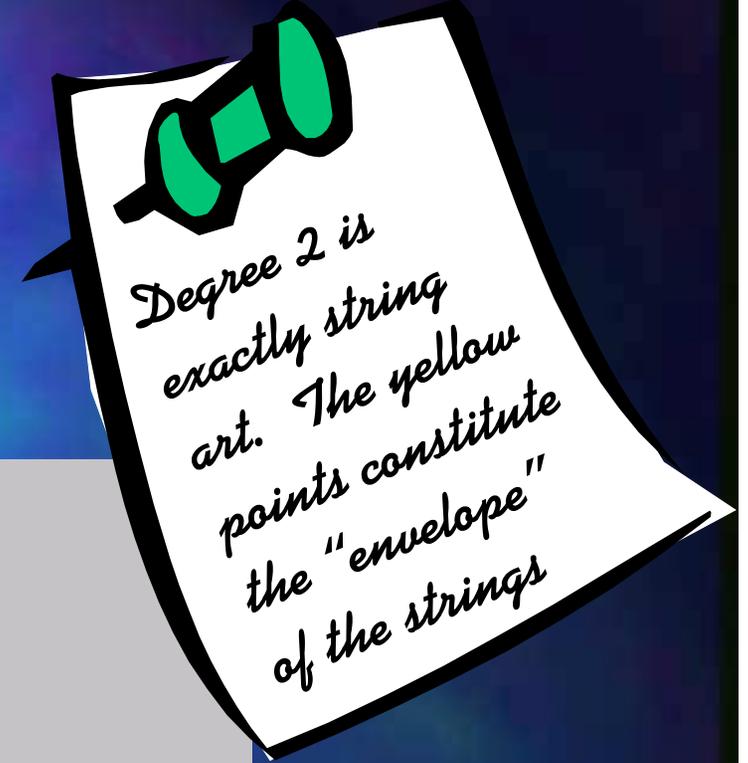
$1/2$ {



Point on curve

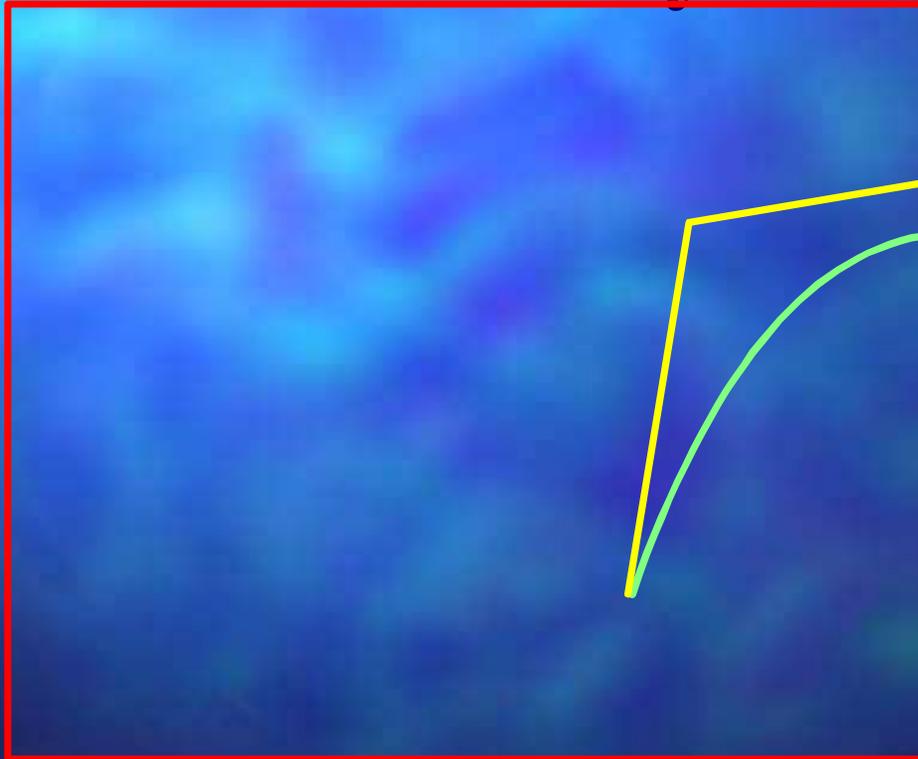
}
 $1/2$

String Art



Subdivision

Why?

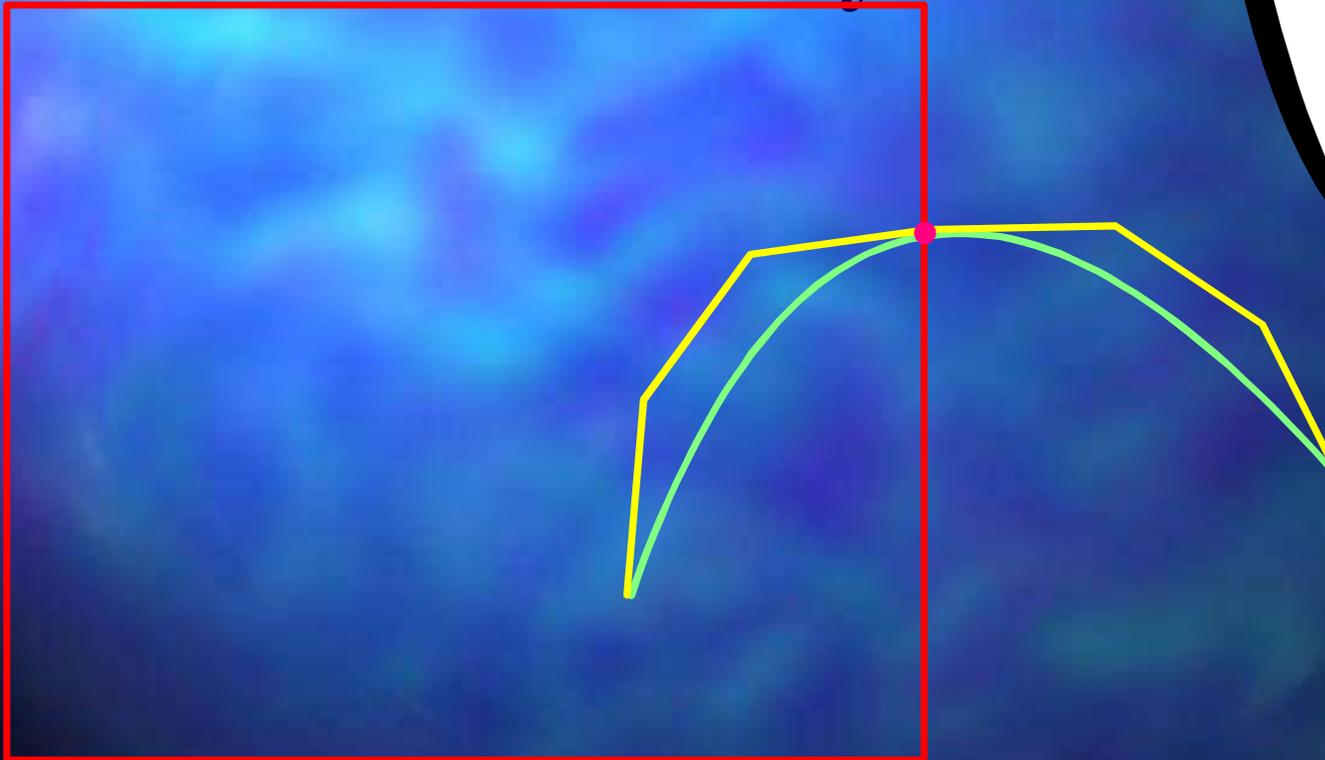


Clip Bezier curve

*Divide & Conquer -
a basic algorithm in
all computer science
including graphics.
The de Casteljau
algorithm provides
subdivision for
Bezier curves*

Subdivision

Why?



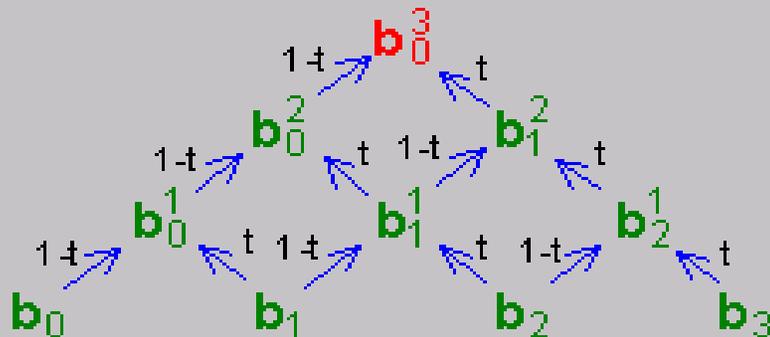
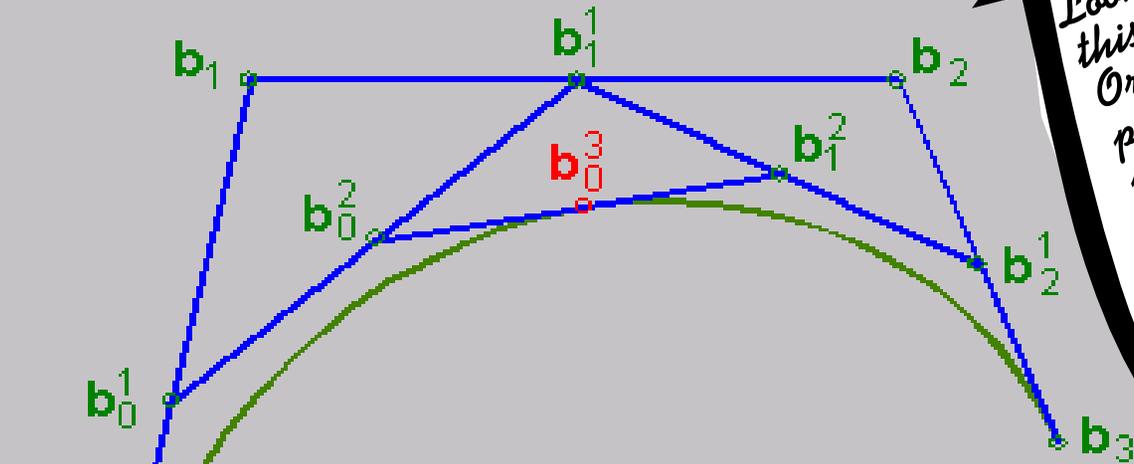
The two new curves exactly match the old curve. We throw away the part outside the window

Divide at window boundary

Use de Casteljano

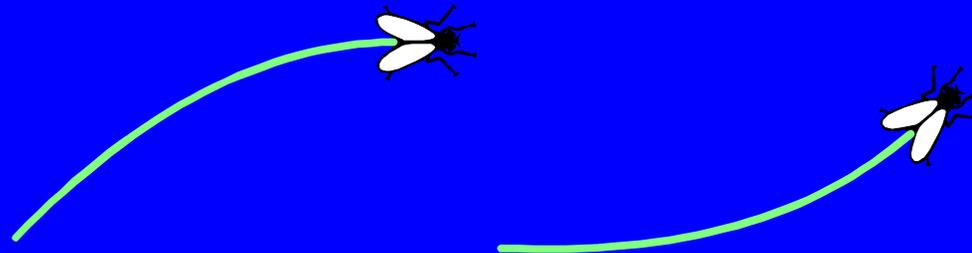


Look carefully at this scheme.
Original control points - no superscripts.
What are control points for left and right subdivided curve? What is their relation to triangle array?

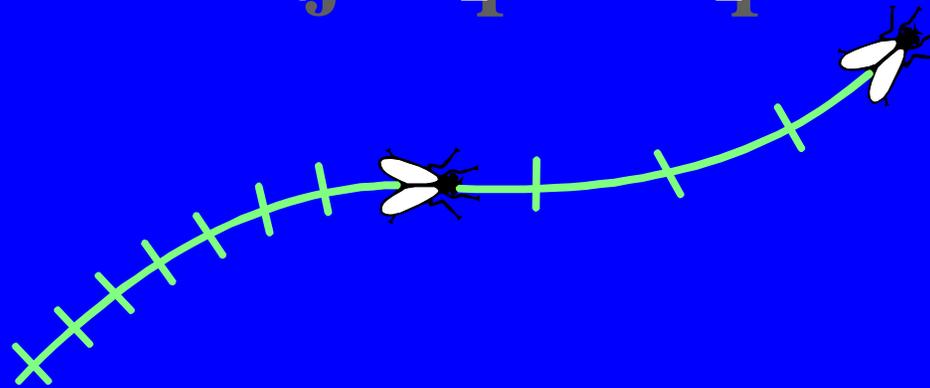


Continuity - Traditional

“ C^0 ” = no breaks like this



“ C^1 ” = no jumps in speed like this



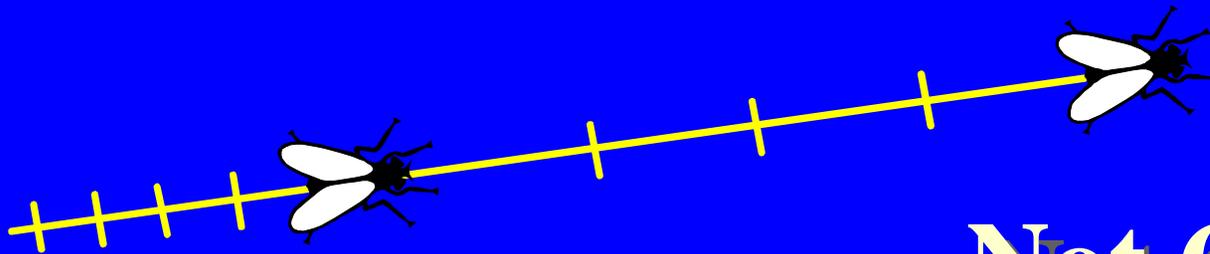
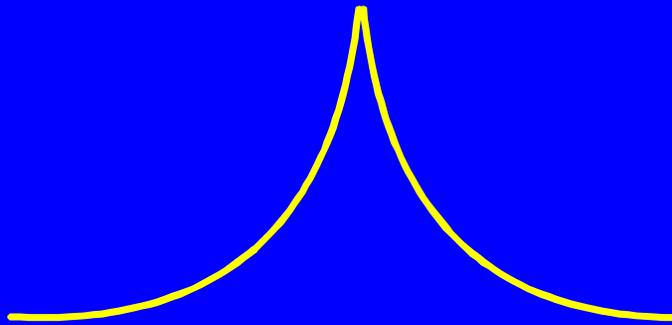
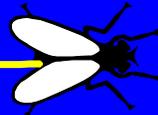
Traditional concepts of continuity are messed up for parametric curves.

Problem

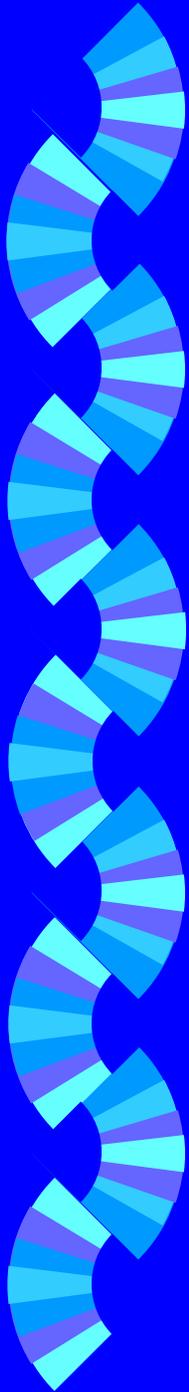


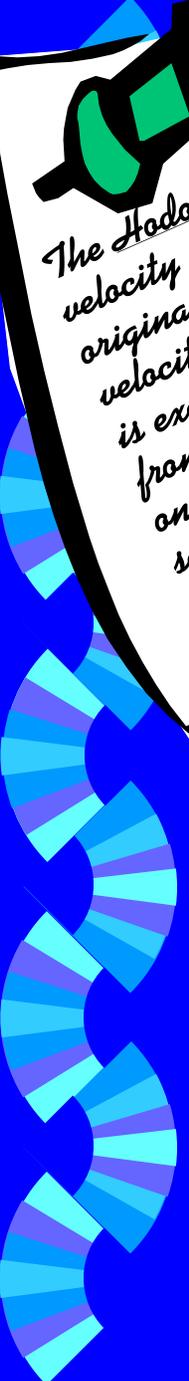
If I drive my car to a stop, then change wheel direction and drive away is my speed continuous? My tracks in the snow? (I live in Colorado)

C^1 (?)

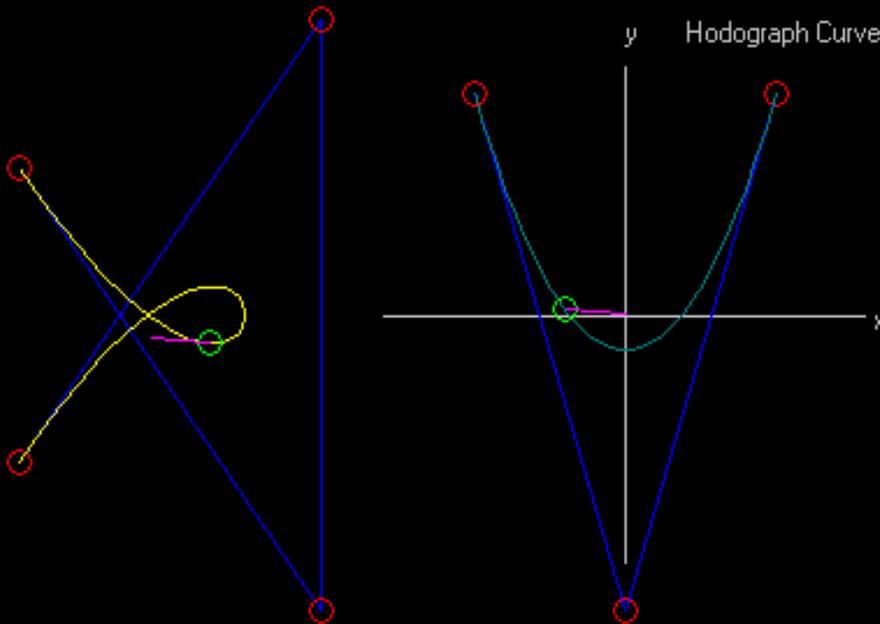


Not C^1 (?)





The Hodograph is the velocity curve of the original. Note the velocity vector on left is exactly the vector from origin to point on hodograph for same parameter. What happens if curve kinks?



$G^1 =$ geometric continuity

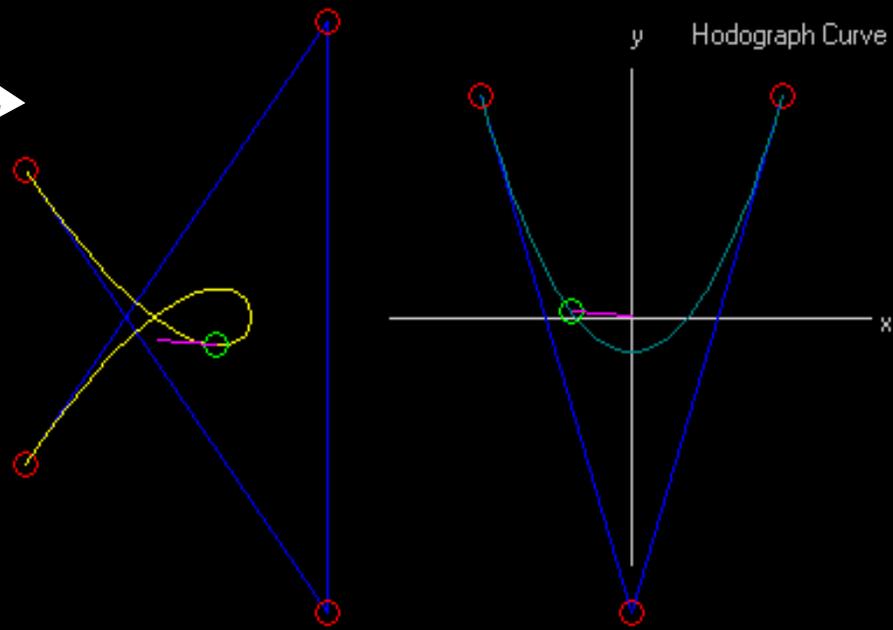
(Continuous direction)

Not G^1

Alternate notion for continuity is concerned with direction of curve tangent, not length. It corresponds better to our idea of smoothness

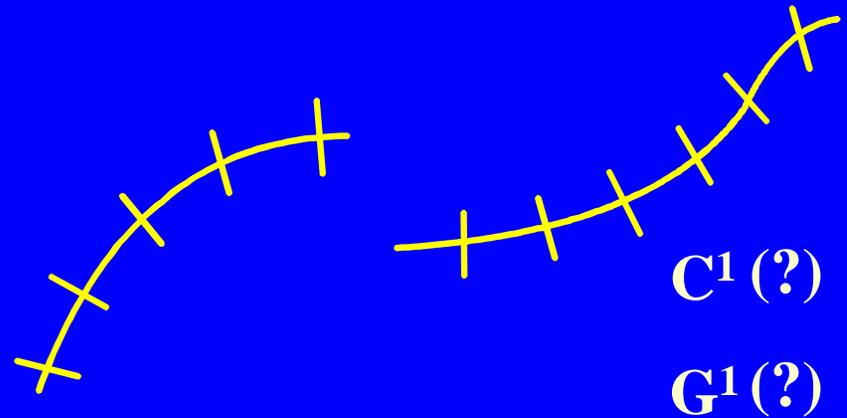
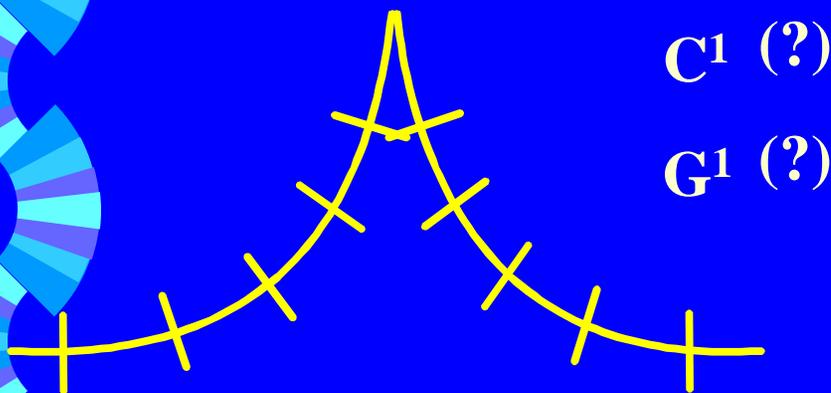
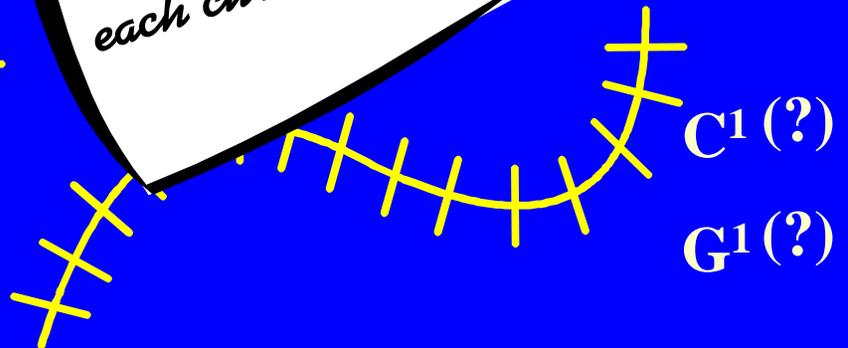
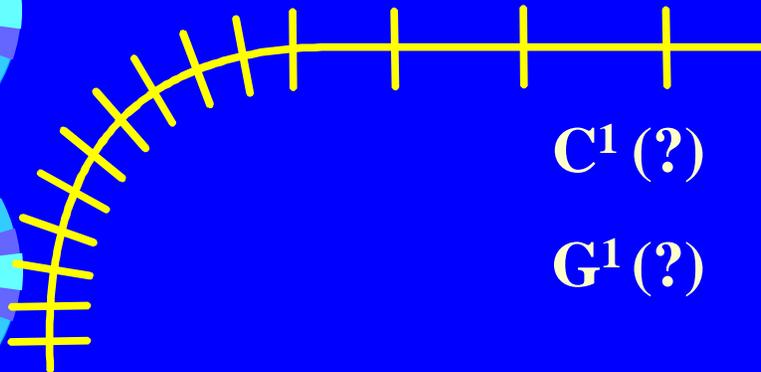
G^1
Abrupt change of speed, but curve is smooth

The Hodograph is
itself a Bezier curve
of one degree less.
How many control
points does this one
have compared to
original curve.



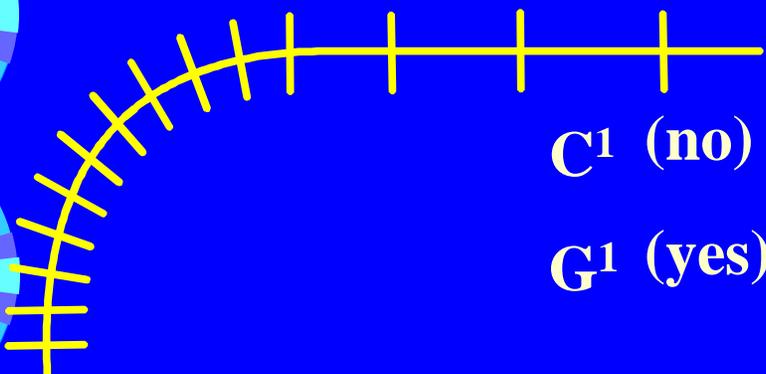
Examples

Can you fill in questions for each curve?



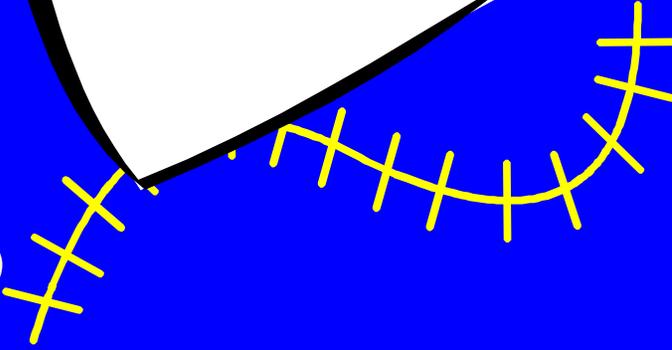
Examples

How did you do?



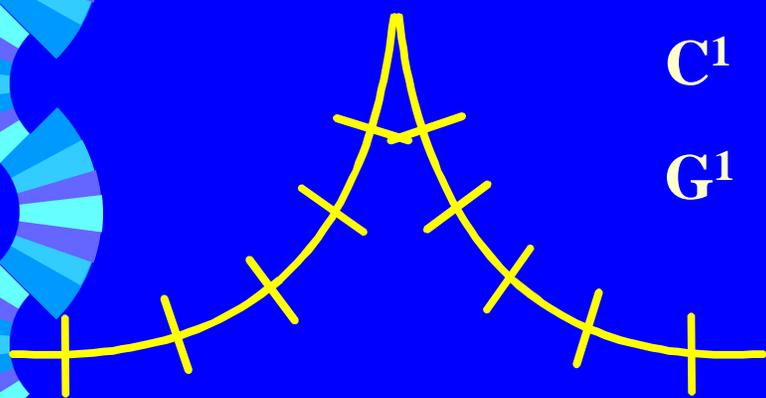
C1 (no)

G1 (yes)



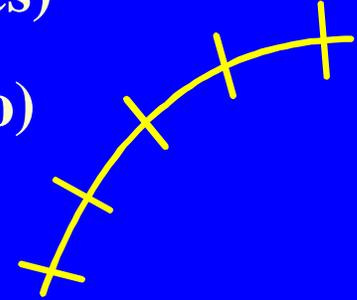
C1 (yes)

G1 (yes)



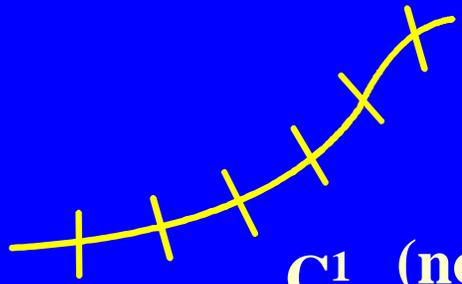
C1 (yes)

G1 (no)



C1 (no)

G1 (no)



Bezier curves with G^1

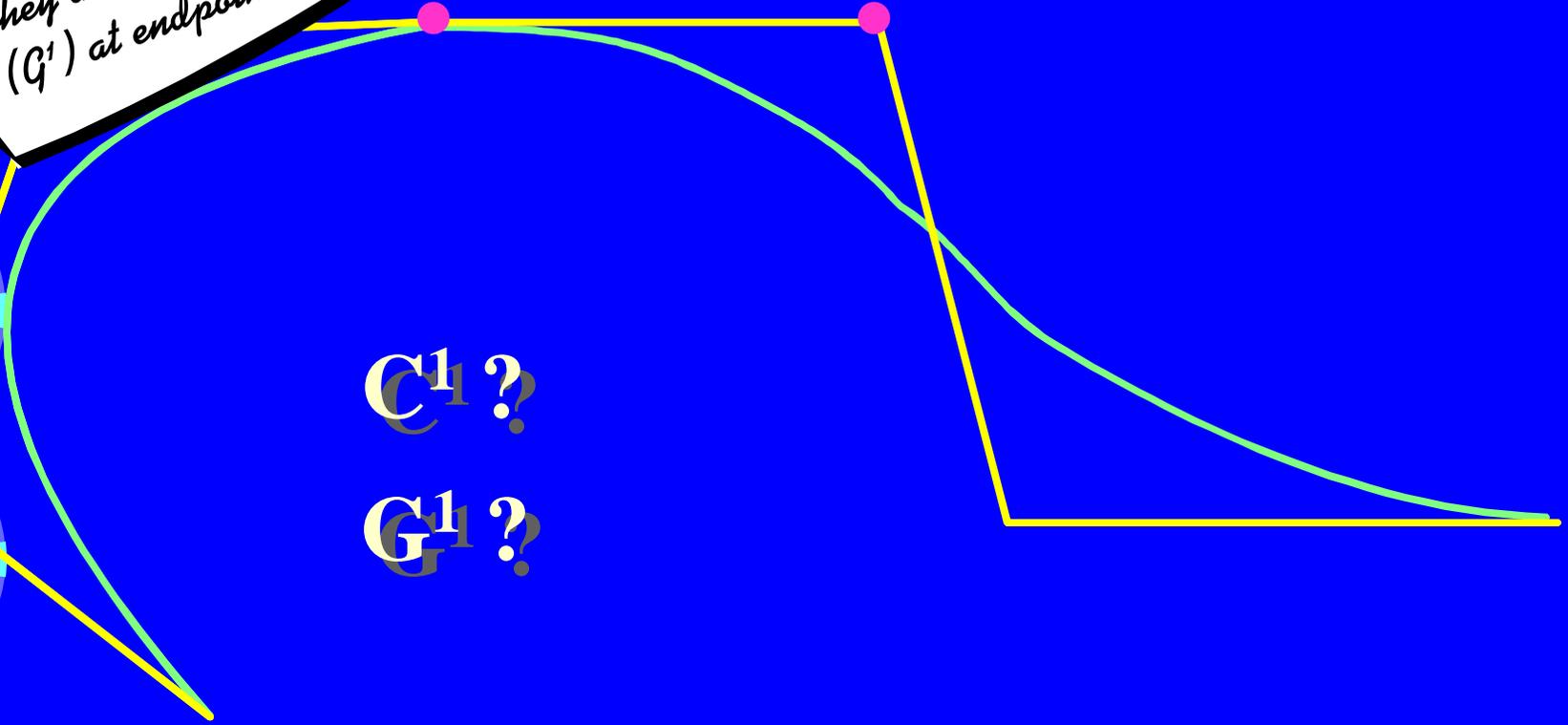
This is where
Bezier gets utility -
joins them together -
for complex designs.
How do you ensure
they are smooth
(G^1) at endpoints?

?

← Same direction

C^1 ?

G^1 ?



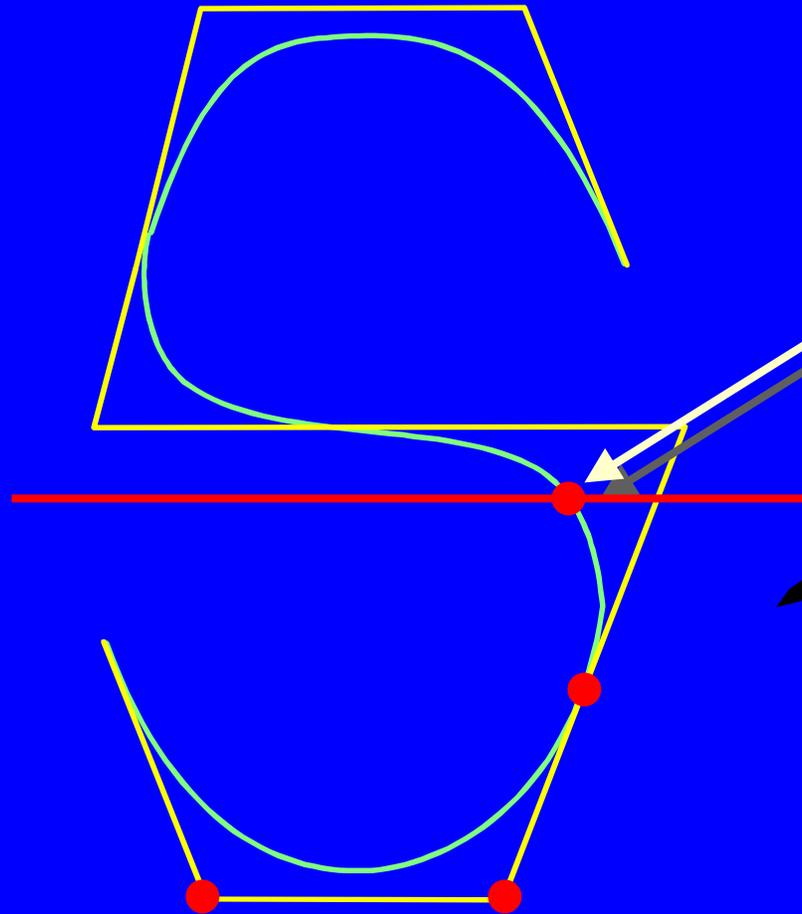
Designing with Bezier

(a la Postscript)



*How do you think
fonts are designed
and represented for
computer use? Can
you figure out how
to do simple italic
operations?*

Question: How to find intersect of raster with "S"

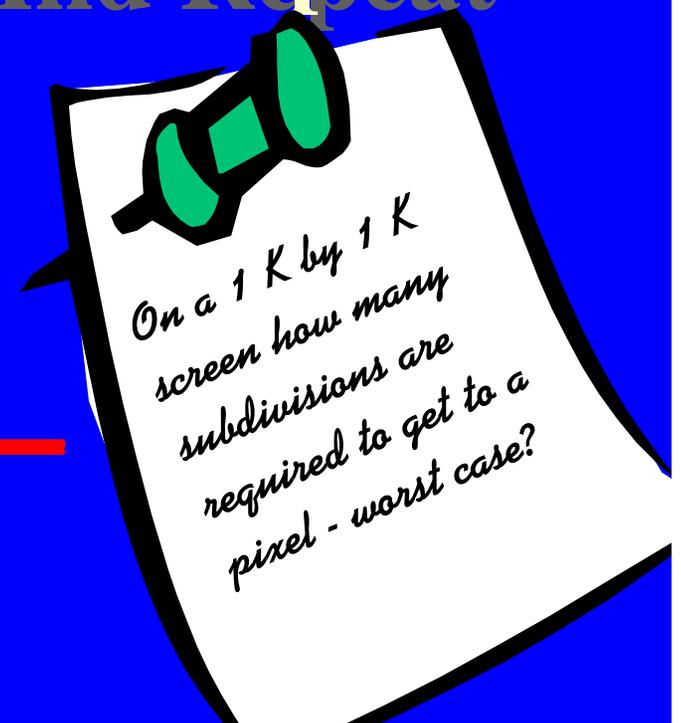
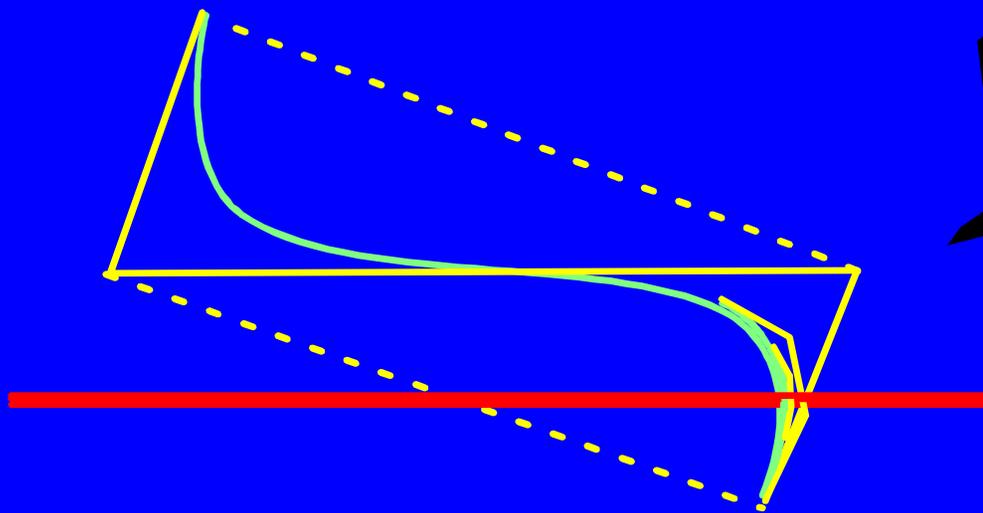


Answer:

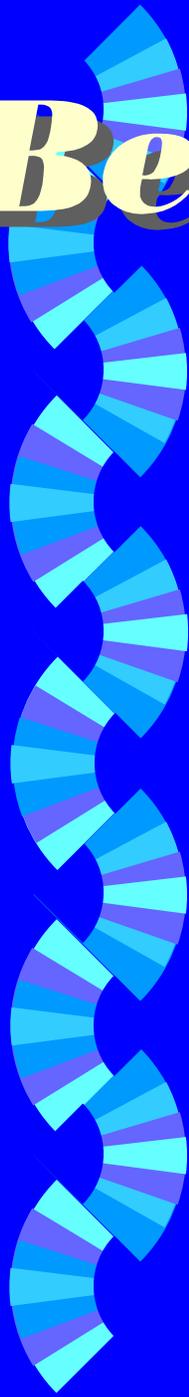
Convex Hull test

Subdivide

And Repeat



Bezier Vocabulary:



Bezier curves: coefficients with meaning - end point, end tangents, etc.

Bernstein polynomials: what Bezier uses

Affine invariance: Rotate, scale translate control points same as curve

Control points: coefficients of Bezier, because they control

Control polygon: connected control points

Variation diminishing: no more wiggles in curve than in control polygon

Convex hull: smallest polygon containing control points. Curve is contained in this

de Casteljau: algorithm for evaluating Bezier curve

Subdivision: Dividing Bezier curve into two matching Bezier curves

Bezier Vocabulary:

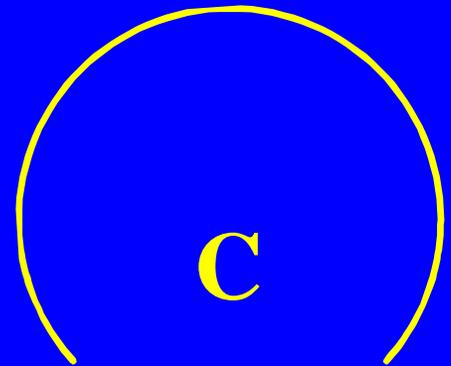
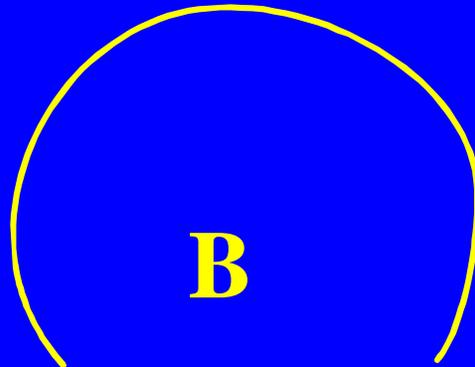
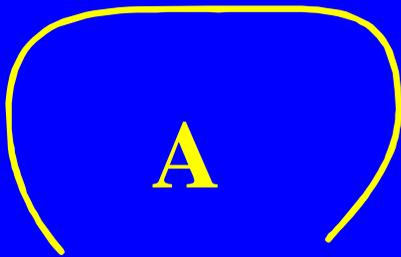
Parametrization: Where the fly is at time t

C^1 : Smoothly changing velocity

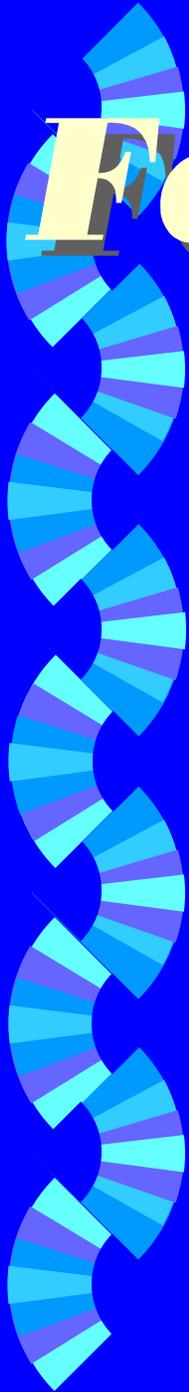
G^1 : Smoothly changing direction

Fairness

Which curve looks best ?

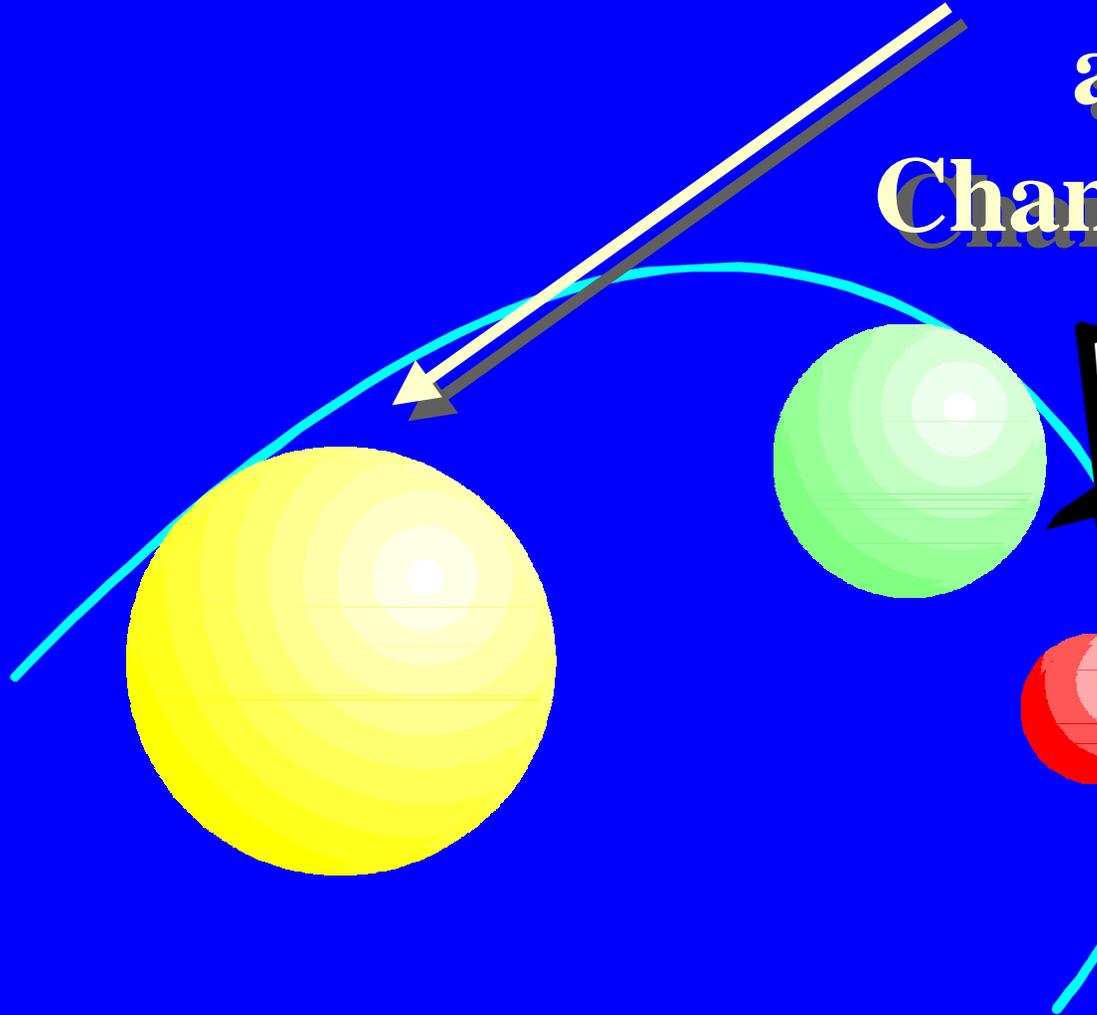
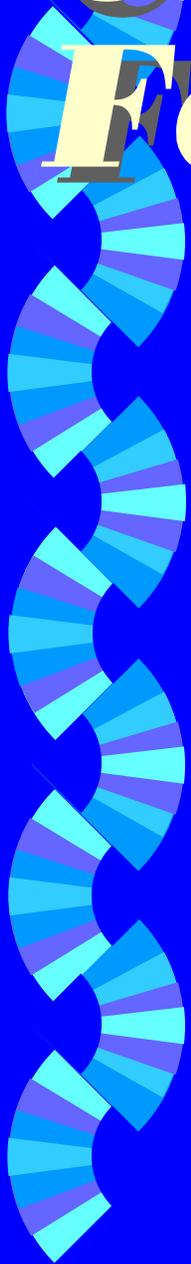


Why ?

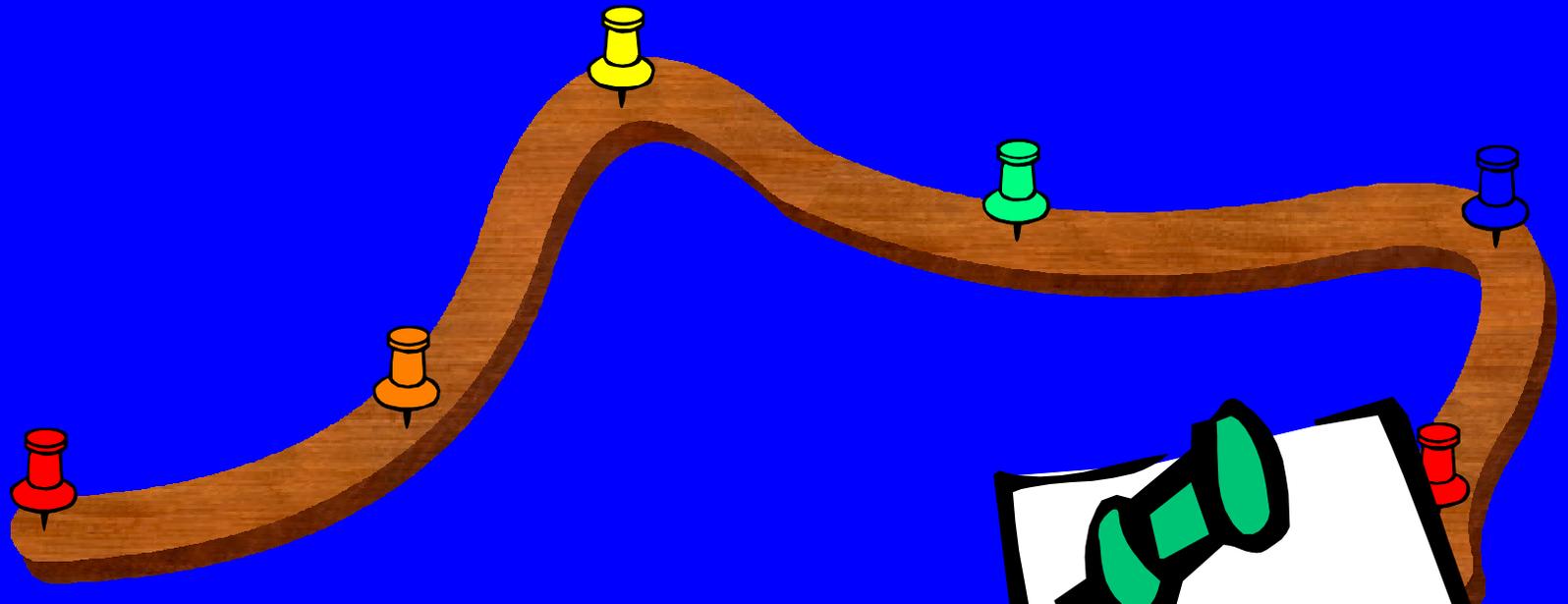


Criteria for Fairness

Curvature of
a Ball
Changes evenly



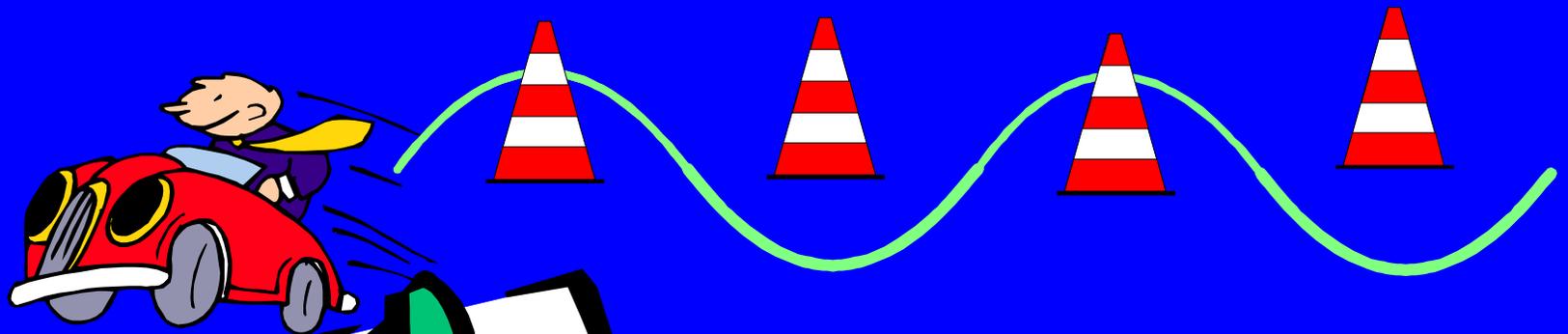
Stress dissipation



Wooden “spline”

Here's another
fram whence the
word spline comes

Minimize accelerations



And another

Many others ...

Recall:

Many others:

- Hermite with endpoints & tangents

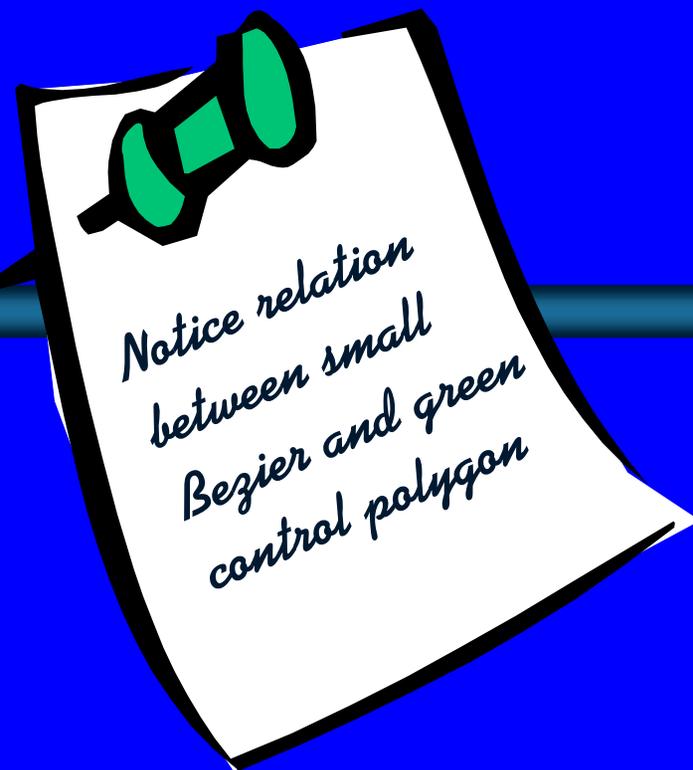
- Lag range thru all points

⌘ Polynomial \Rightarrow rearrange terms so coefficients have meaning

\Rightarrow Bezier curve

⌘ Are there other rearrangements with meaning?

Here is one:



What attributes

Endpoint Interpolation

Tangency at Endpoints

Convex Hull

Variation Diminishing

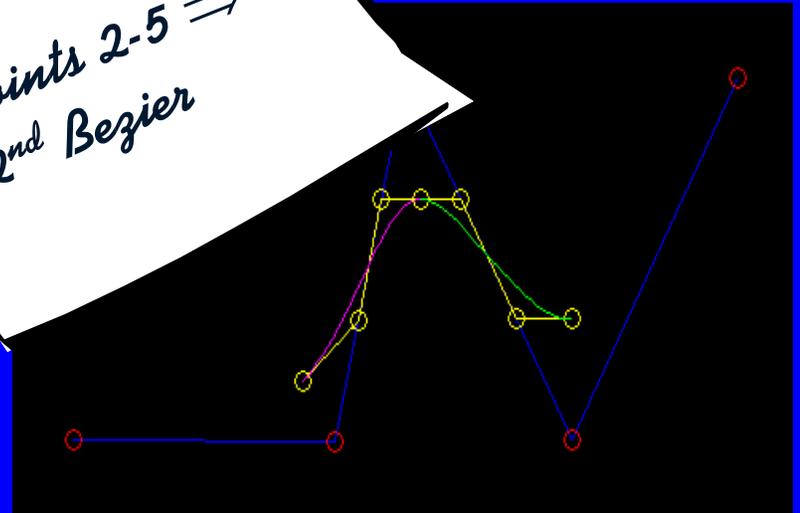
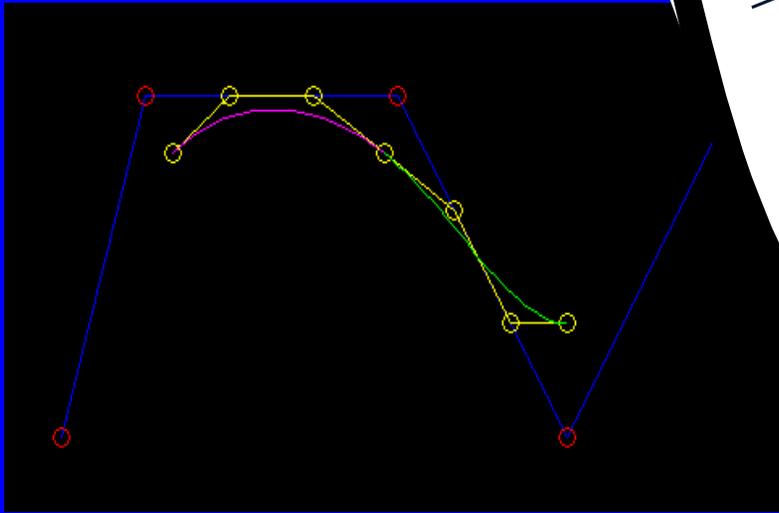
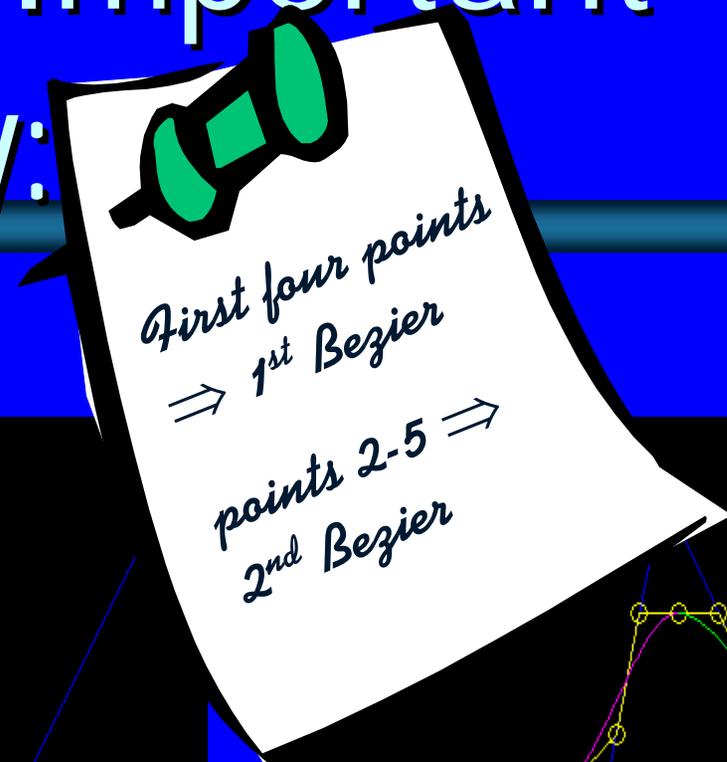
Linear Precision

Affine Invariance

Weak Bezier ?



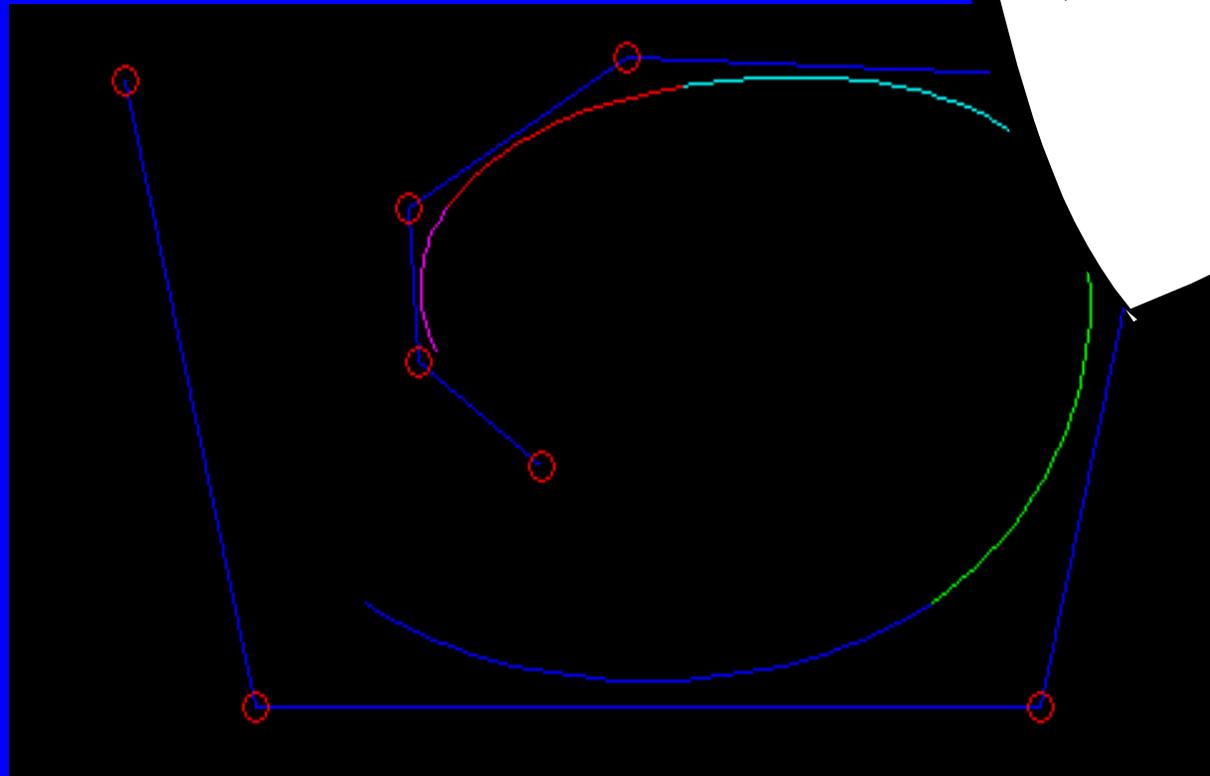
Another Important Property:



Add another control point

Organizes Bezier with Continuity!

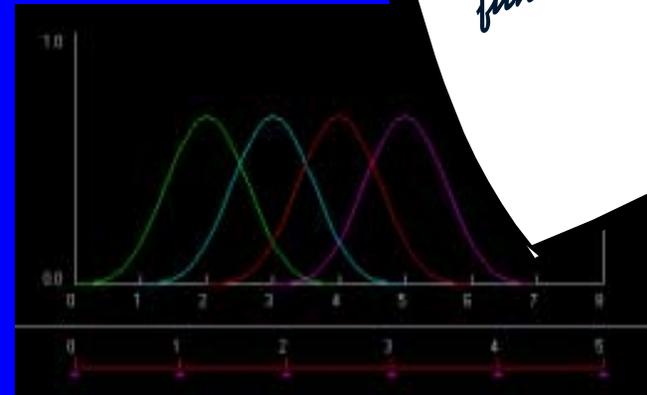
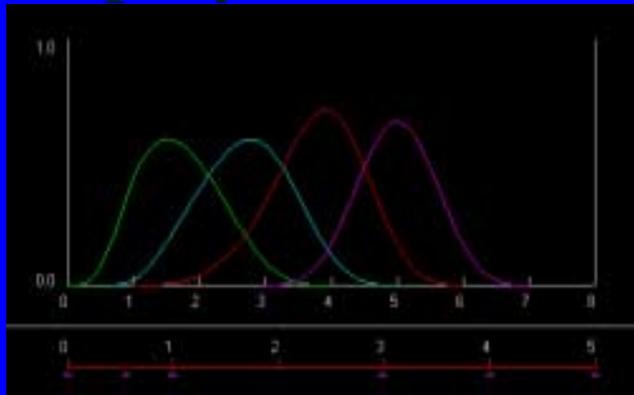
It is called a B-Spline



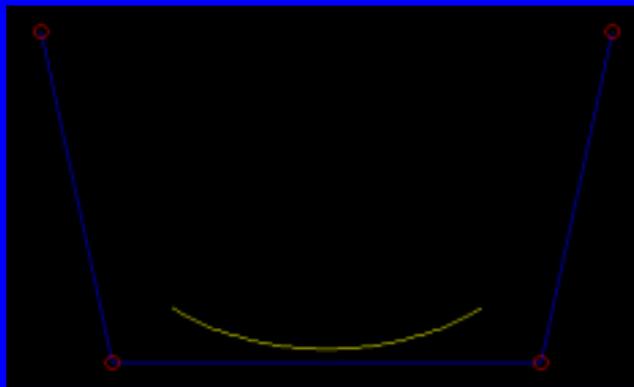
NURBS - very briefly

Stands for Non-Uniform B-Splines because there are many basis functions

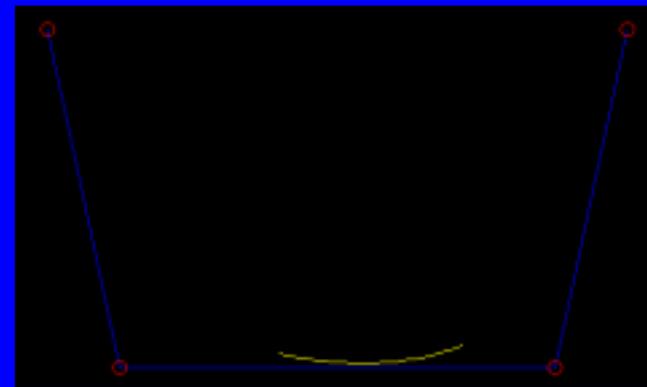
Basis polynomials



Repeat

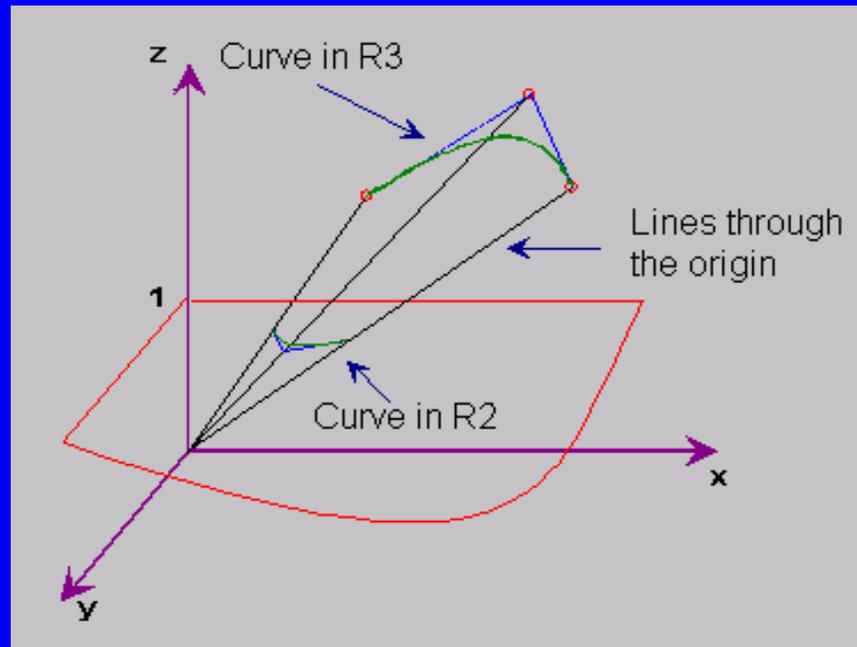
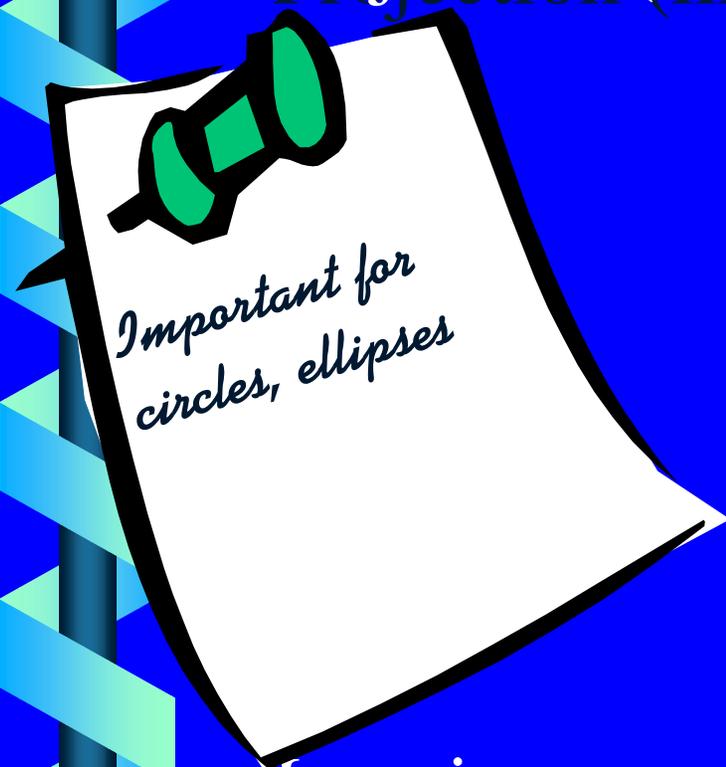


Reshape - Non-uniform



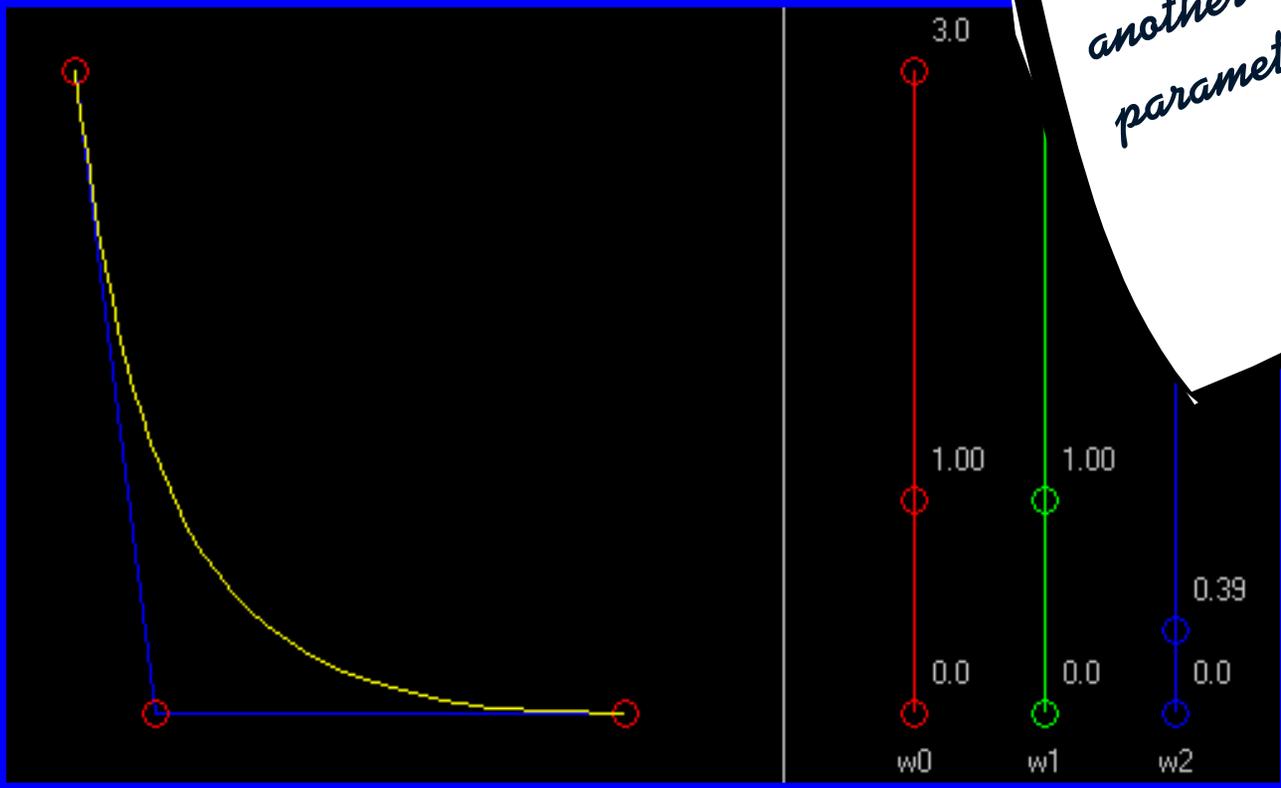
Rational

Projection (like Perspective)



Imagine curve as projection

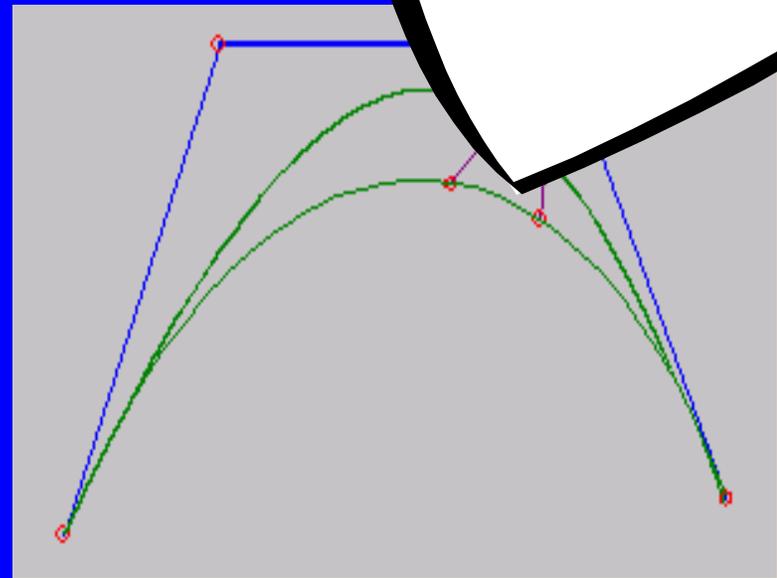
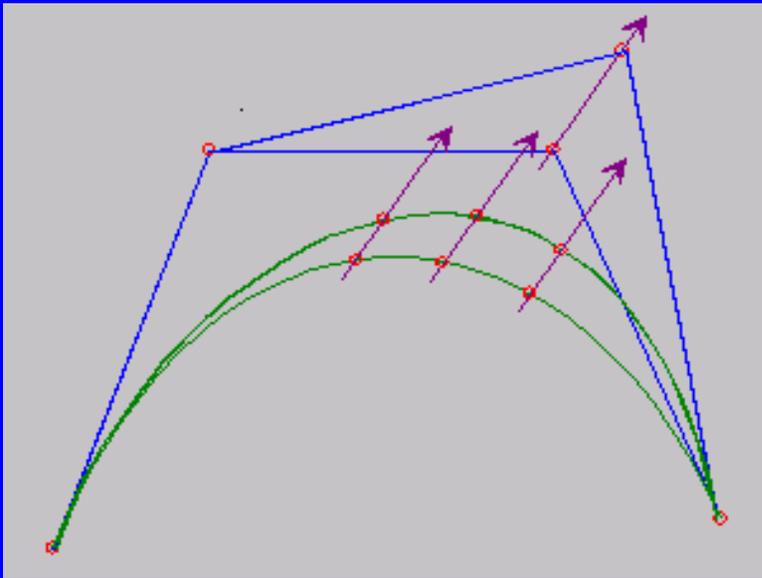
Pull points in higher dimensions



NURBS - Non-Uniform Rational B-Splines

Why Rationals ?

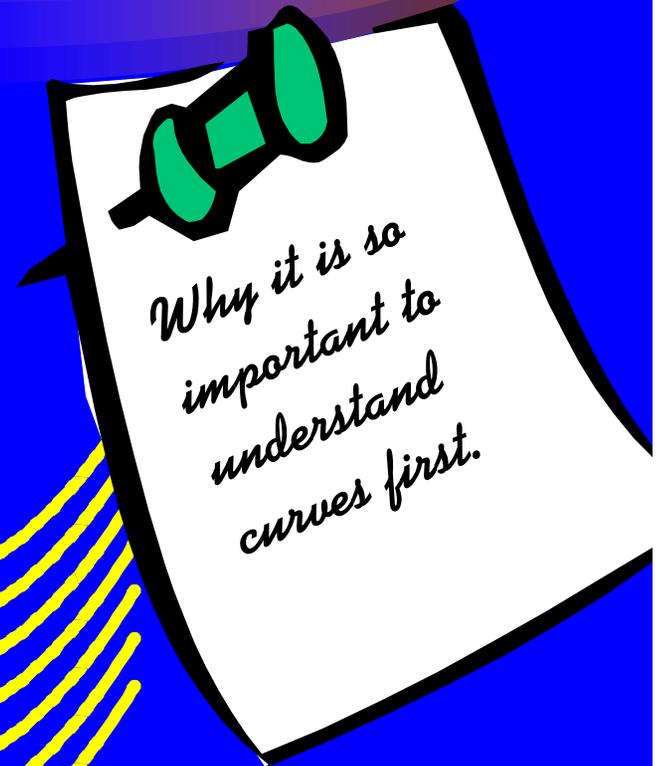
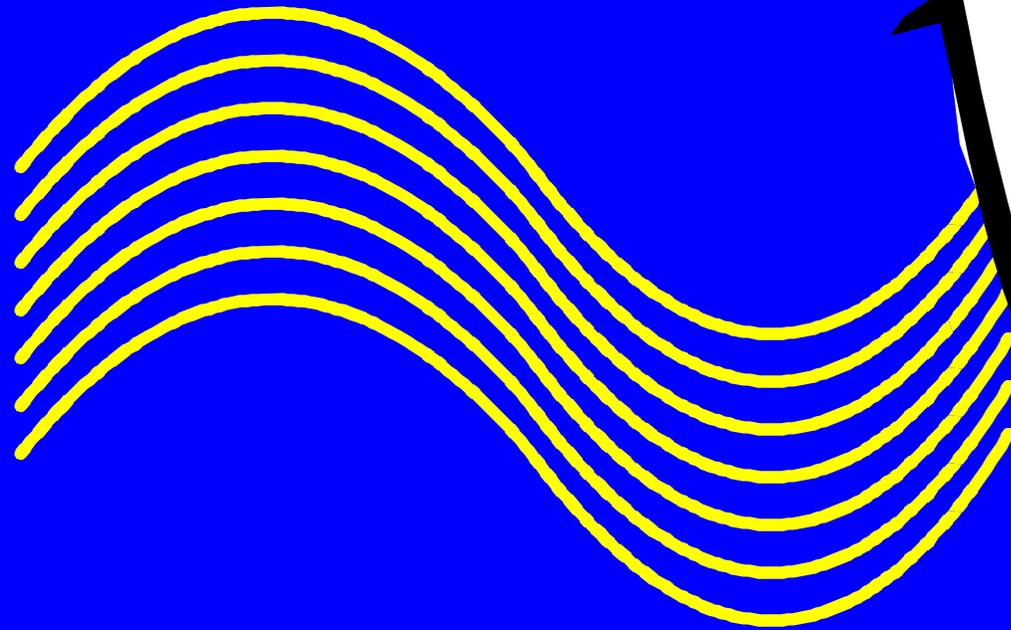
Ball-bearings
alone justify
it.



Polynomials can't do perfect circles
so rationals are needed

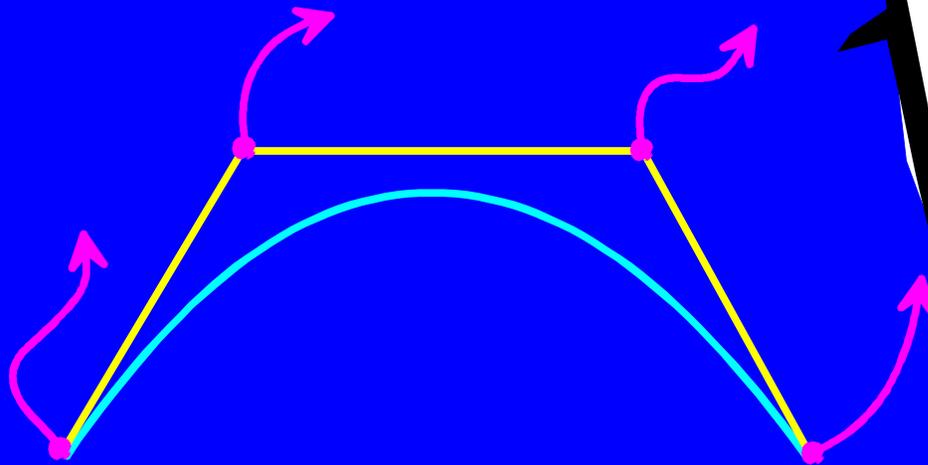
Surfaces

... are just a bundle of curves.



Control points in Space

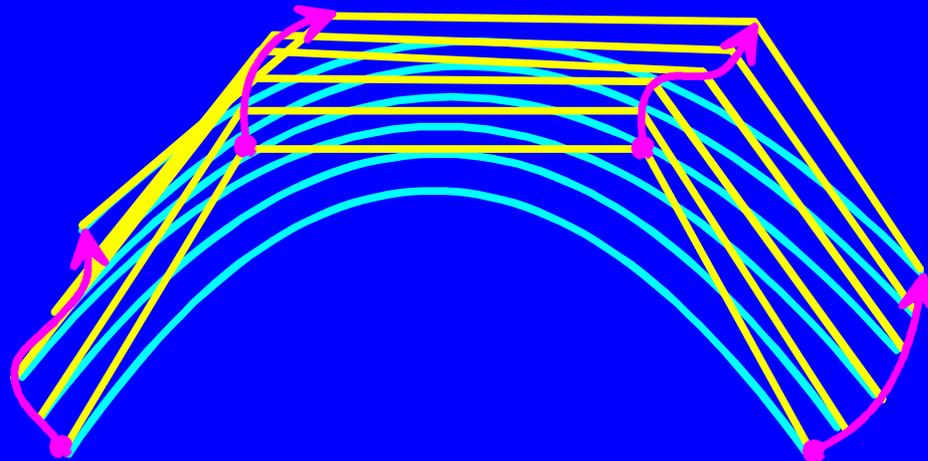
What if control points are moved in space along a curve?



They sweep out a surface

Control points in Space

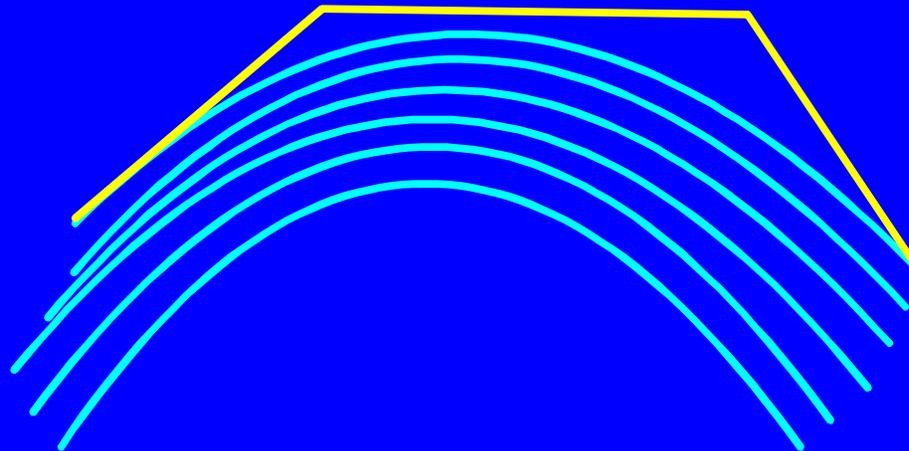
What if control points are moved in space along a curve?



They sweep out a surface

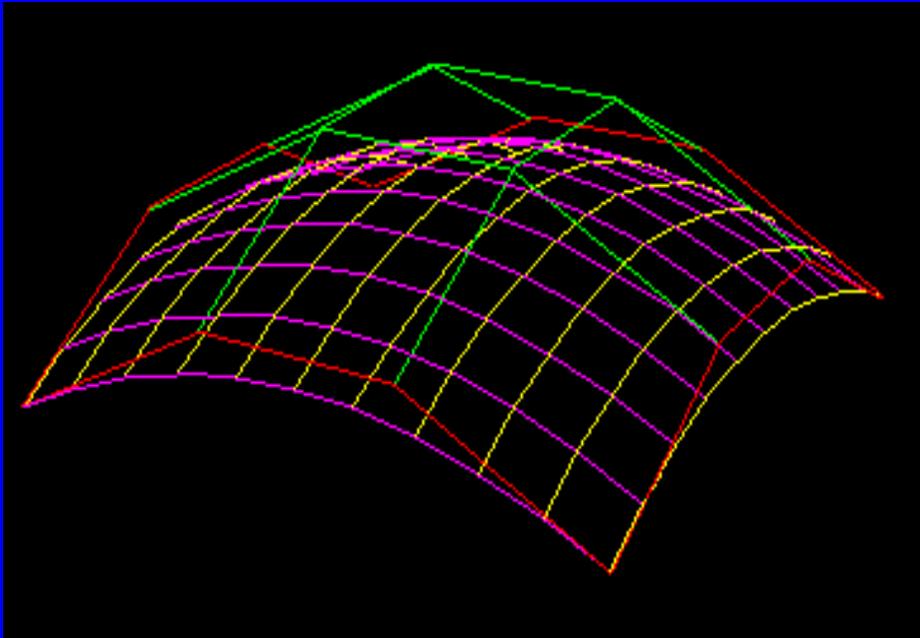
Control points in Space

What if control points are moved in space
along a curve?

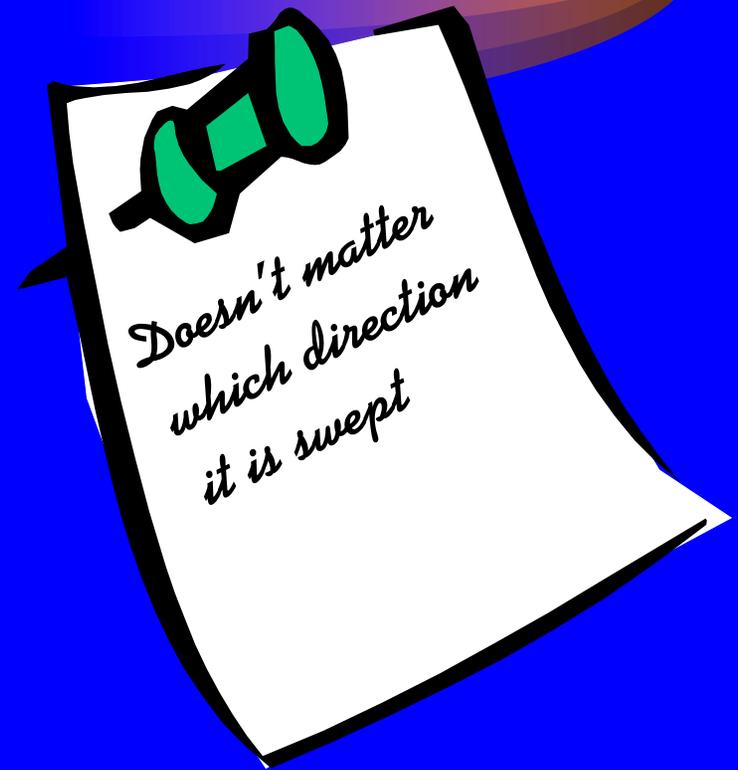


They sweep out a surface

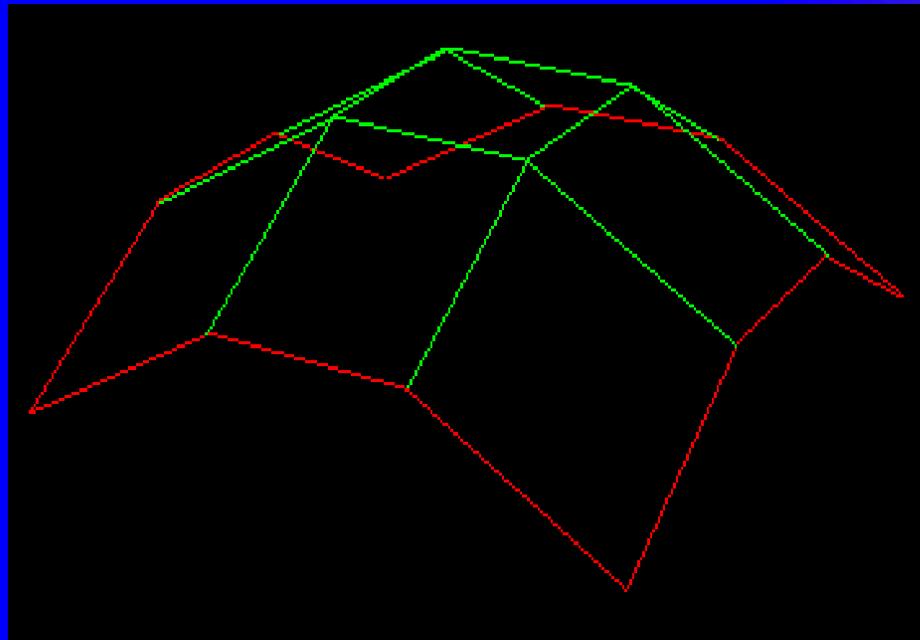
Bezier curves along Bezier “Sweeps”



Bezier
Bezier
“sweeps”
sweeps

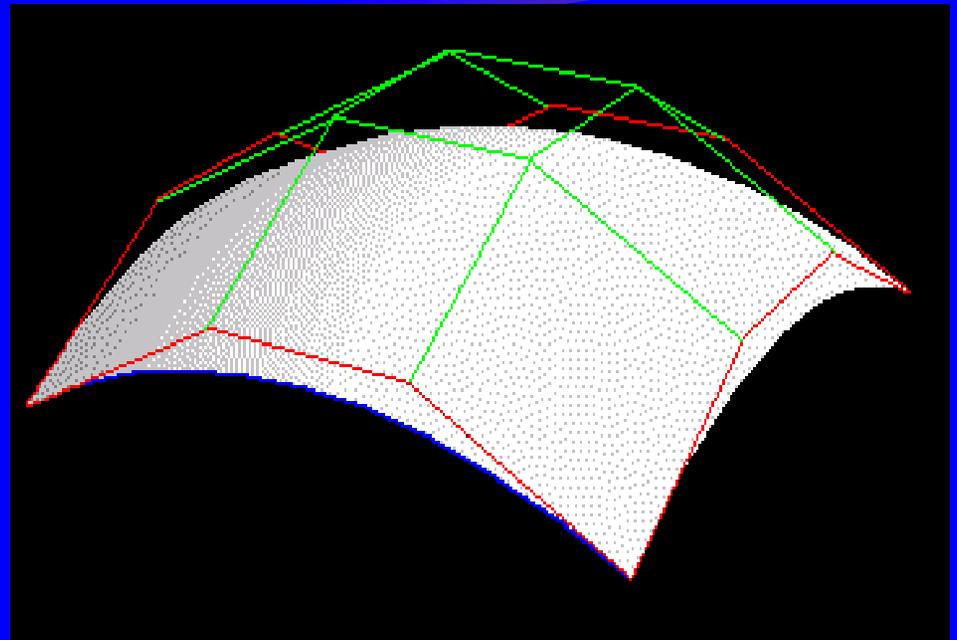


Bezier curves along Bezier “Sweeps”



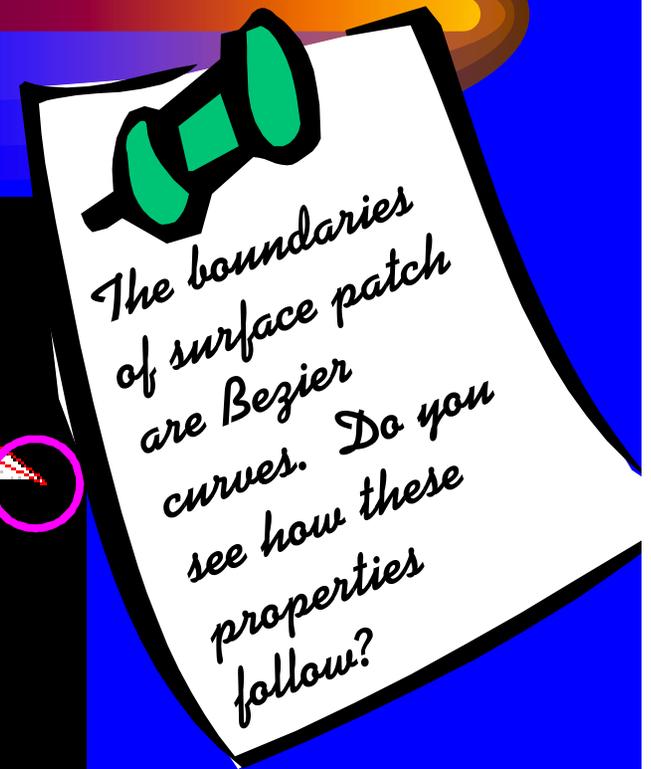
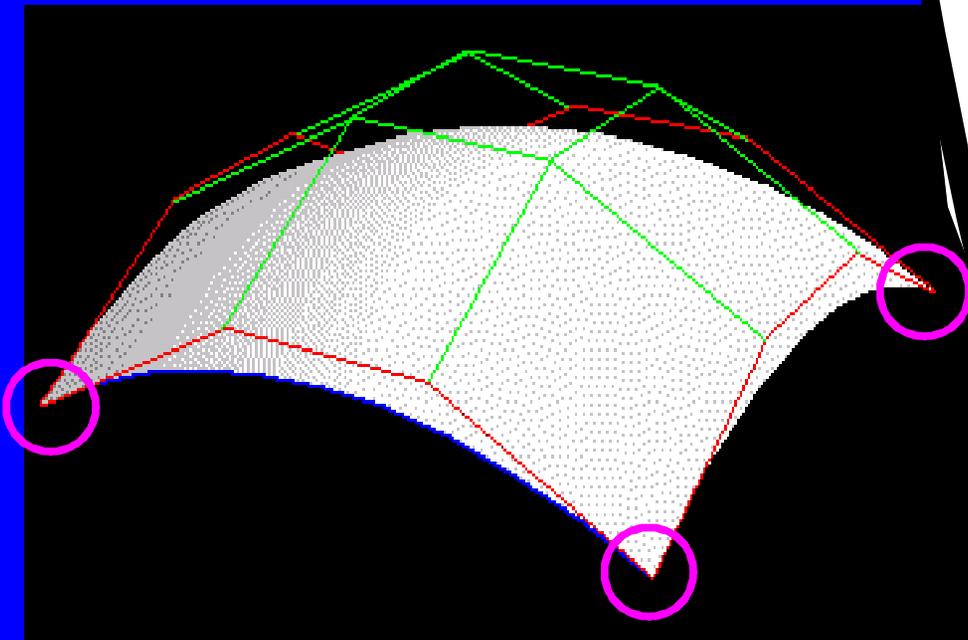
Control
Mesh

Bezier curves along Bezier “Sweeps”



Bezier
Bezier
Surface

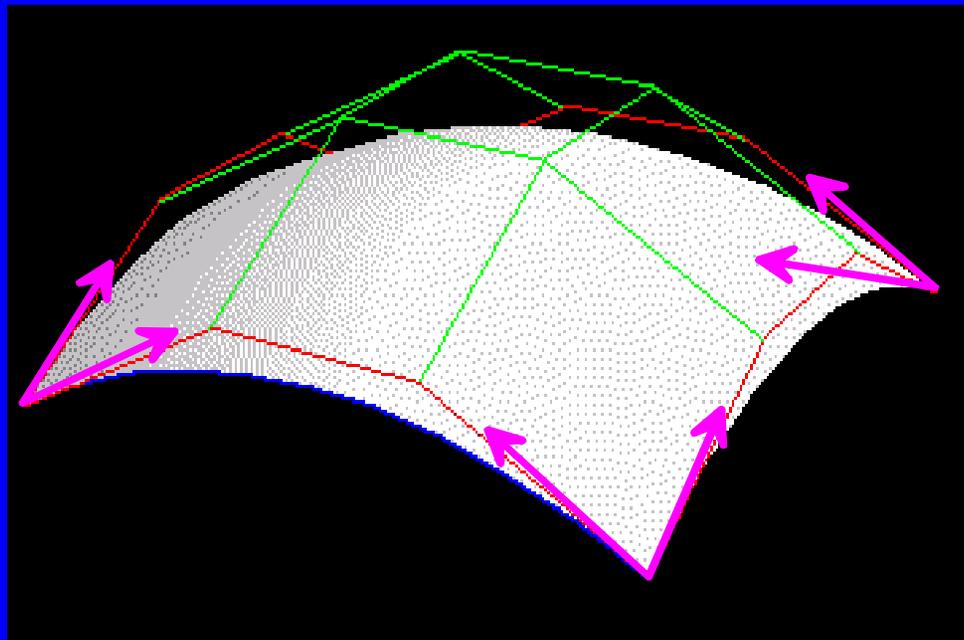
Properties



Endpoints?
Endpoints?

At corners
At corners

Properties



End tangencies?

At corners

Do you see how
these properties
follow from curves?

Properties

Convex Hull ? (Yes)

Affine Invariance ? (Yes)

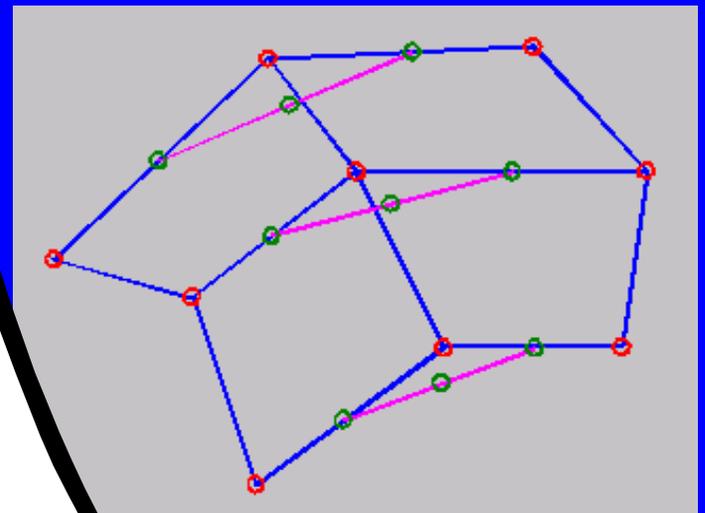
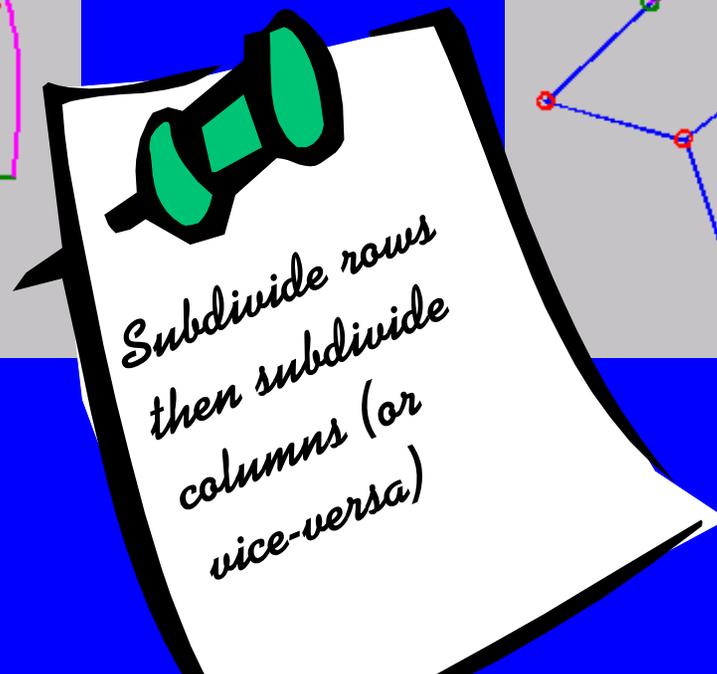
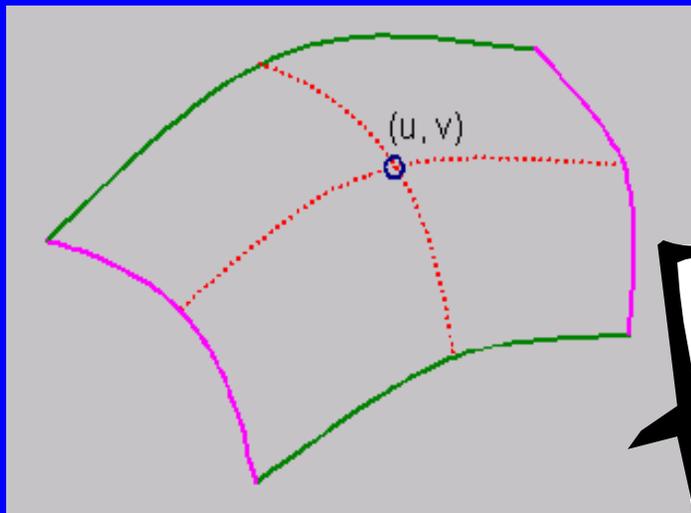
Planar Precision ? (Yes)

Variation Diminishing ? (?)

Subdivision

(Yes)

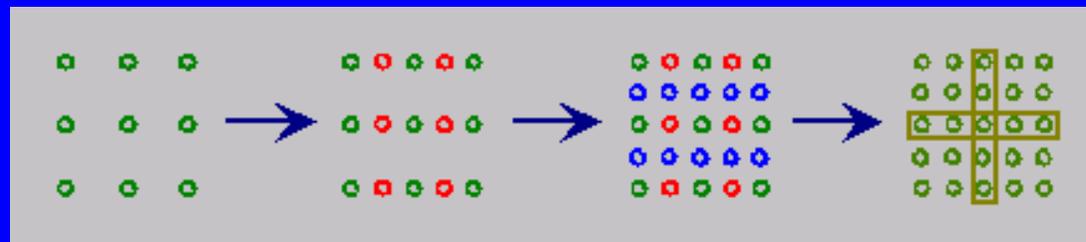
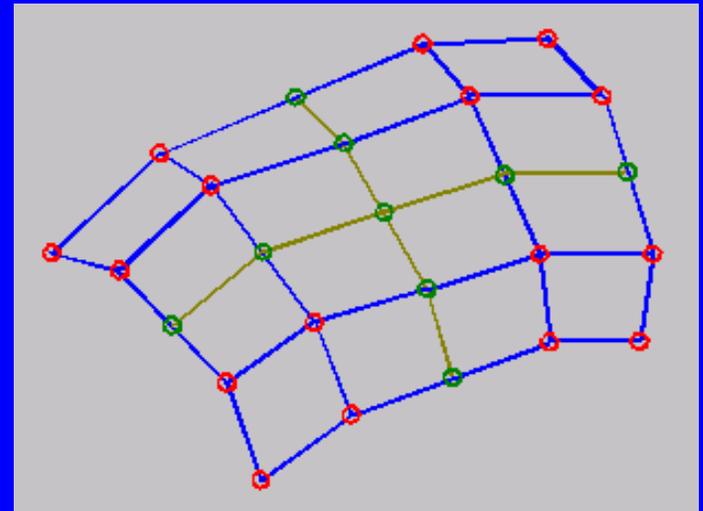
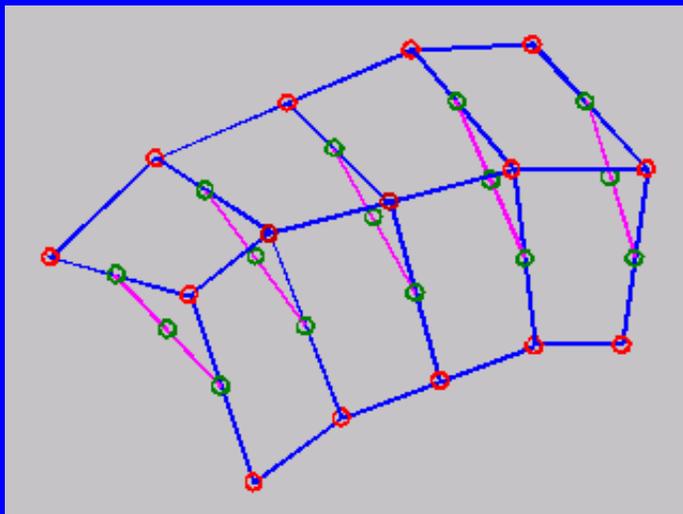
Pyramids instead of triangles



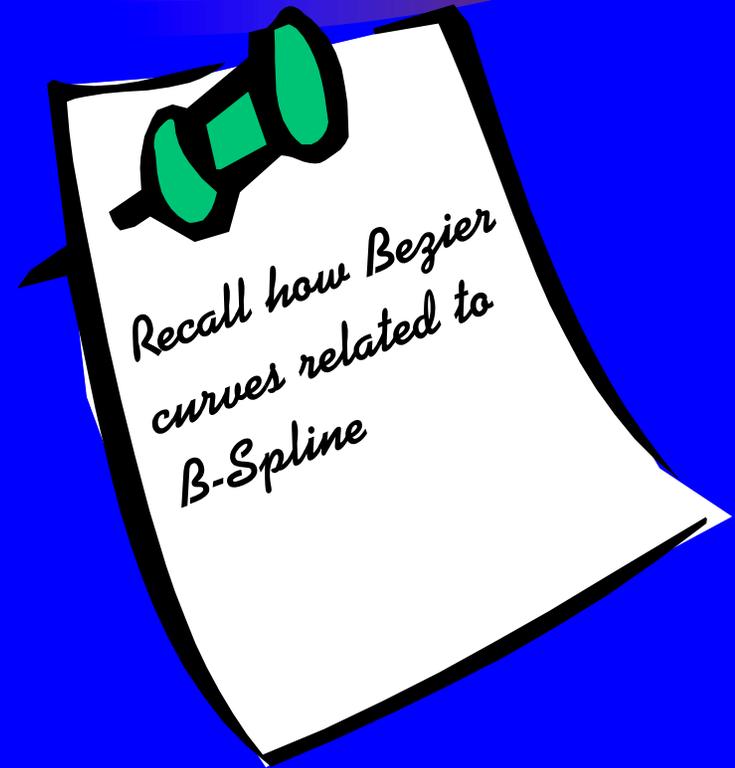
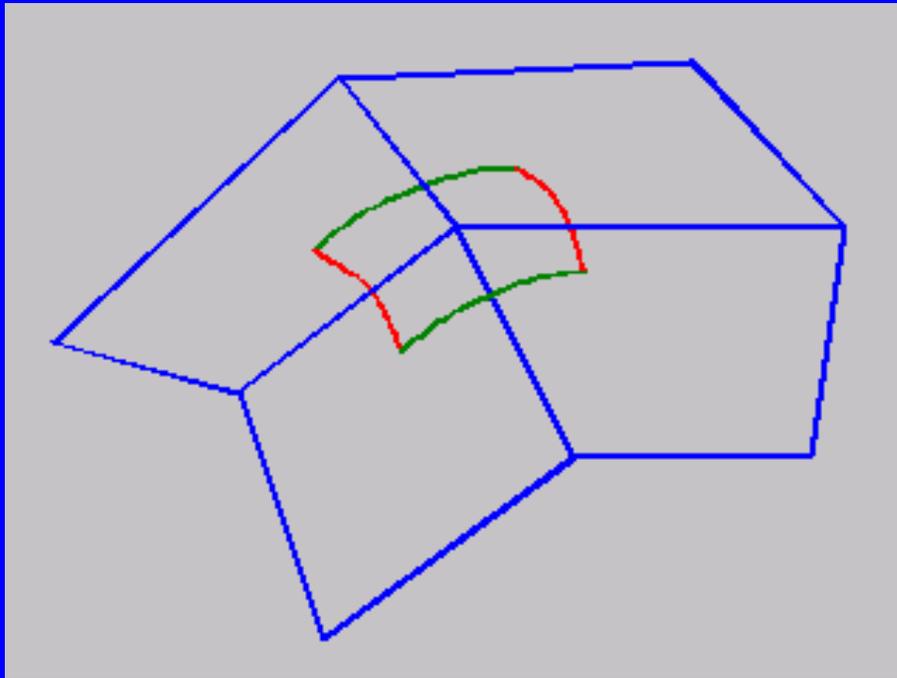
Subdivision

(Yes)

Pyramids instead of triangles



B-Spline Patch



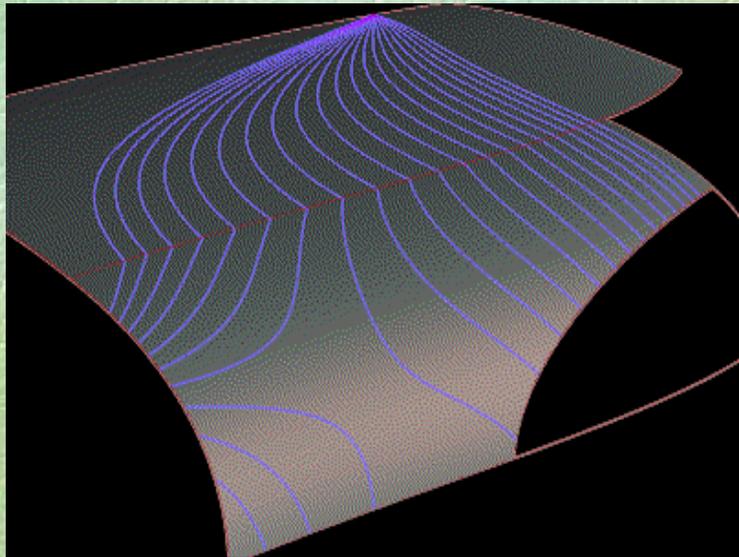
Advantage ?

B-Spline organizes Bezier

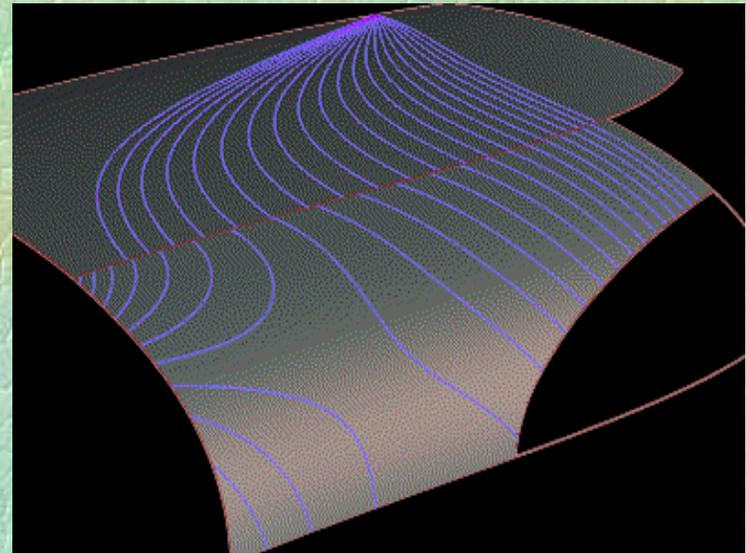


Common wisdom \Rightarrow Design B-spline; operate
Bezier

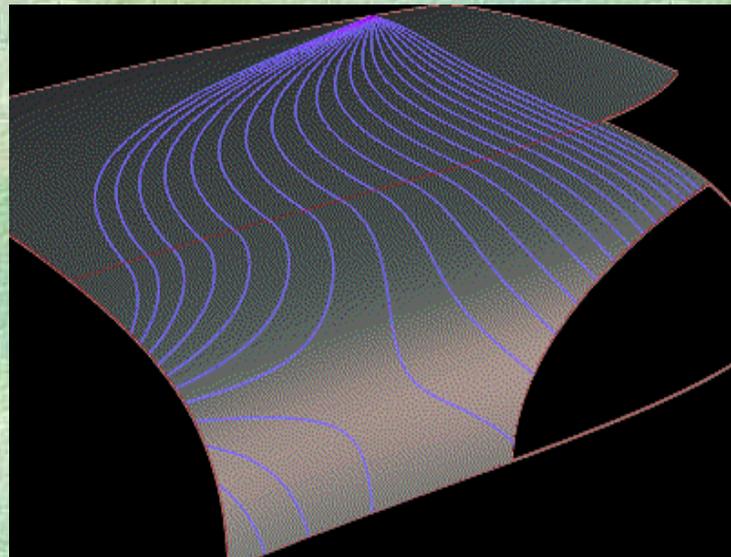
Design Issues



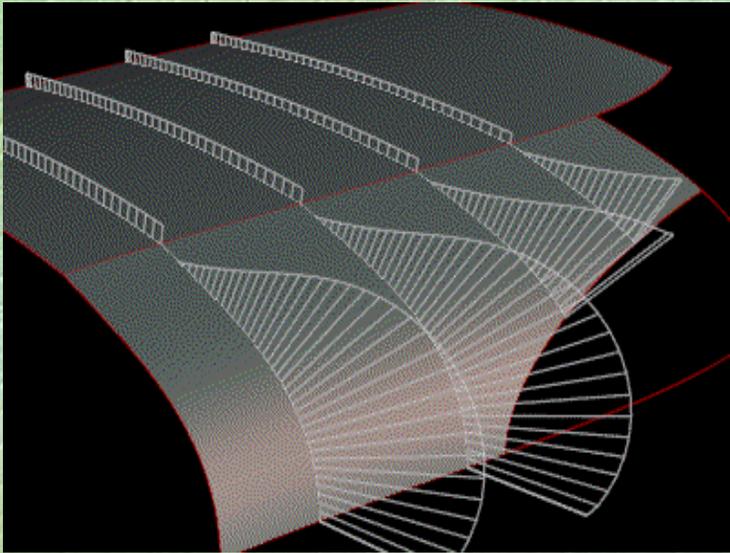
G^0



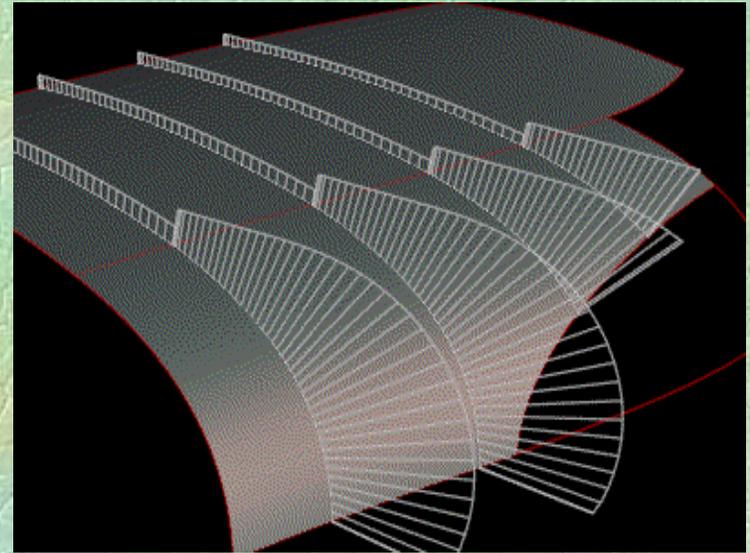
G^1



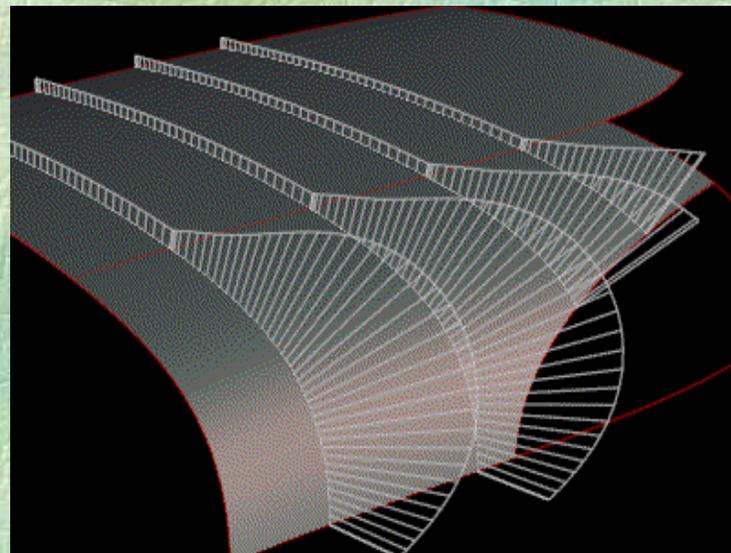
G^2



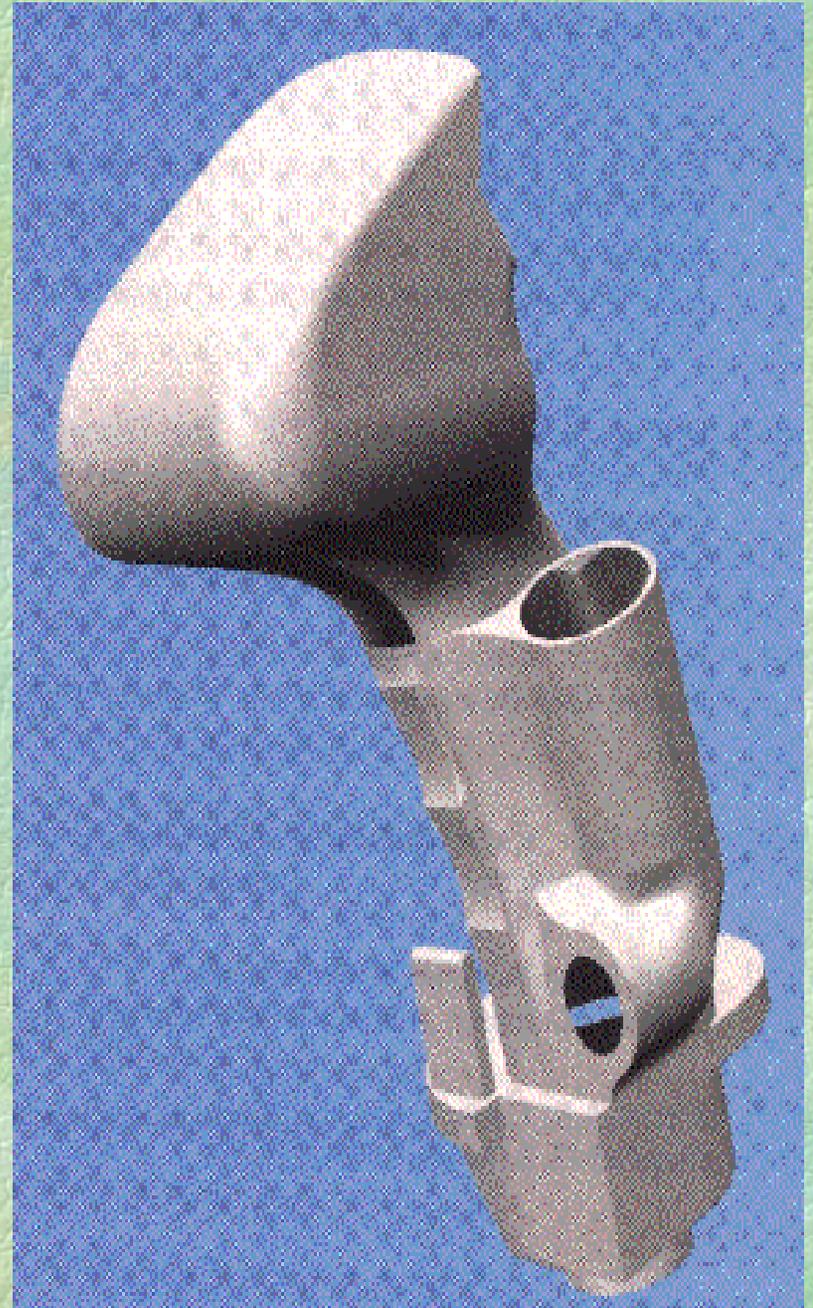
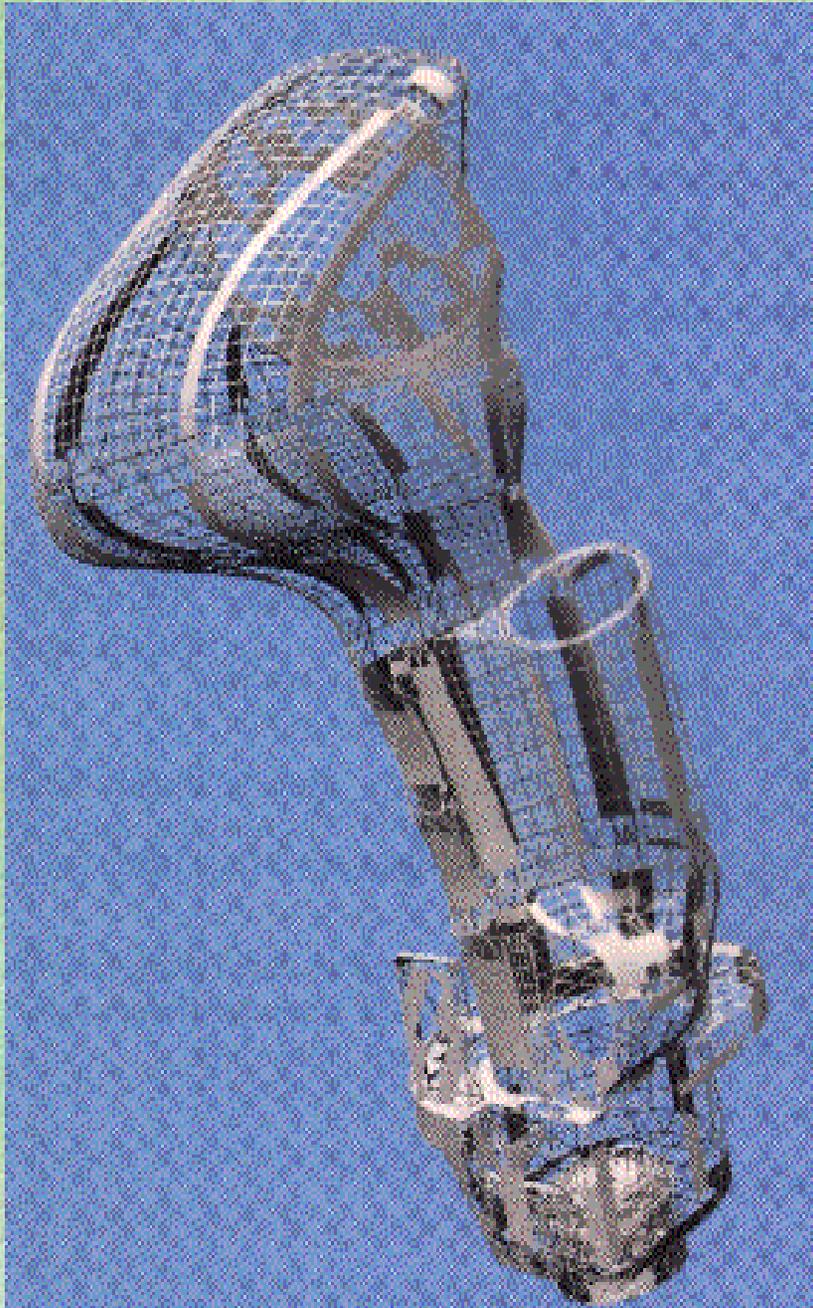
Hedgehog G^0



Hedgehog G^1



Hedgehog G^2

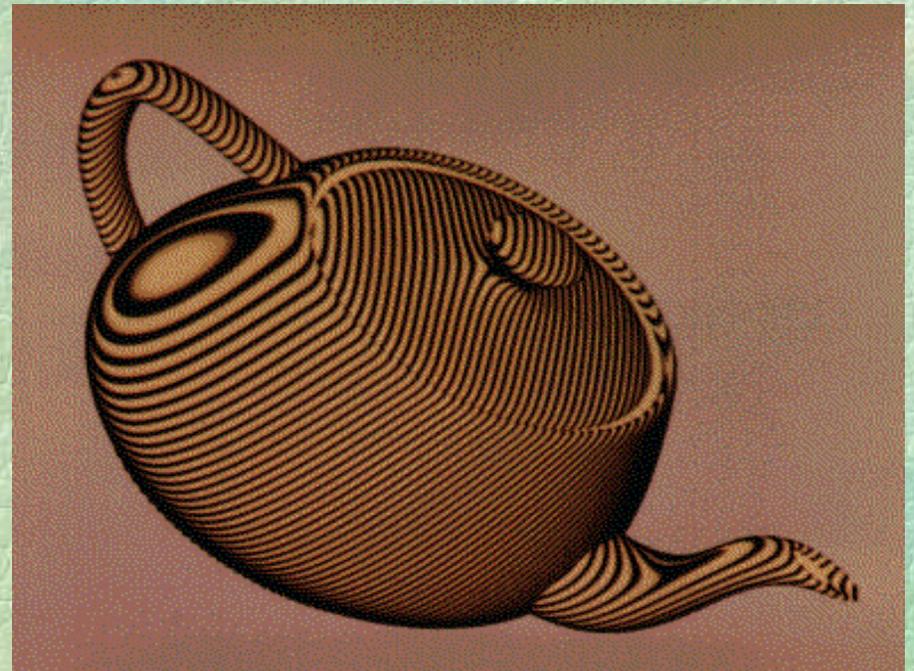


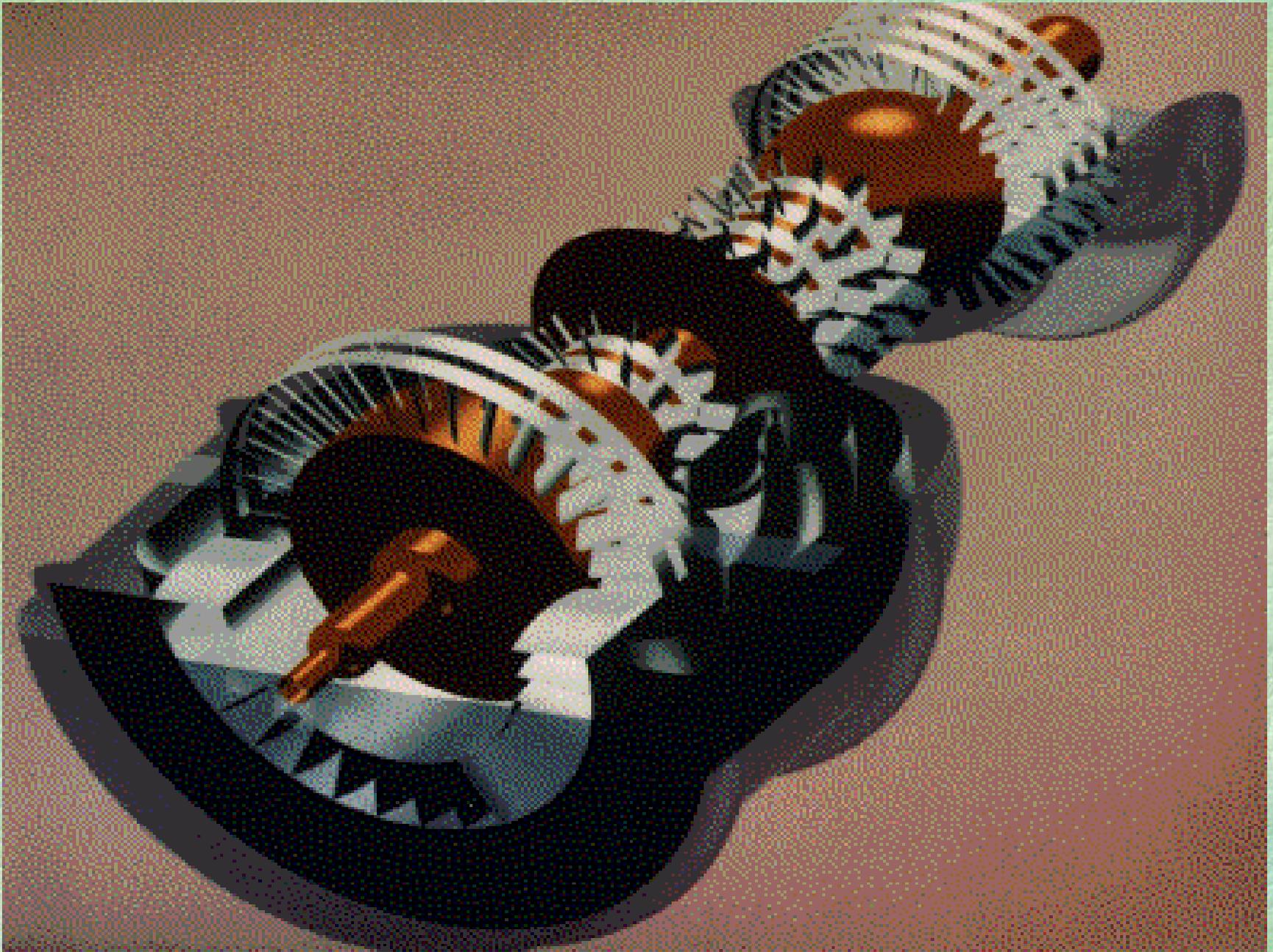
Alyn Rockwood



Peter Chambers

Alyn Rockwood

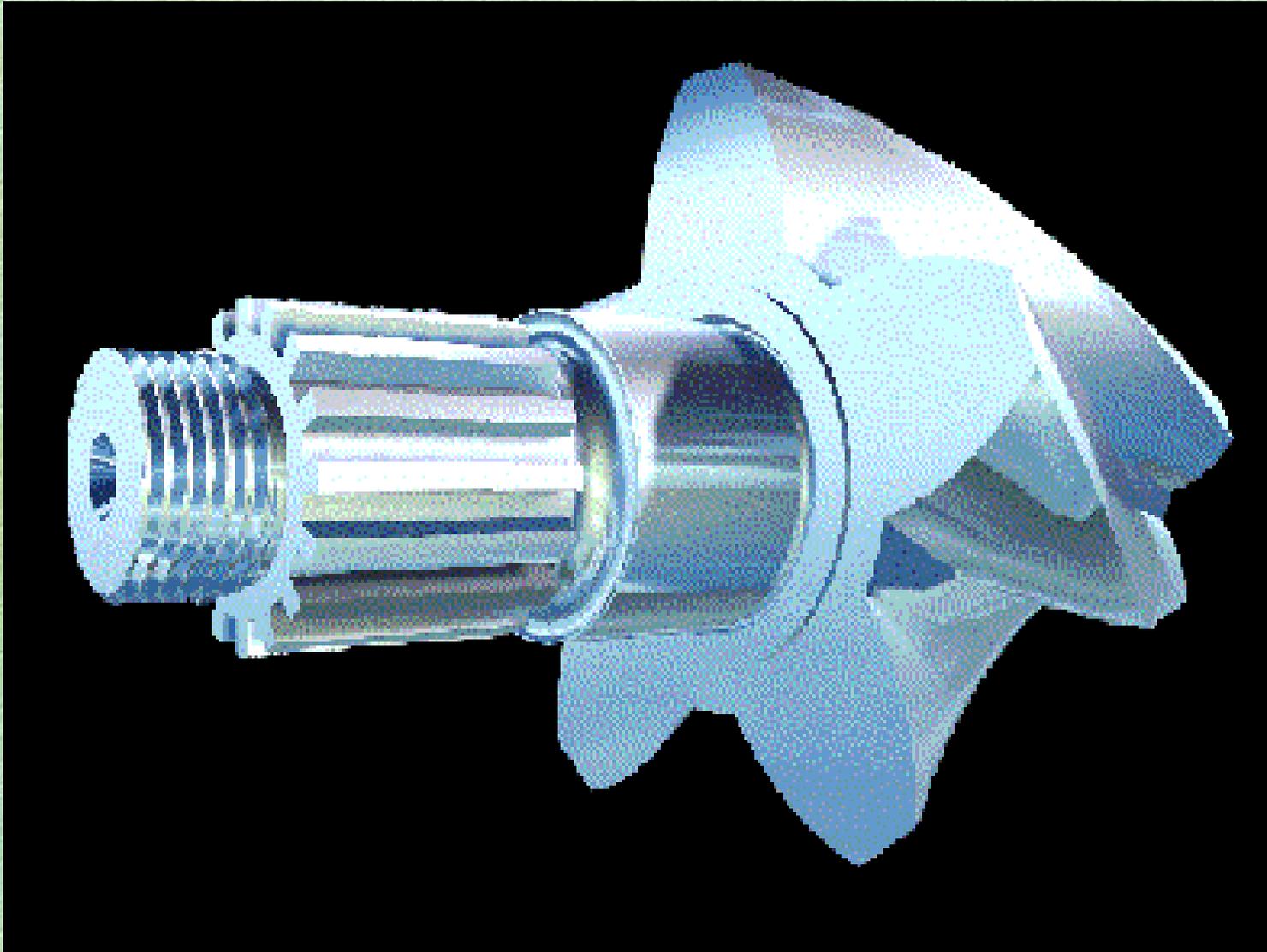




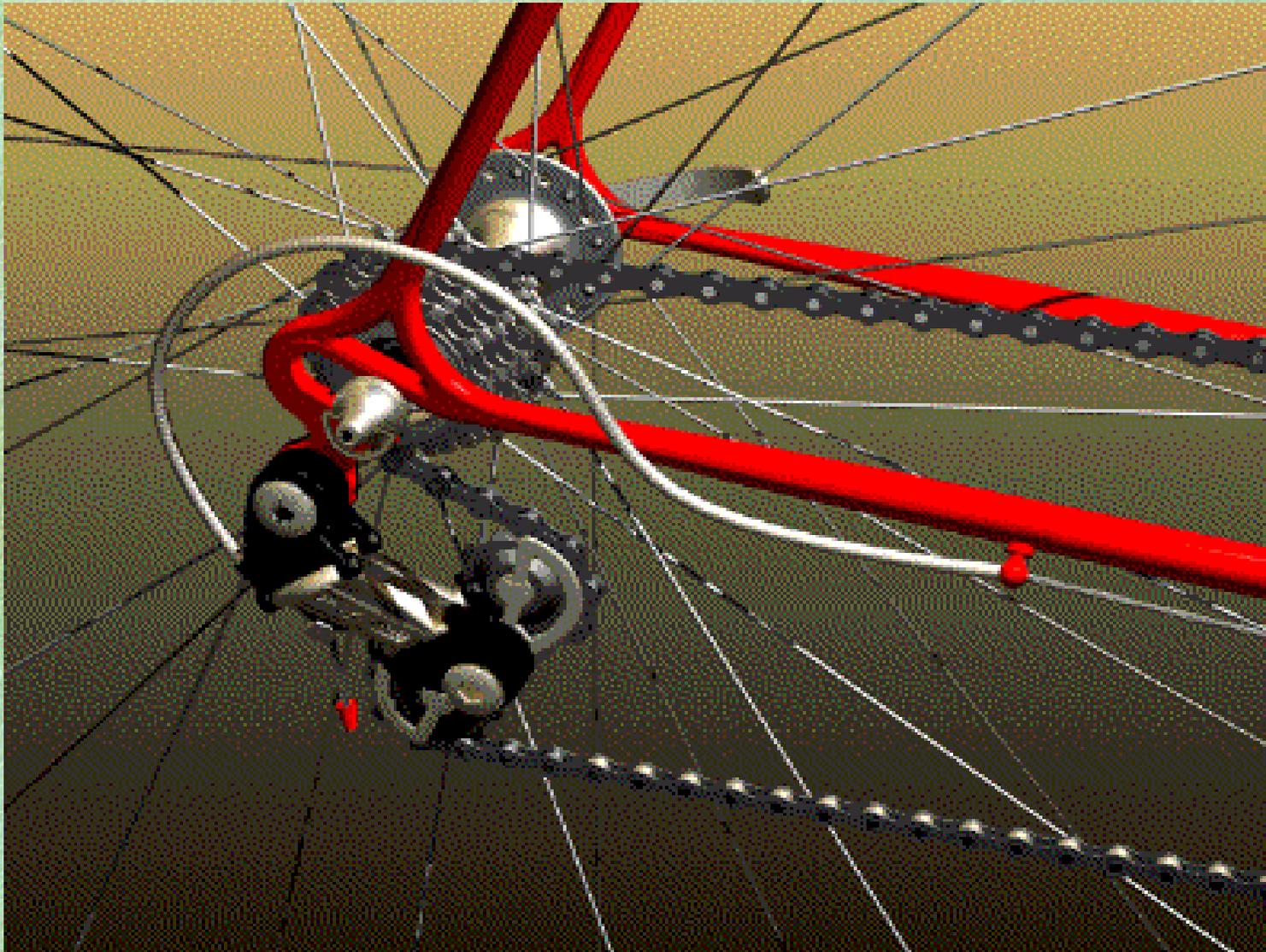
Alyn Rockwood



Courtesy of Parametric Technology Corp. © 1995
Data generated by Lan Zaback, Pro/CDRS,
rendered in Pro/PHOTORENDER



© 1994 Autodesk, Inc. Reprinted with permission. From the Second Annual Autodesk International Image and Animation Contest, 1994. Submitted by John W. Barton, Jr.



© 1994 Autodesk, Inc. Reprinted with permission. From the Second Annual Autodesk International Image and Animation Contest, 1994. Submitted by Andrew Rodriguez

Dynamic NURBS (D-NURBS)

Demetri Terzopoulos

New York University

Courant Institute of Mathematical Sciences

Physics-Based CAGD

***Geometric models + physical laws =
Dynamic models***

- Generalization of geometric design process
- Standard geometric toolkits remain usable
- Additional physics-based toolkits
 - *sculpting forces*
 - *linear & nonlinear constraints*

Advantages of Physics-Based Design

- Energies express global “fairness” criteria
- Forces support direct, interactive sculpting
- Constraints permit functional design
- Shape optimization via evolution to equilibrium
- Dynamics allow time-varying conceptual design
- Automatic selection of DOFs

Dynamic NURBS (D-NURBS)

Physics-based generalization of NURBS

(Developed with Hong Qin)

- Lagrangian mechanics formulation
- NURBS DOFs are D-NURBS generalized coordinates
- Mass and damping distribution
- Deformation energy
- Applied forces
- Geometric constraints

D-NURBS Curve Geometry & Kinematics

- Curve:

$$\mathbf{c}(u, t) = \frac{\sum_{i=0}^n \mathbf{p}_i(t) w_i(t) \mathbf{B}_{i,k}(u)}{\sum_{i=0}^n w_i(t) \mathbf{B}_{i,k}(u)}$$

control point → weight → B-spline basis fn

parametric coordinate ↑ time
- DOF vector:

$$\mathbf{p}(t) = [\mathbf{p}_0^T, w_0, \dots, \mathbf{p}_n^T, w_n]^T$$
- Kinematics:

$$\dot{\mathbf{c}}(u, \mathbf{p}) = \mathbf{J} \dot{\mathbf{p}} \quad \mathbf{c}(u, \mathbf{p}) = \mathbf{J} \mathbf{p}$$

Jacobian matrix

Tensor Product D-NURBS Surface Geometry & Kinematics

- Surface:

$$\mathbf{s}(u, v) = \frac{\sum_{i=0}^n \sum_{j=0}^m \mathbf{p}_{i,j}(t) w_{i,j}(t) \mathbf{B}_{i,k}(u) \mathbf{B}_{j,l}(v)}{\sum_{i=0}^n \sum_{j=0}^m w_{i,j}(t) \mathbf{B}_{i,k}(u) \mathbf{B}_{j,l}(v)}$$
- DOF vector:

$$\mathbf{p}(t) = [\mathbf{p}_0^T, w_0, \dots, \mathbf{p}_N^T, w_N]^T$$

$N = n \times m$
- Kinematics:

$$\dot{\mathbf{s}}(u, v, \mathbf{p}) = \mathbf{J} \dot{\mathbf{p}} \quad \mathbf{s}(u, v, \mathbf{p}) = \mathbf{J} \mathbf{p}$$

Lagrangian Mechanics

Equations of motion

kinetic energy damping energy potential energy

$$\frac{\partial}{\partial t} \left(\frac{\partial T}{\partial \dot{p}_i} \right) + \frac{\partial F}{\partial \dot{p}_i} + \frac{\partial U}{\partial p_i} = f_i$$

generalized coordinate

external generalized force on p_i

Kinetic and Damping Energies

- Kinetic energy

$$T = \frac{1}{2} \iiint \mu \dot{\mathbf{s}}^T \dot{\mathbf{s}} \, du \, dv = \frac{1}{2} \dot{\mathbf{p}}^T \mathbf{M} \dot{\mathbf{p}}$$

- Damping energy

$$F = \frac{1}{2} \iiint \gamma \dot{\mathbf{s}}^T \dot{\mathbf{s}} \, du \, dv = \frac{1}{2} \dot{\mathbf{p}}^T \mathbf{D} \dot{\mathbf{p}}$$

Potential Energy

Energy of deformation

- Example: Thin-plate under tension model

$$U = \frac{1}{2} \iint (\alpha_{1,1} \mathbf{s}_u^2 + \alpha_{2,2} \mathbf{s}_v^2 + \beta_{1,1} \mathbf{s}_{uu}^2 + \beta_{1,2} \mathbf{s}_{uv}^2 + \beta_{2,2} \mathbf{s}_{vv}^2) du dv$$

rigidity control functions

tension and rigidity are functions of (u,v)

D-NURBS Matrices

- Mass matrix: $\mathbf{M}(\mathbf{p}) = \frac{1}{2} \iint \mu \mathbf{J}^T \mathbf{J} du dv$
- Damping matrix: $\mathbf{D}(\mathbf{p}) = \frac{1}{2} \iint \gamma \mathbf{J}^T \mathbf{J} du dv$
- Stiffness matrix:

$$\mathbf{K}(\mathbf{p}) = \frac{1}{2} \iint (\alpha_{1,1} \mathbf{J}_u^T \mathbf{J}_u + \alpha_{2,2} \mathbf{J}_v^T \mathbf{J}_v + \beta_{1,1} \mathbf{J}_{uu}^T \mathbf{J}_{uu} + \beta_{1,2} \mathbf{J}_{uv}^T \mathbf{J}_{uv} + \beta_{2,2} \mathbf{J}_{vv}^T \mathbf{J}_{vv}) du dv$$

D-NURBS Dynamics

Equations of motion

- Second-order ODEs

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}_p - \mathbf{I}\dot{\mathbf{p}}$$

generalized forces

$$\mathbf{f}_p = \iint \mathbf{J}^T \mathbf{f} \, du \, dv$$

applied force distribution $\mathbf{f}(u, v, t)$

inertia matrix

$$\mathbf{I} = \iint \mu \mathbf{J}^T \mathbf{J} \, du \, dv$$

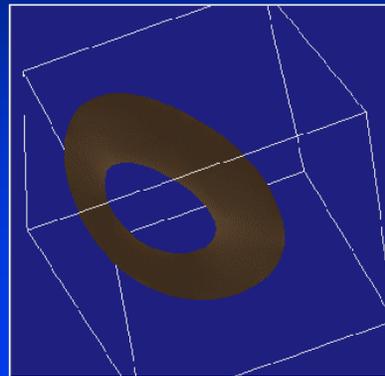
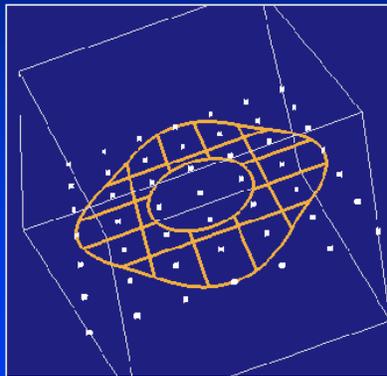
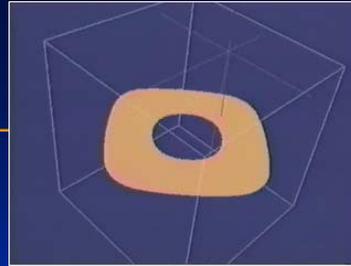
Numerical Implementation

Finite element method approach

- D-NURBS patch is a finite element
- Gaussian quadrature to assemble element matrices \mathbf{M} , \mathbf{D} , \mathbf{K}
 - use *sparse matrix techniques*
- Numerical time integration of motion equation
 - *conjugate gradient iteration during timestep*
- Efficient parallel implementation

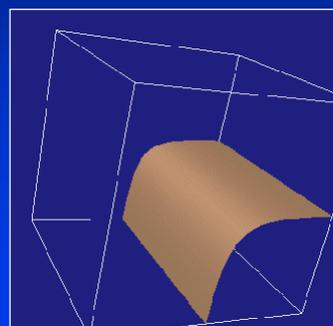
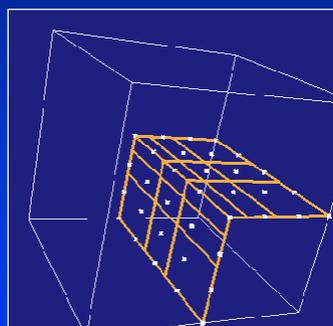
Trimming D-NURBS

NURBS trimming curves



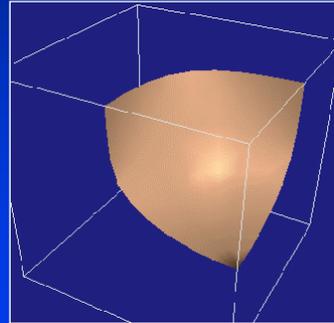
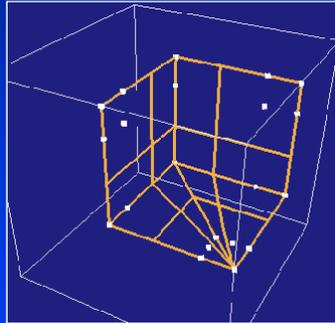
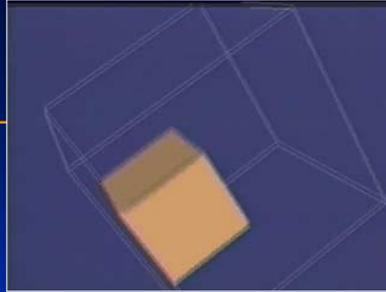
Solid Rounding

Deformation energy rounds an edge



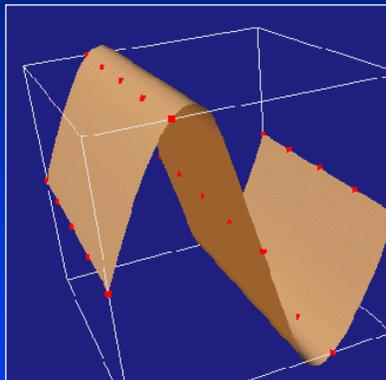
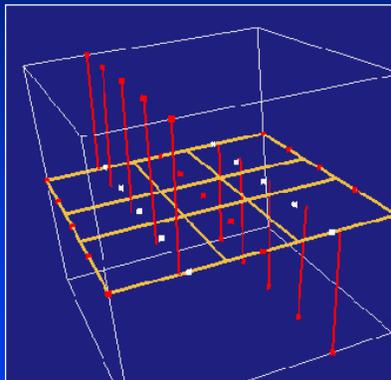
Solid Rounding

Deformation energy rounds a corner



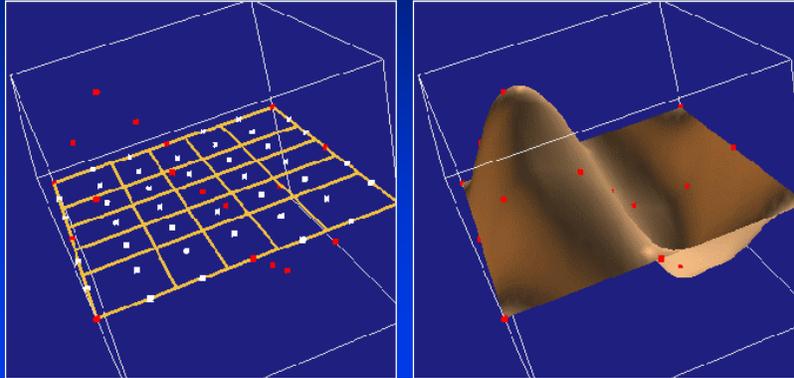
Fitting D-NURBS Surfaces

Scattered data apply constraint forces



Fitting D-NURBS Surfaces

Scattered data apply constraint forces



D-NURBS with Hard Geometric Constraints

Nonlinear constraint $C(\mathbf{p}) = 0$

- Equations of motion

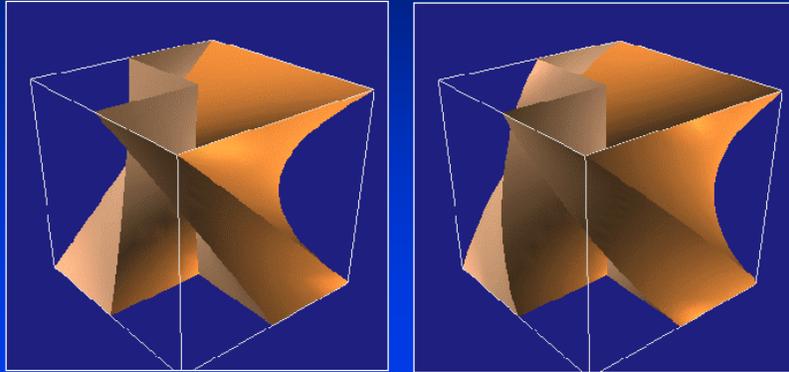
$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}_p - \underbrace{\mathbf{I}\dot{\mathbf{p}} - \mathbf{C}_p^T \boldsymbol{\lambda}}_{\text{generalized constraint forces}}$$

- Dynamics

constraint Jacobian matrix
Lagrange multiplier vector

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}_p^T \\ \mathbf{C}_p & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{p}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\mathbf{D}\dot{\mathbf{p}} - \mathbf{K}\mathbf{p} - \mathbf{I}\dot{\mathbf{p}} + \mathbf{f}_p \\ \sigma - 2a\mathbf{C}_p\dot{\mathbf{p}} - b^2\mathbf{C} \end{bmatrix}$$

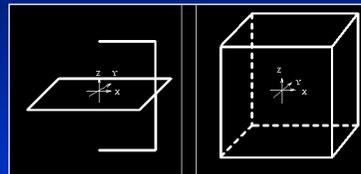
Constrained Skinning



NURBS Swung Surfaces

Geometric NURBS surfaces for cross-sectional design

- Two NURBS generator curves
 - *profile curve on x-z plane*
 - *trajectory curve on x-y plane*
- Symmetries and topological variability
- Broad geometric coverage with $O(m+n)$ DOFs



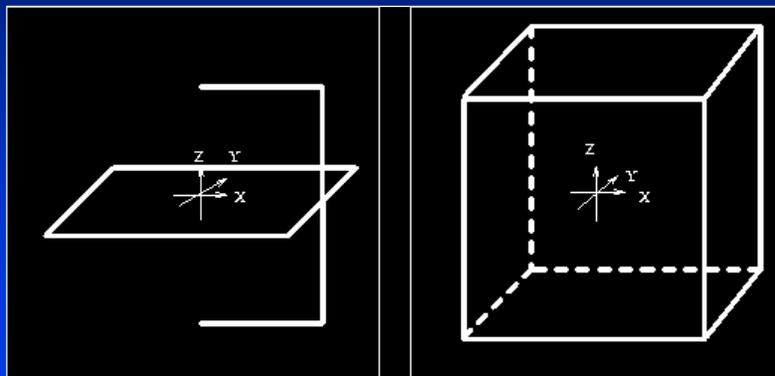
NURBS Swung Surfaces

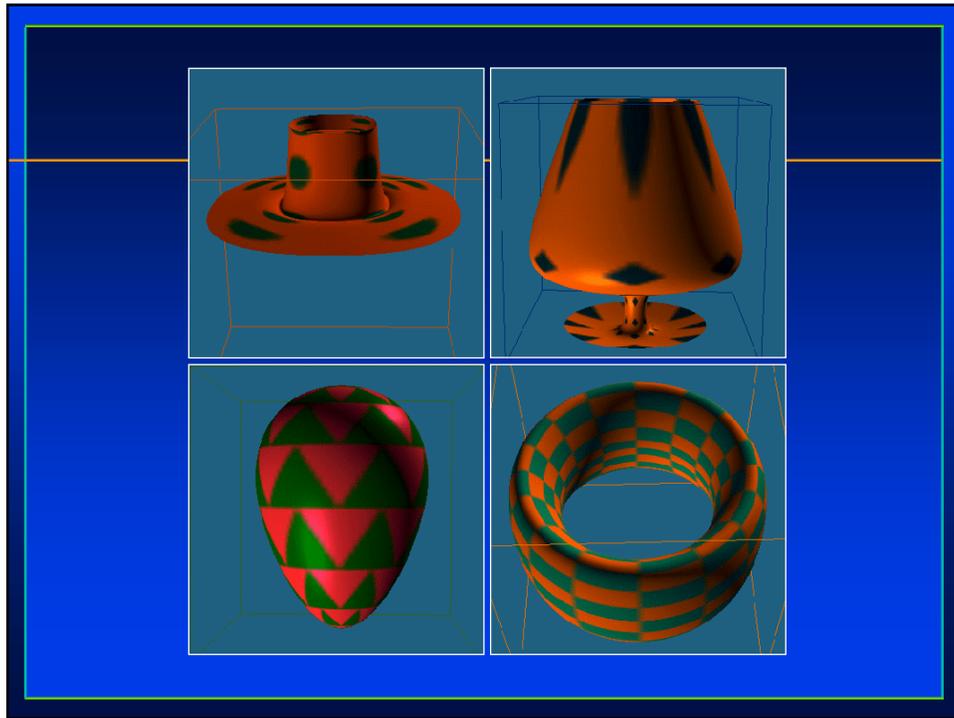
Geometric NURBS surfaces for cross-sectional design

- Two NURBS generator curves
 - a profile curve on x-z plane
 - a trajectory curve on x-y plane
- Symmetries and topological variability
- Broad geometric coverage with $O(m+n)$ DOFs

Swinging Operation

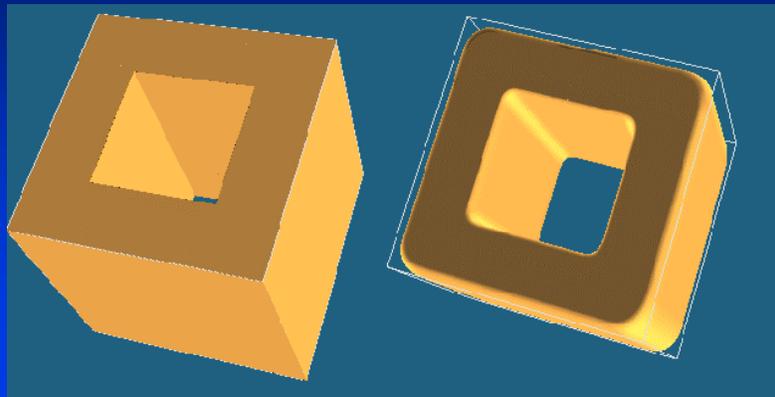
Generating a cube





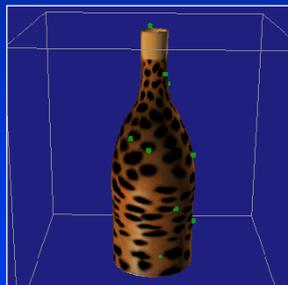
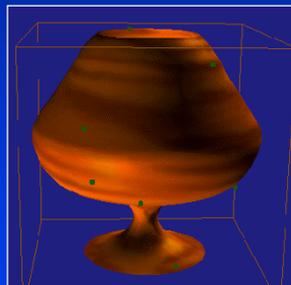
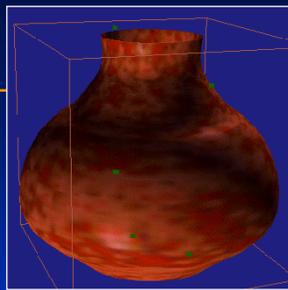
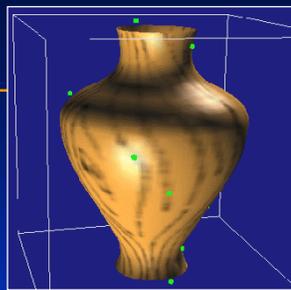
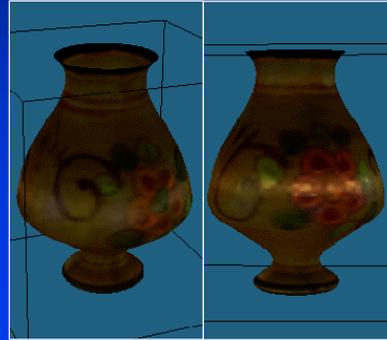
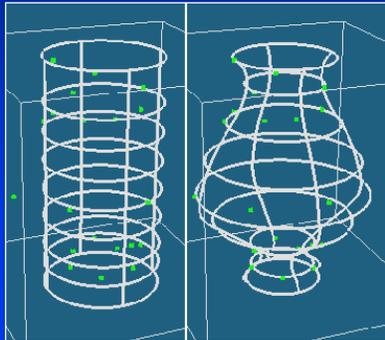
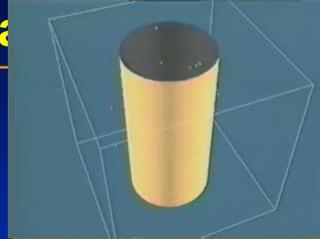
Solid Rounding

Swung D-NURBS solid



Fitting Swung D-NURBS to Scattered Data

NRC pot



D-NURBS Family

D-NURBS curves

Tensor-product D-NURBS surfaces

Swung D-NURBS surfaces

Triangular D-NURBS surfaces

Rational Gaussian (RaG) Curves and Surfaces

Ardy Goshtasby

Computer Science and Engineering
Wright State University

Overview

- ① RaG formulation
- ② Curves for design
- ③ Curves for data fitting
- ④ Surfaces for design
- ⑤ Surfaces for data fitting
- ⑥ Surface-fitting to severely irregular points

RaG Formulation

$P(\mathbf{u})$: Point on shape at \mathbf{u}

V_i : i th control point

$g_i(\mathbf{u})$: i th basis function

$G_i(\mathbf{u})$: Gaussian centered at u_i

u_i : i th node

W_i : i th weight

σ_i : Standard deviation of i th Gaussian

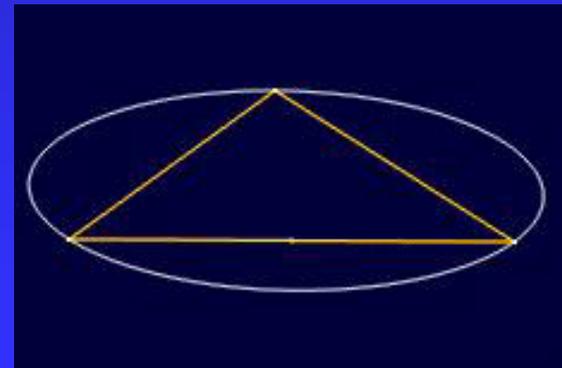
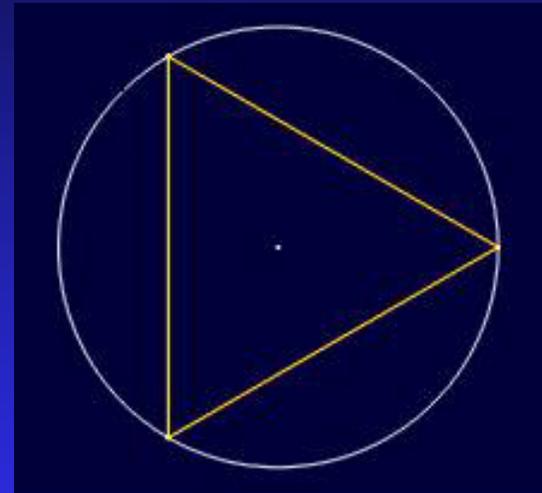
$$P(\mathbf{u}) = \sum_{i=1}^n V_i g_i(\mathbf{u})$$

$$g_i(\mathbf{u}) = \frac{G_i(\mathbf{u})}{\sum_{j=1}^n G_j(\mathbf{u})}$$

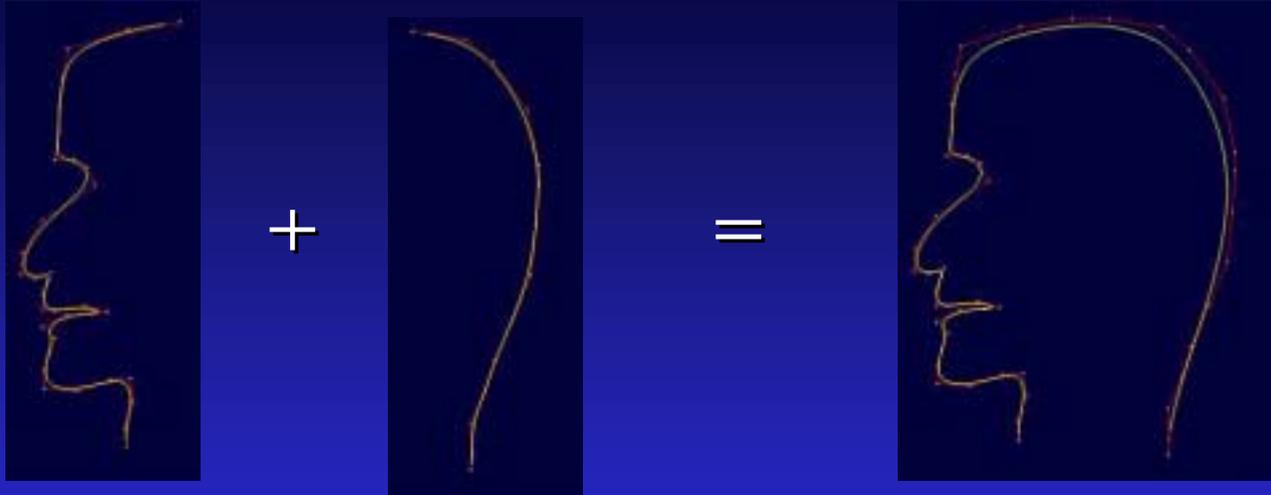
$$G_i(\mathbf{u}) = W_i \exp\left(-\frac{(\mathbf{u} - \mathbf{u}_i)^2}{2\sigma_i^2}\right)$$

Curves for Design

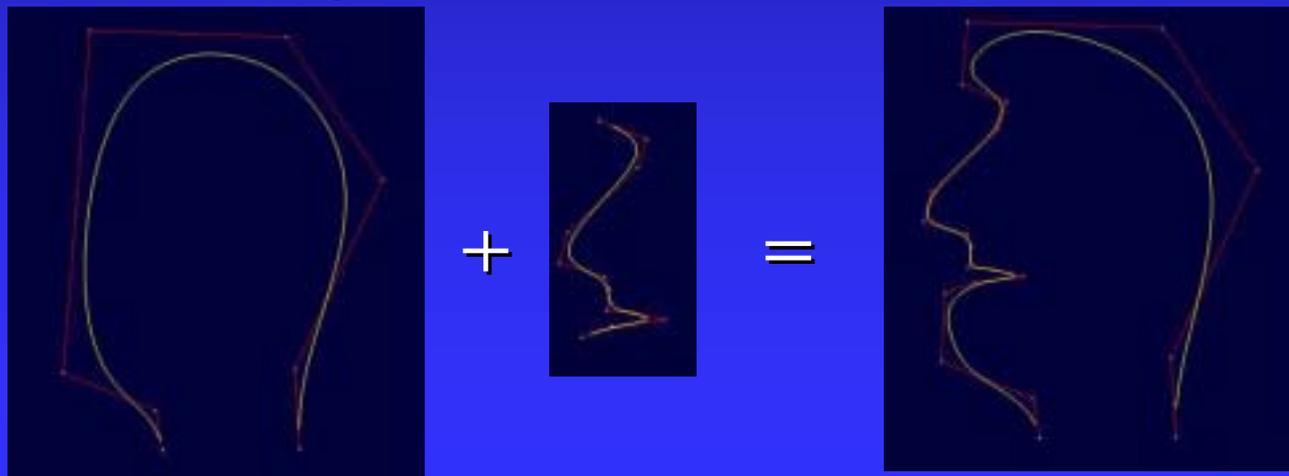
- A circle is obtained by fitting a RaG curve to vertices of an equilateral triangle.
- An ellipse is obtained by fitting a RaG curve to vertices of any triangle.



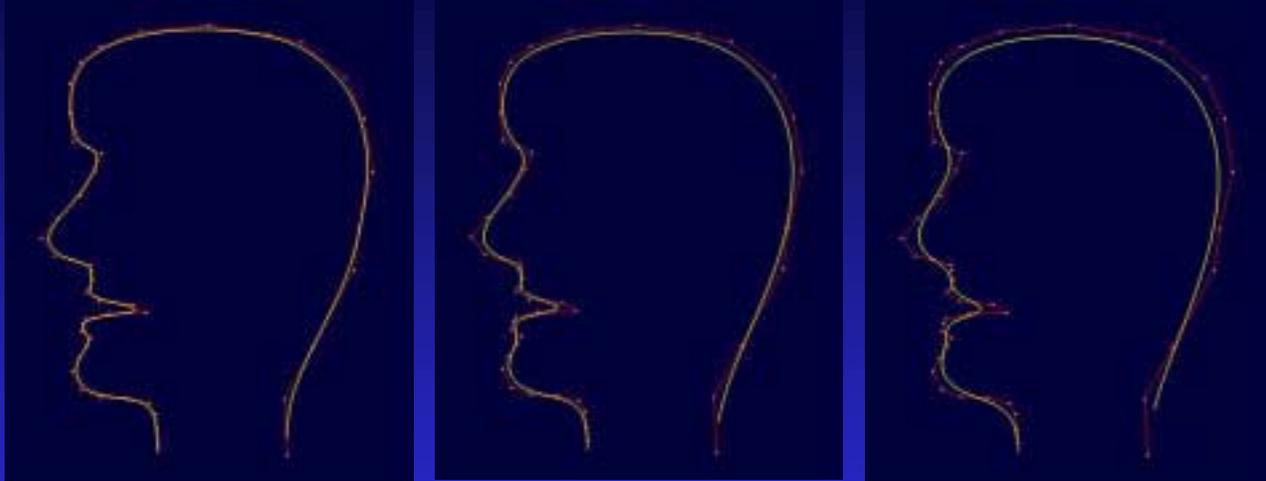
- A composite curve: $P(w) = P(u) + P(v)$



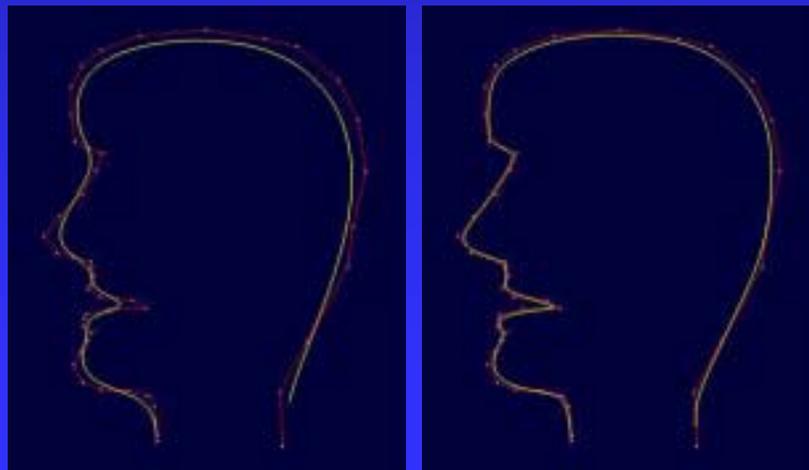
- A blending curve: $P(w) = P(u) + P(v)$



■ Smoothness parameters: σ_i 's

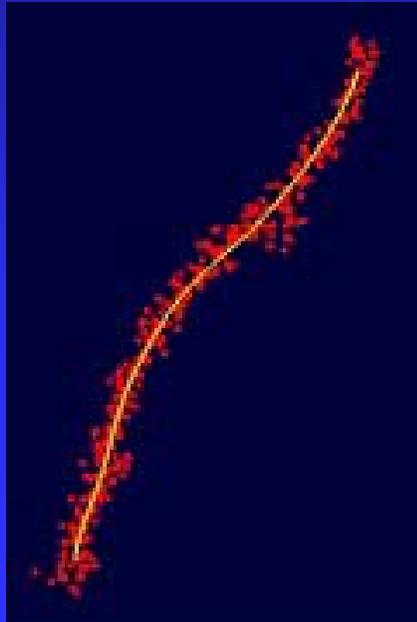
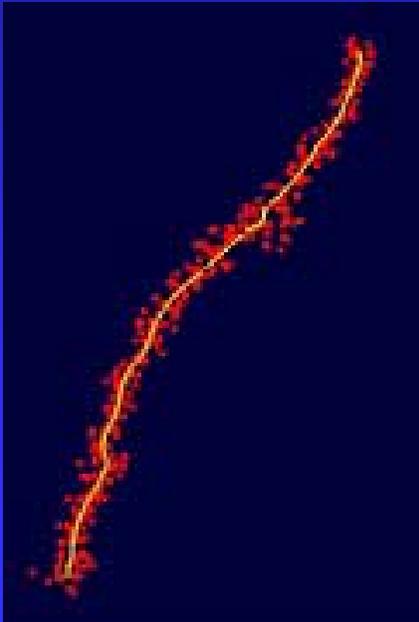


■ Weights: W_i 's



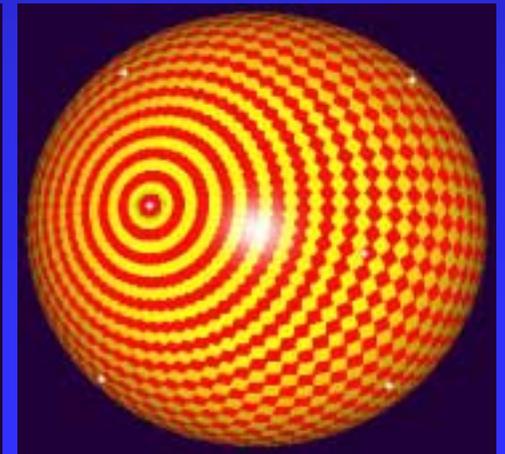
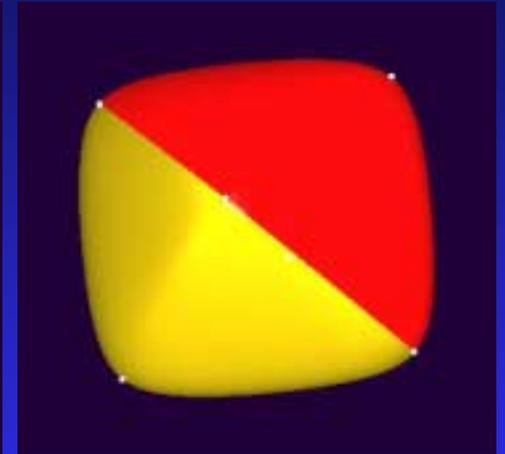
Curves for Data Fitting

- Points can be dense and irregularly spaced.
- Points can be noisy.

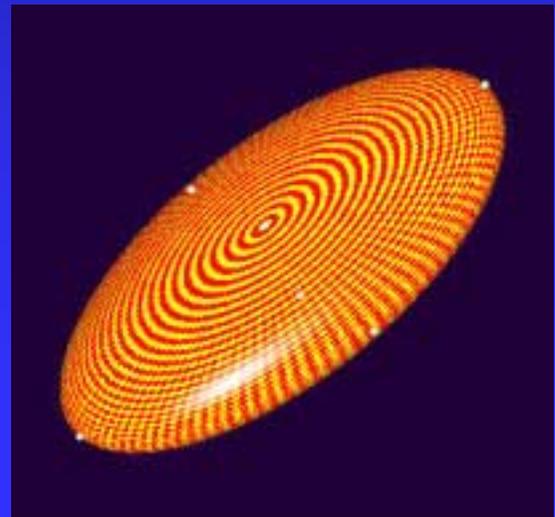
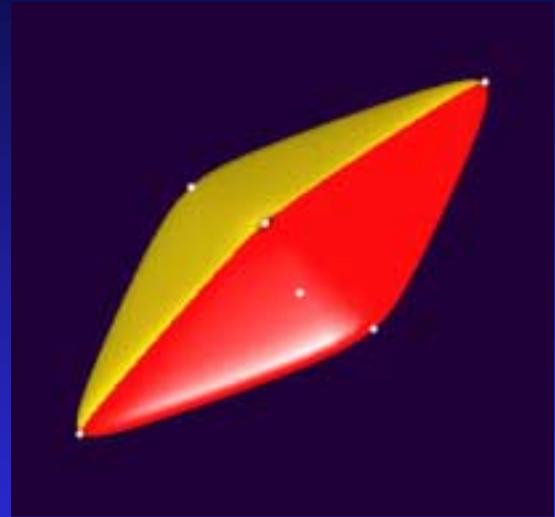
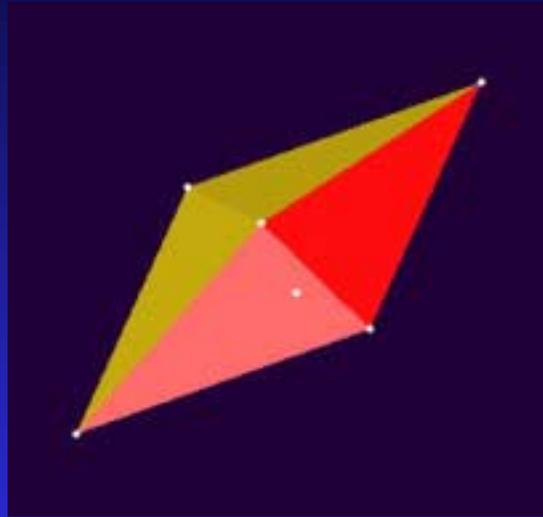


Surfaces for Design

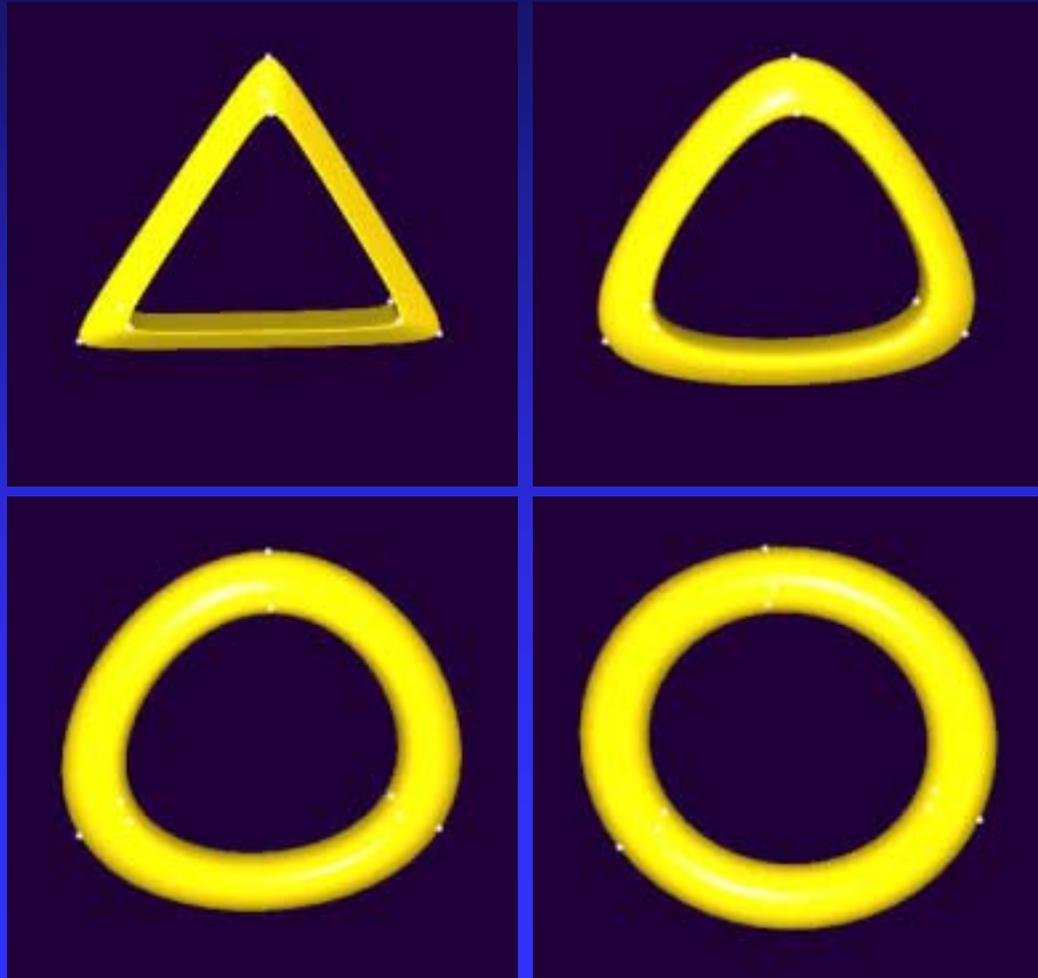
- A sphere is obtained by fitting a RaG surface to 6 points.



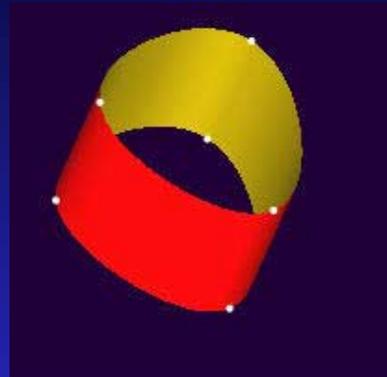
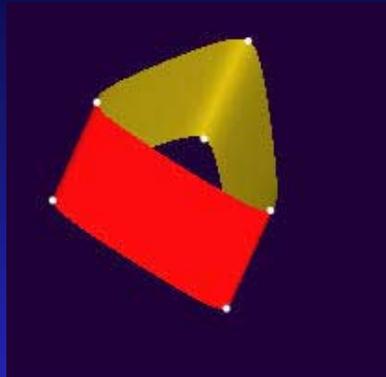
- An ellipsoid is obtained by fitting a RaG surface to 6 points.



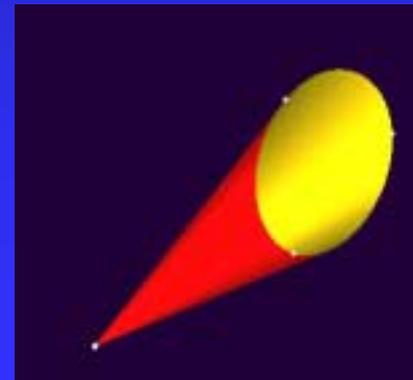
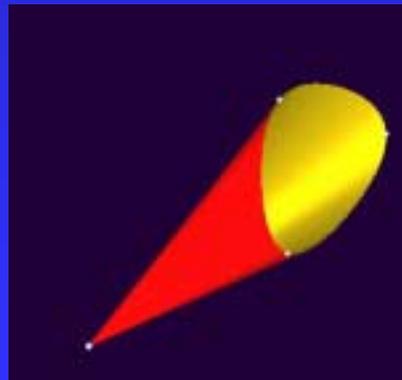
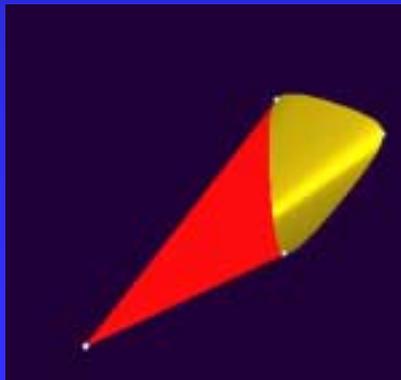
- A torus is obtained by fitting a RaG surface to 9 point.



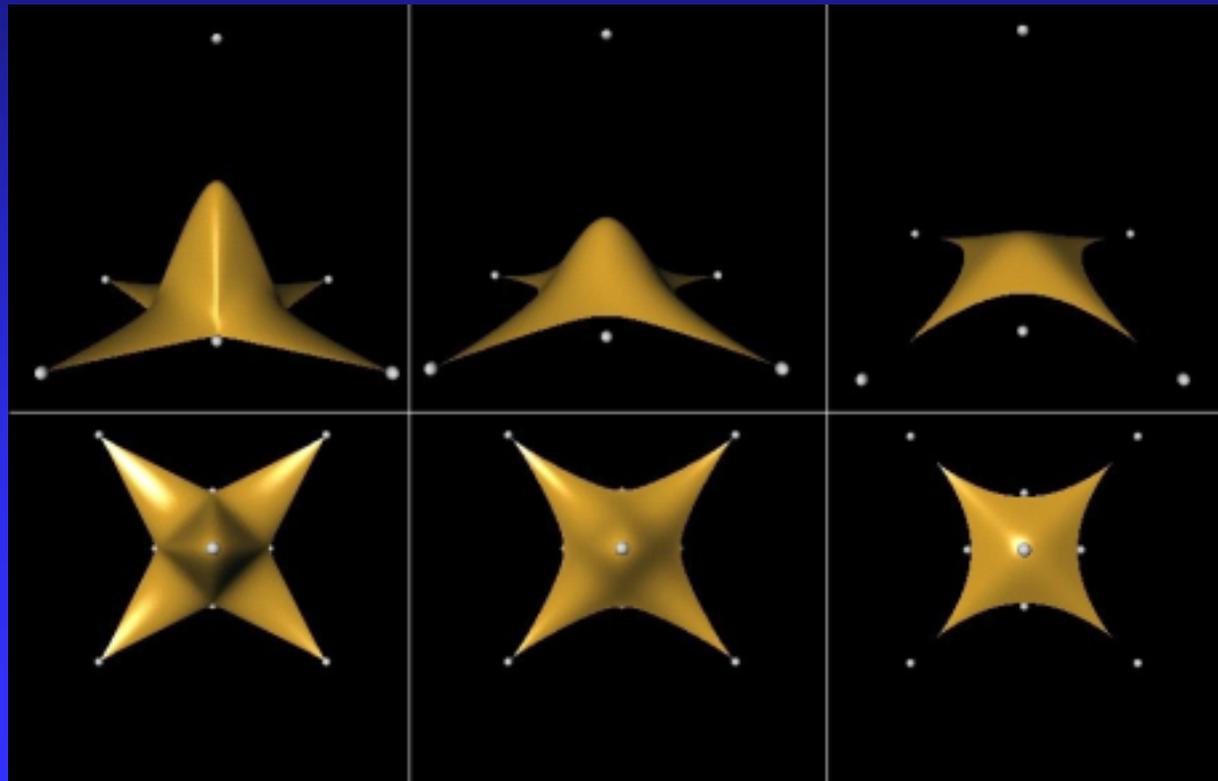
- A cylinder is obtained by fitting a RaG surface to 6 points.



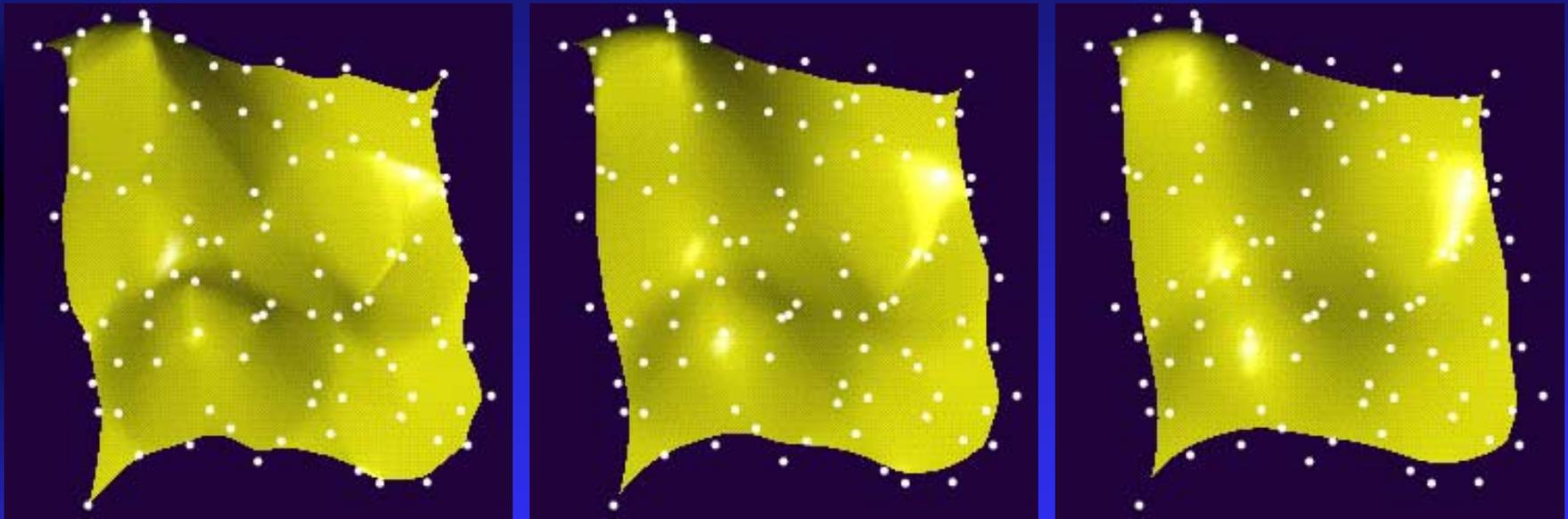
- A cone is obtained by fitting a RaG surface to 4 points.



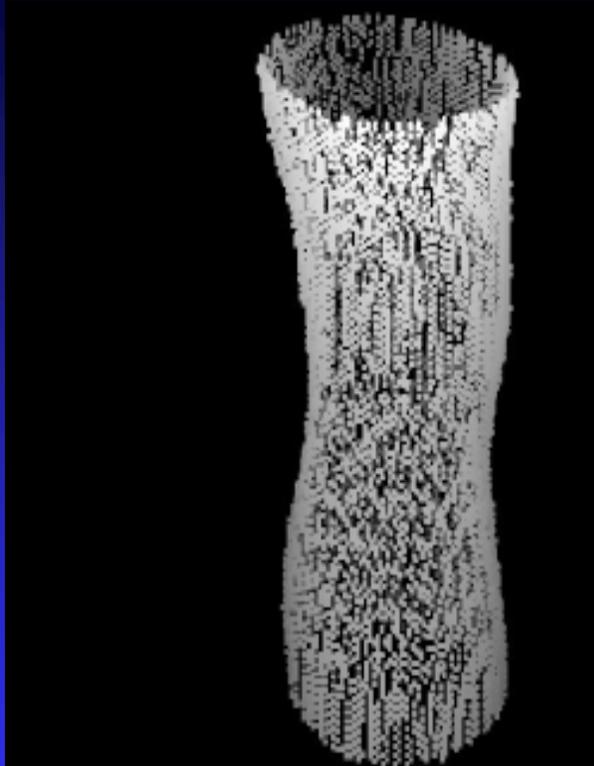
- A free-form shape is obtained by fitting a RaG surface to an arbitrary number of points in an arbitrary arrangement.



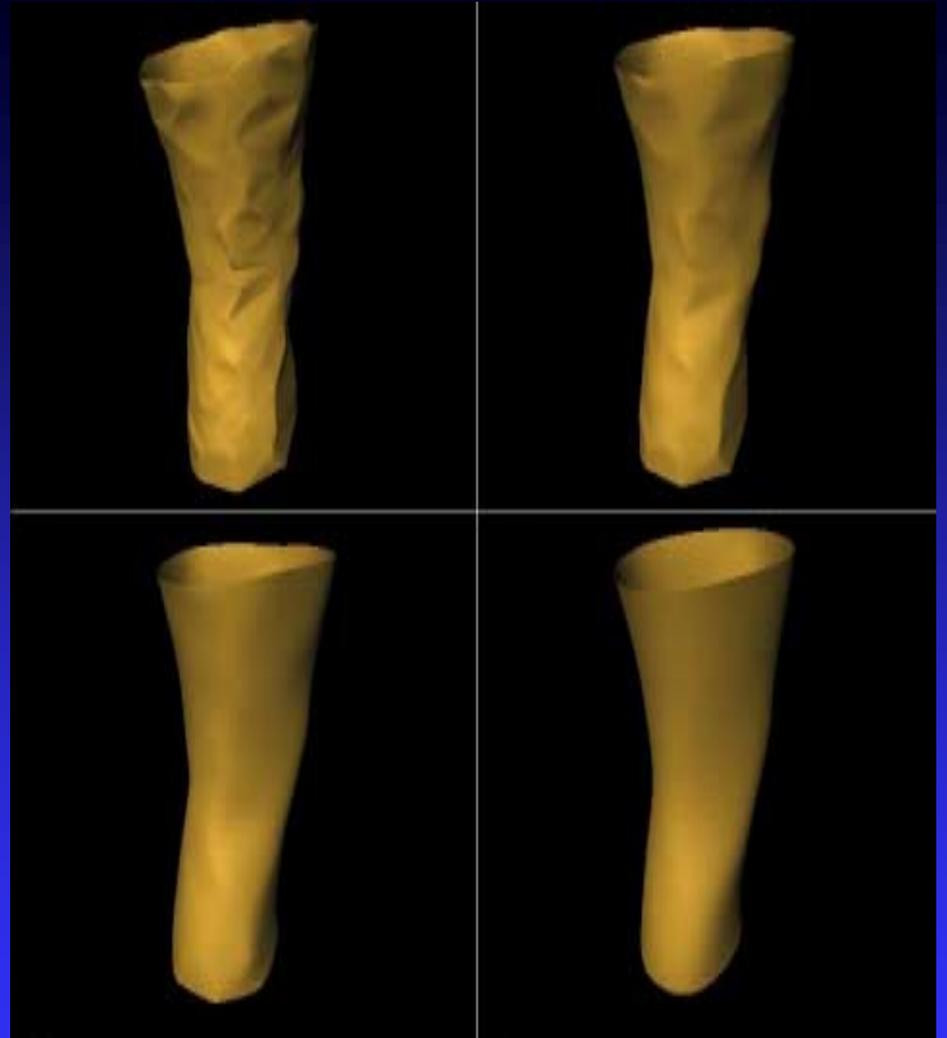
Surfaces for Data Fitting



RaG surfaces fitting to 100 irregularly spaced points with different smoothness levels.



Laser range data

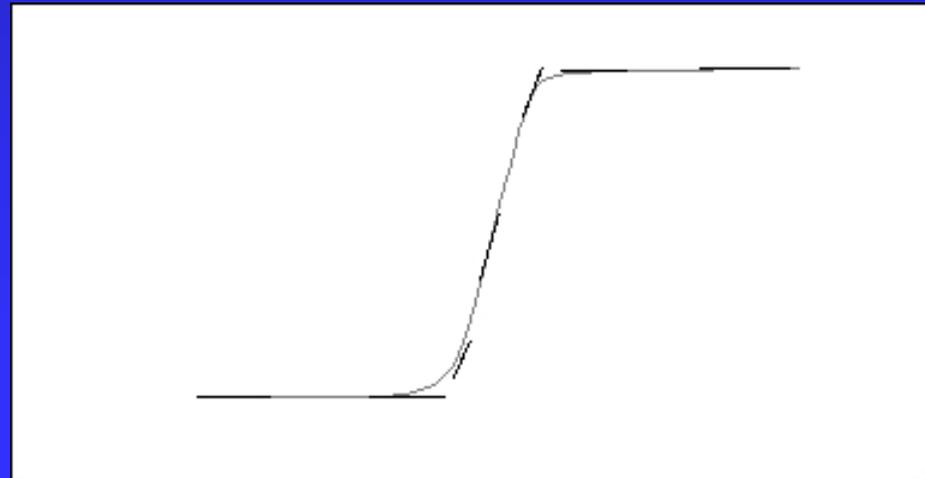
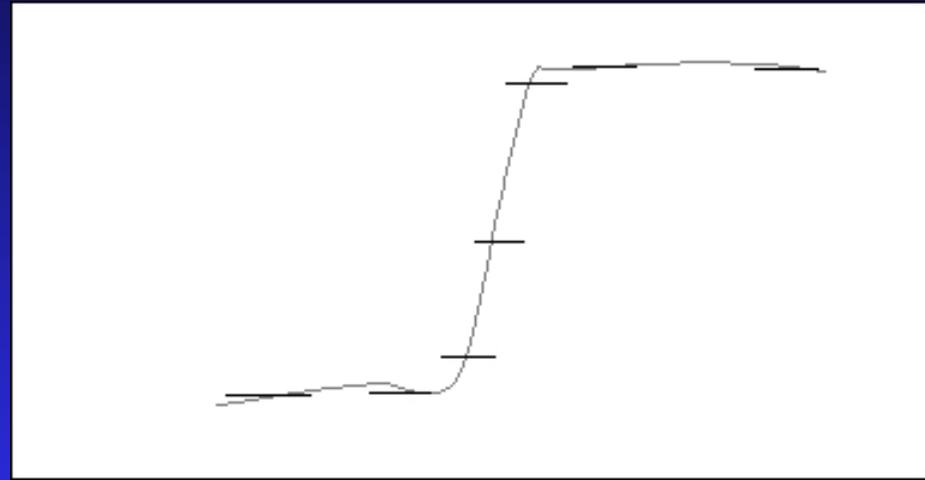


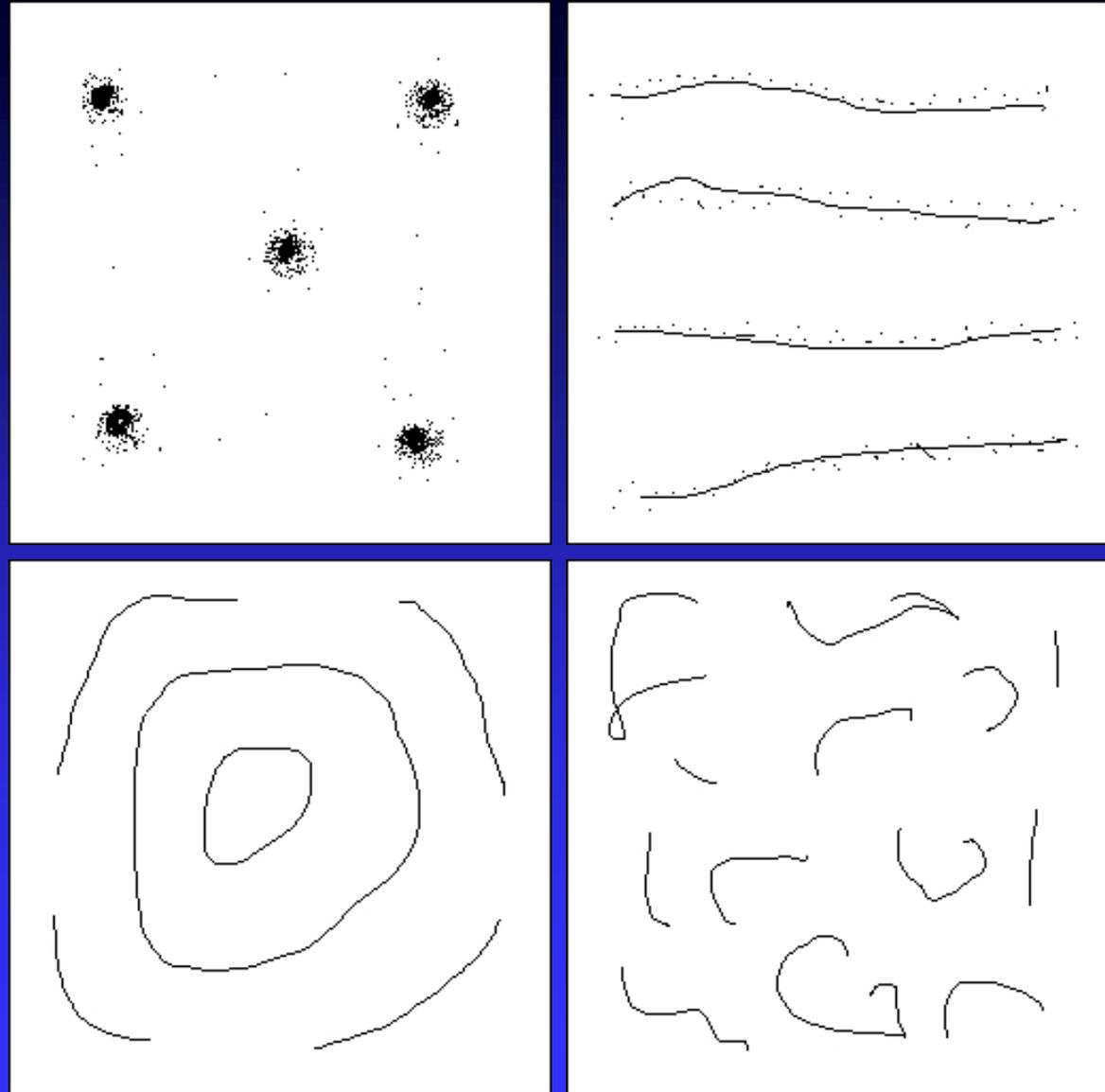
RaG surfaces with different σ 's fitting the same data set.

Surface-Fitting to Severely Irregular Points

The basic idea:

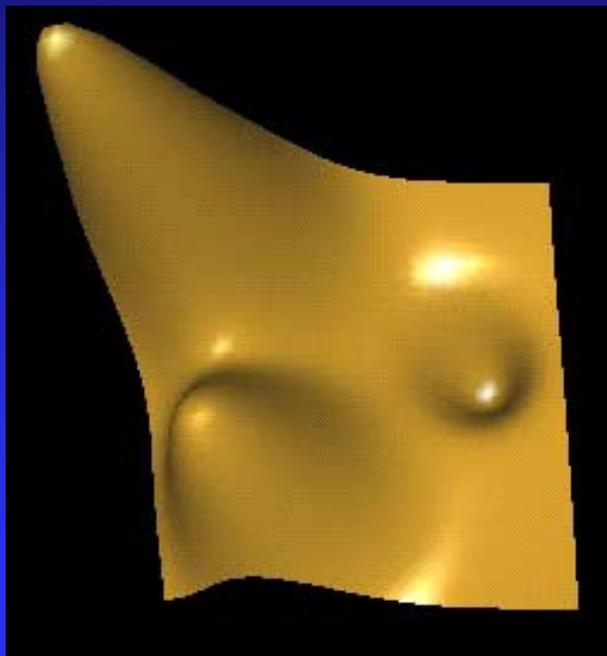
Instead of using fixed control points, use functions that can reproduce desired derivatives at the points.



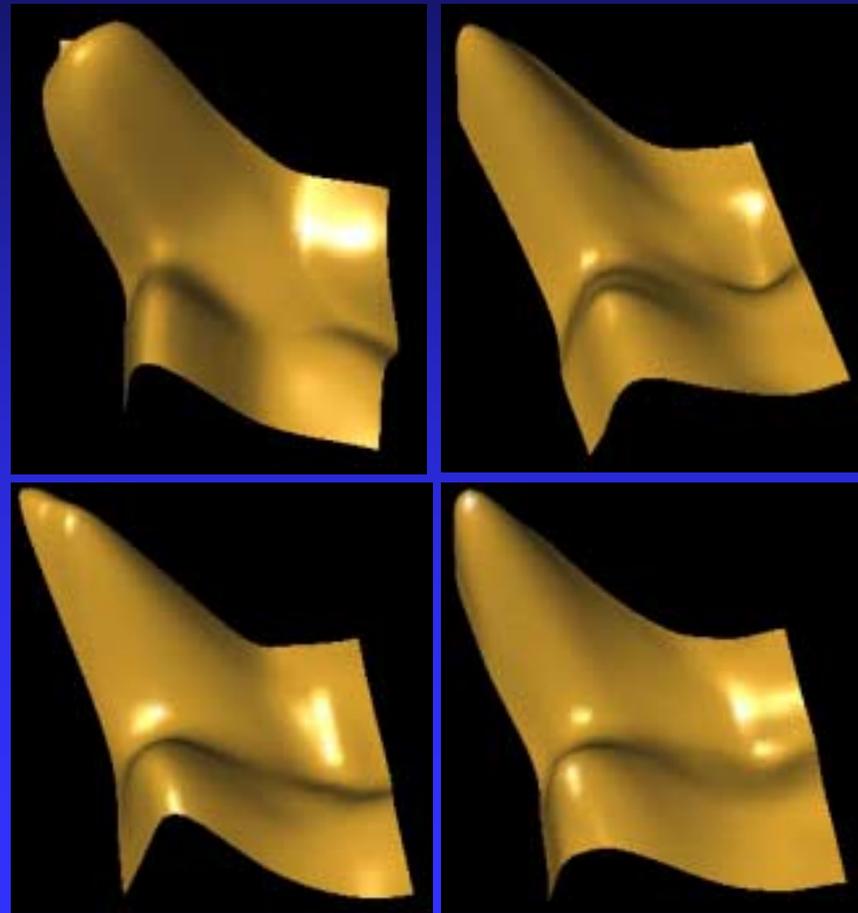


Four severely irregular point sets

Examples

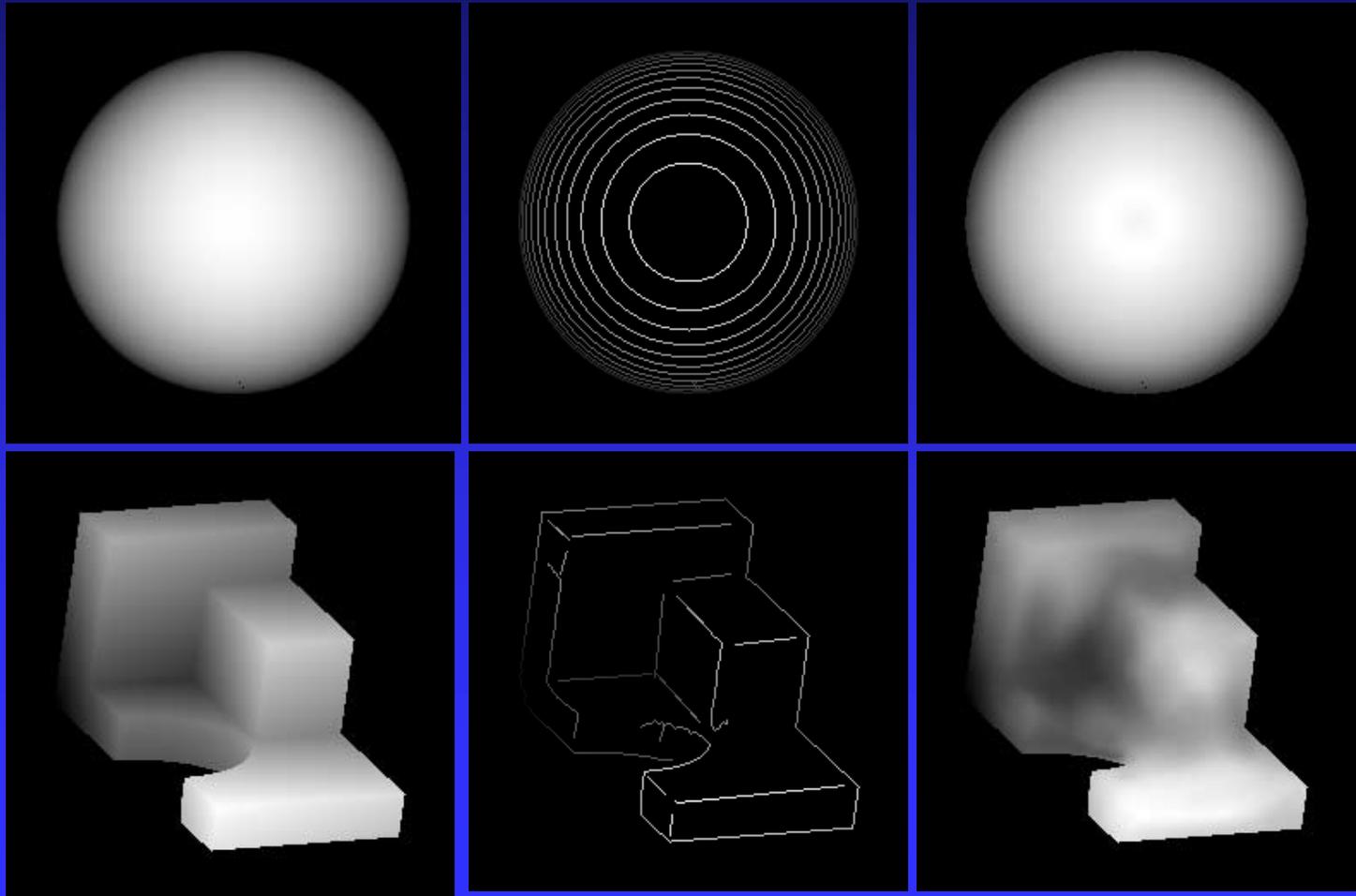


Original



Estimated from the 4 data sets

More Examples



Conclusions

- RaG curves and surfaces provide capabilities similar to those of NURBS curves and surfaces.
- Unlike NURBS, RaGs do not require a regular grid of control points. So they are ideal for approximating/interpolating irregularly spaced points.
- In addition, RaGs provide an effective means to produce shapes at varying levels of resolution from the same data set.

Grouping and Parametrizing Points for Curve and Surface Fitting

Ardy Goshtasby

Computer Science and Engineering

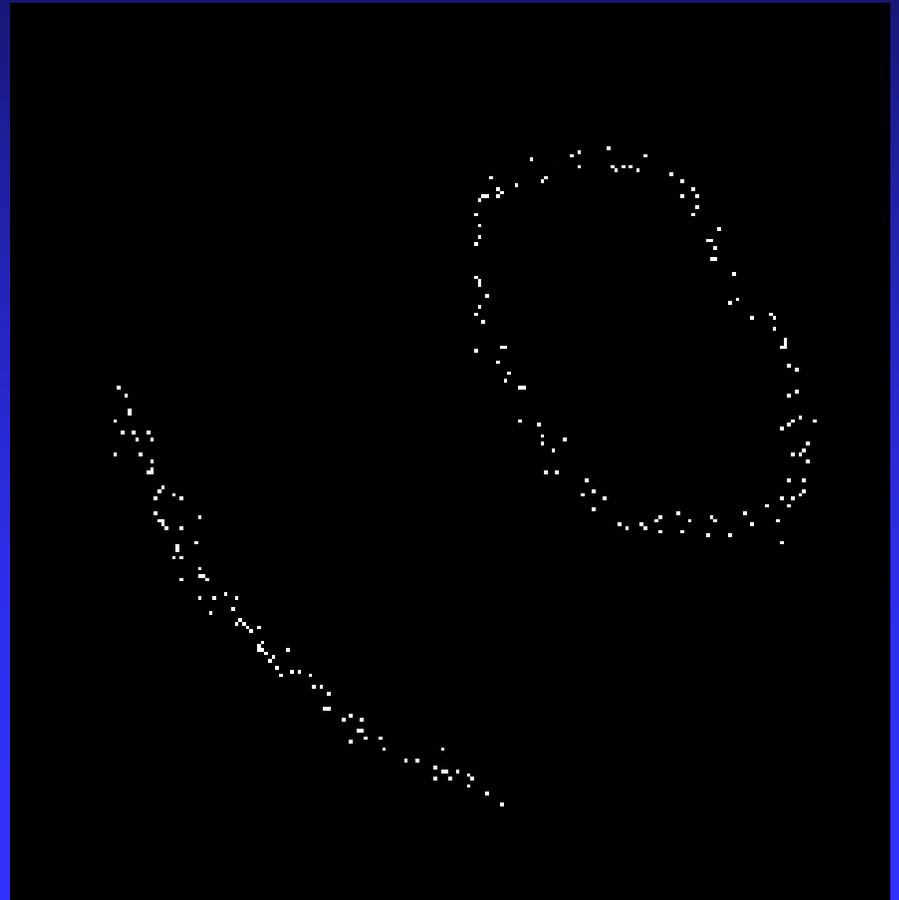
Wright State University

Overview

- ① Parametrization for curve fitting
 - Grouping
 - Parametrization
 - The resolution problem
- ② Parametrization for surface fitting
 - Grouping
 - Parametrization
 - Subdivision

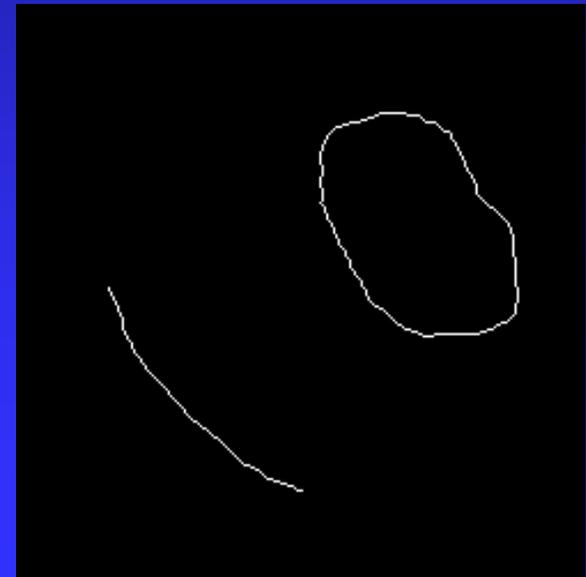
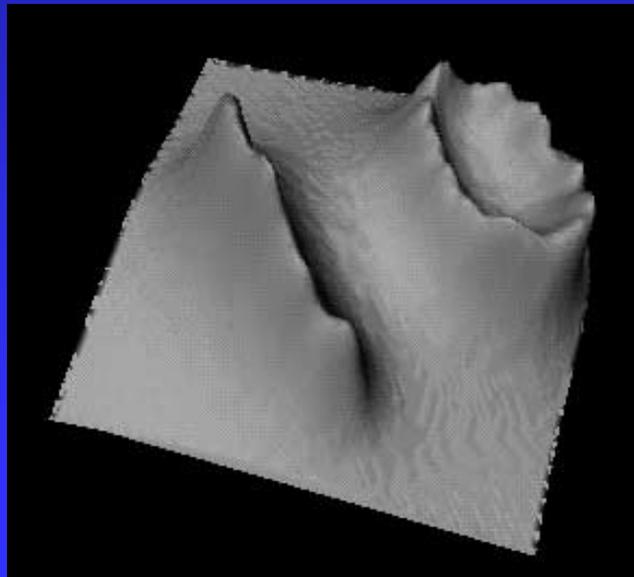
Grouping for Curve Fitting

How many shapes
are hidden among
these points?

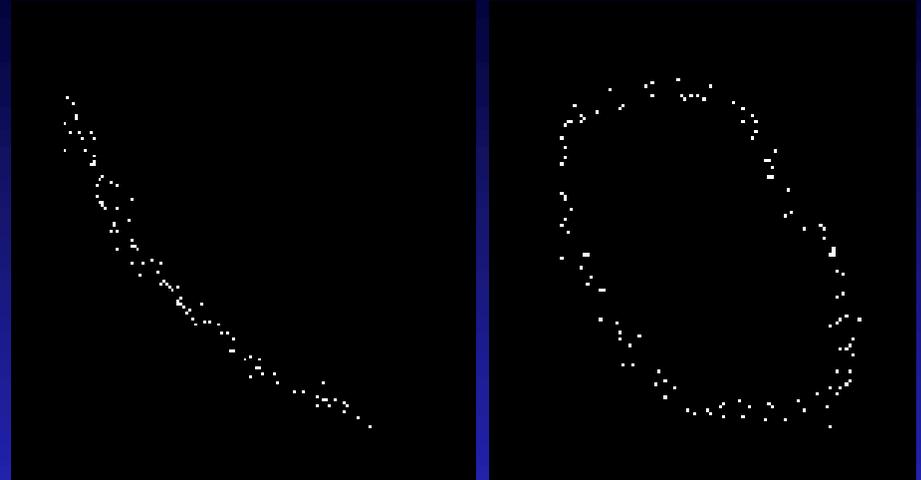


The Grouping Algorithm

1. Center a 2-D monotonically decreasing radial basis function at each point.
2. Find points with locally maximum values.

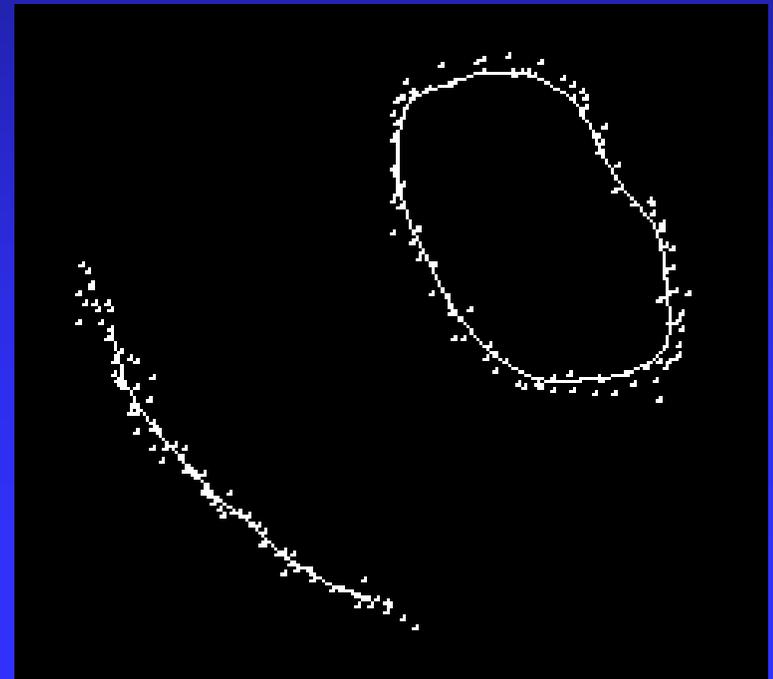


3. Associate a point to the structure that is closest to it.



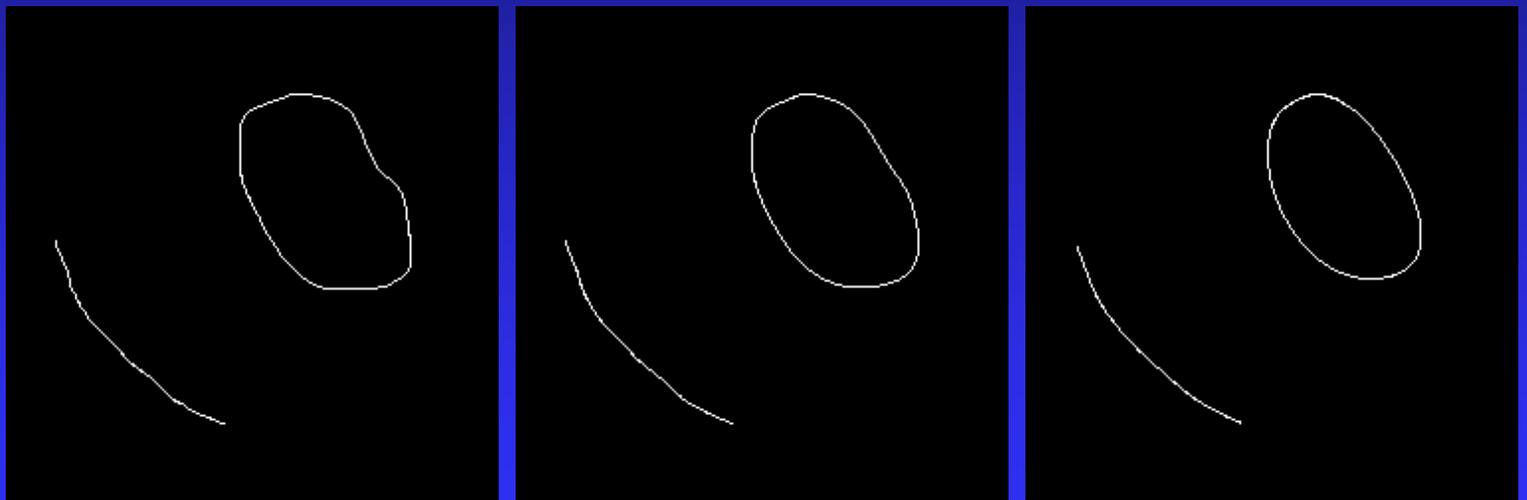
4. Parametrize points in each subset.

3. Fit a curve to each subset.



The Resolution Problem

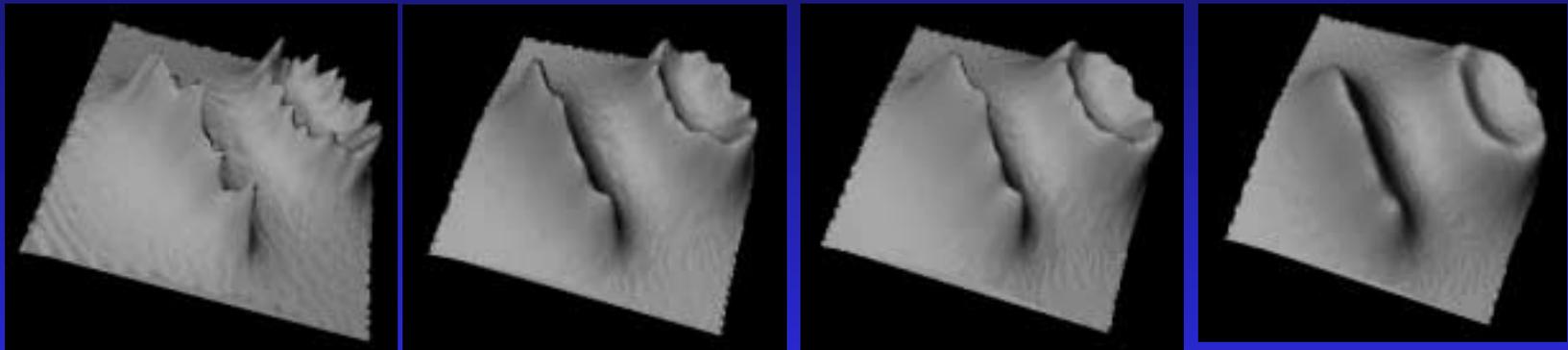
Finer



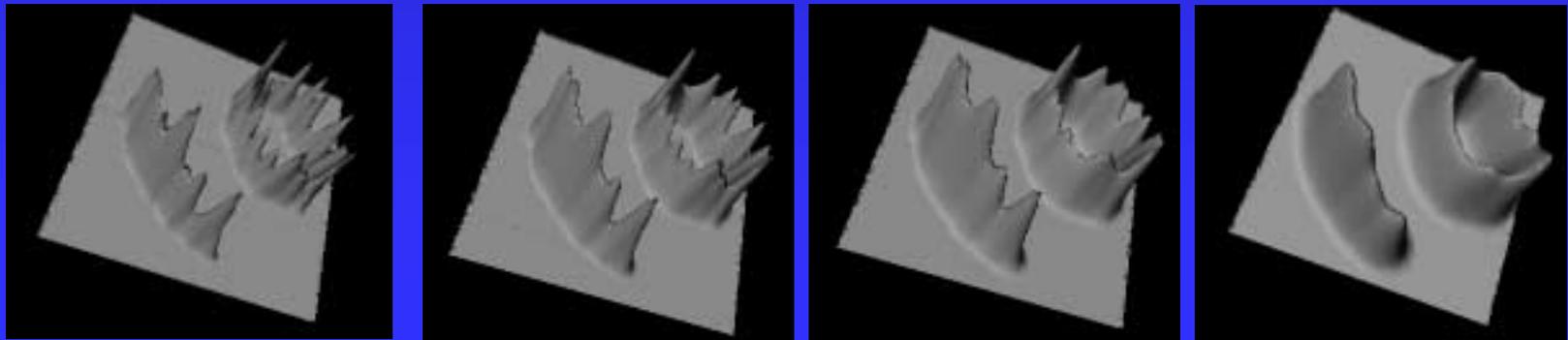
Coarser

The Choice of the Basis Function

Using multiquadric bases:



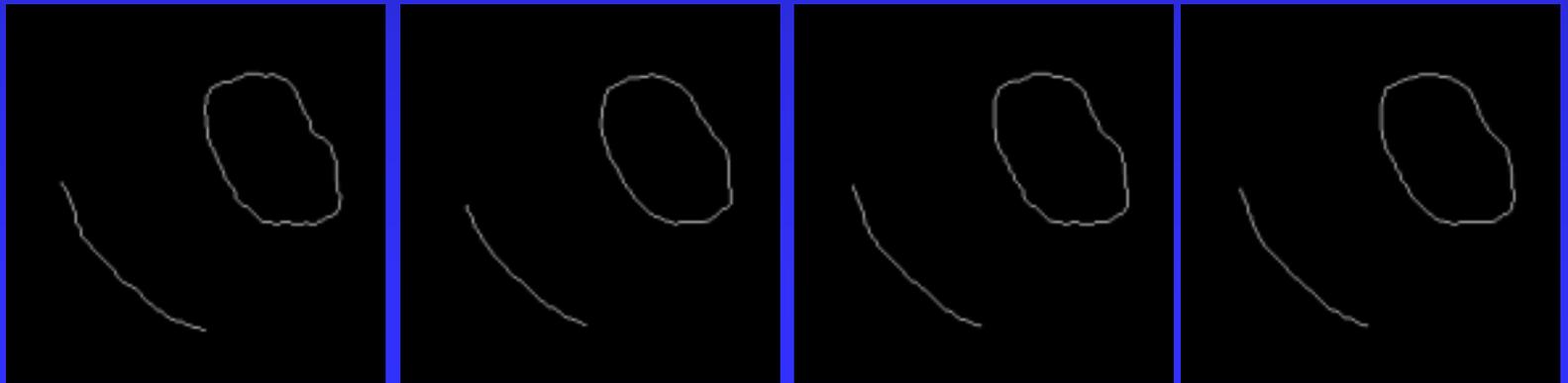
Using Gaussian bases:



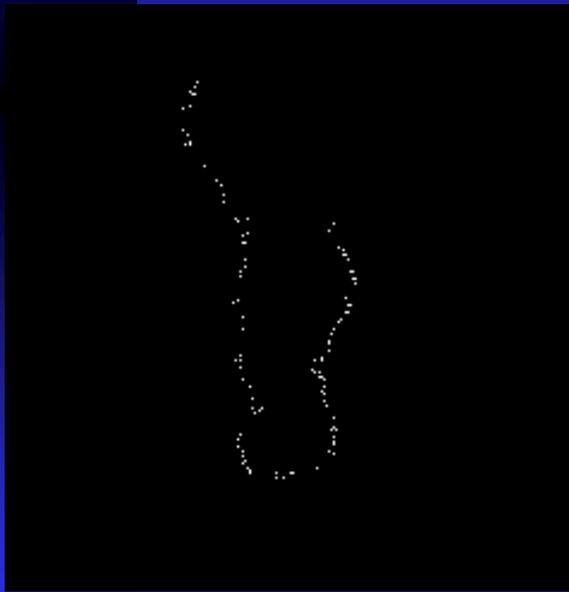
Multiquadric:



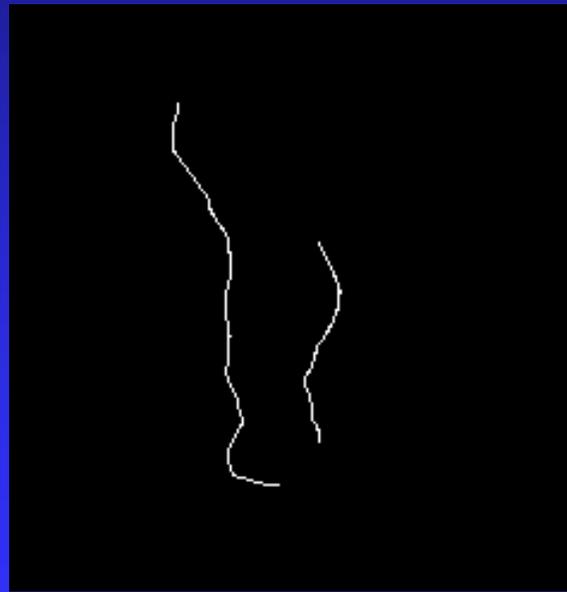
Gaussian:



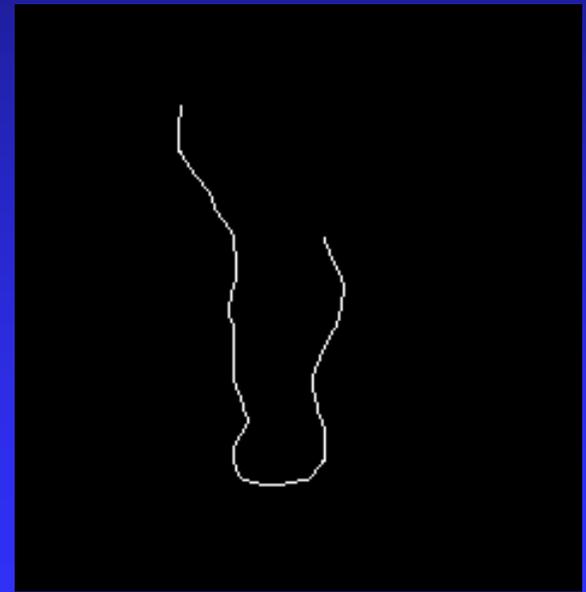
Effect of Basis Functions and Resolution



Point set



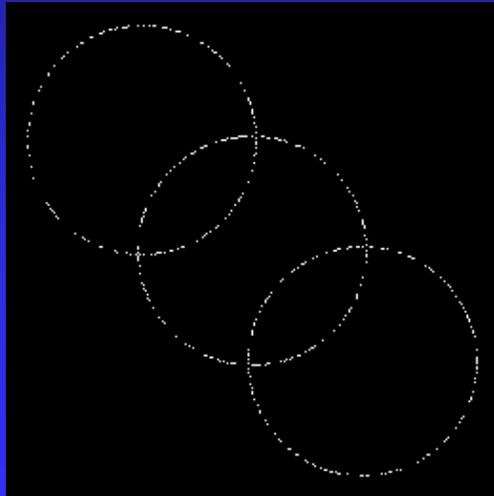
Multiquadric



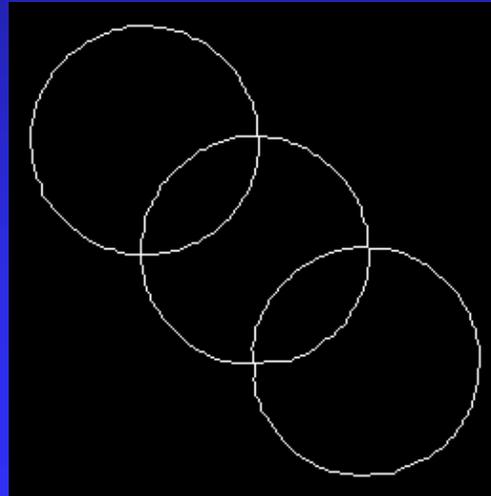
Gaussian

Complex Point Sets

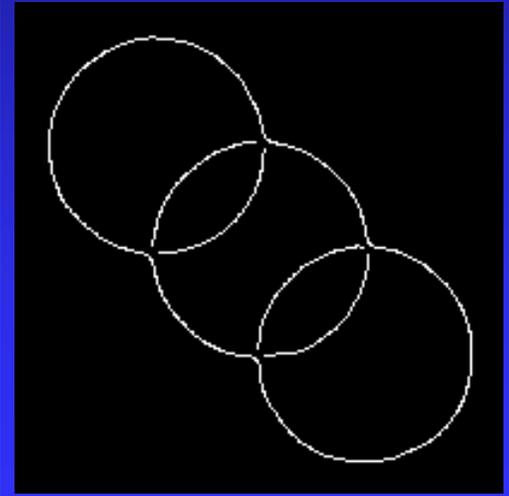
Subdivide overlapping subsets according to a predefined criterion.



Point set



Overlapping subsets



Obtained curves

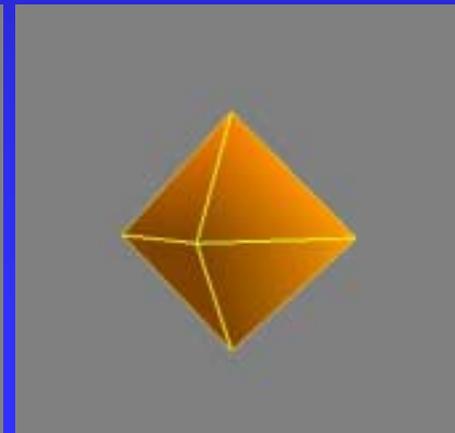
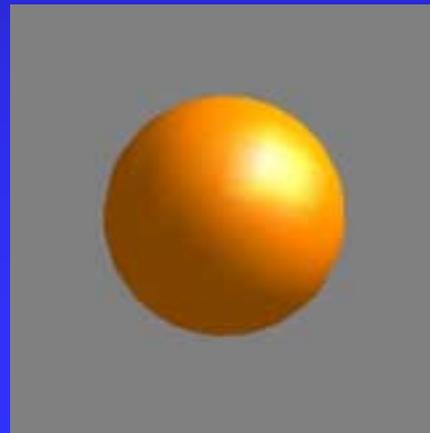
Grouping for Surface Fitting

1. Center a 3-D monotonically decreasing radial basis function at each point.
2. Find regions with values above a threshold value.
3. Associate a point to the region closest to it.
4. Parametrize the points in each subset.
5. Fit a surface to the points in each subset.

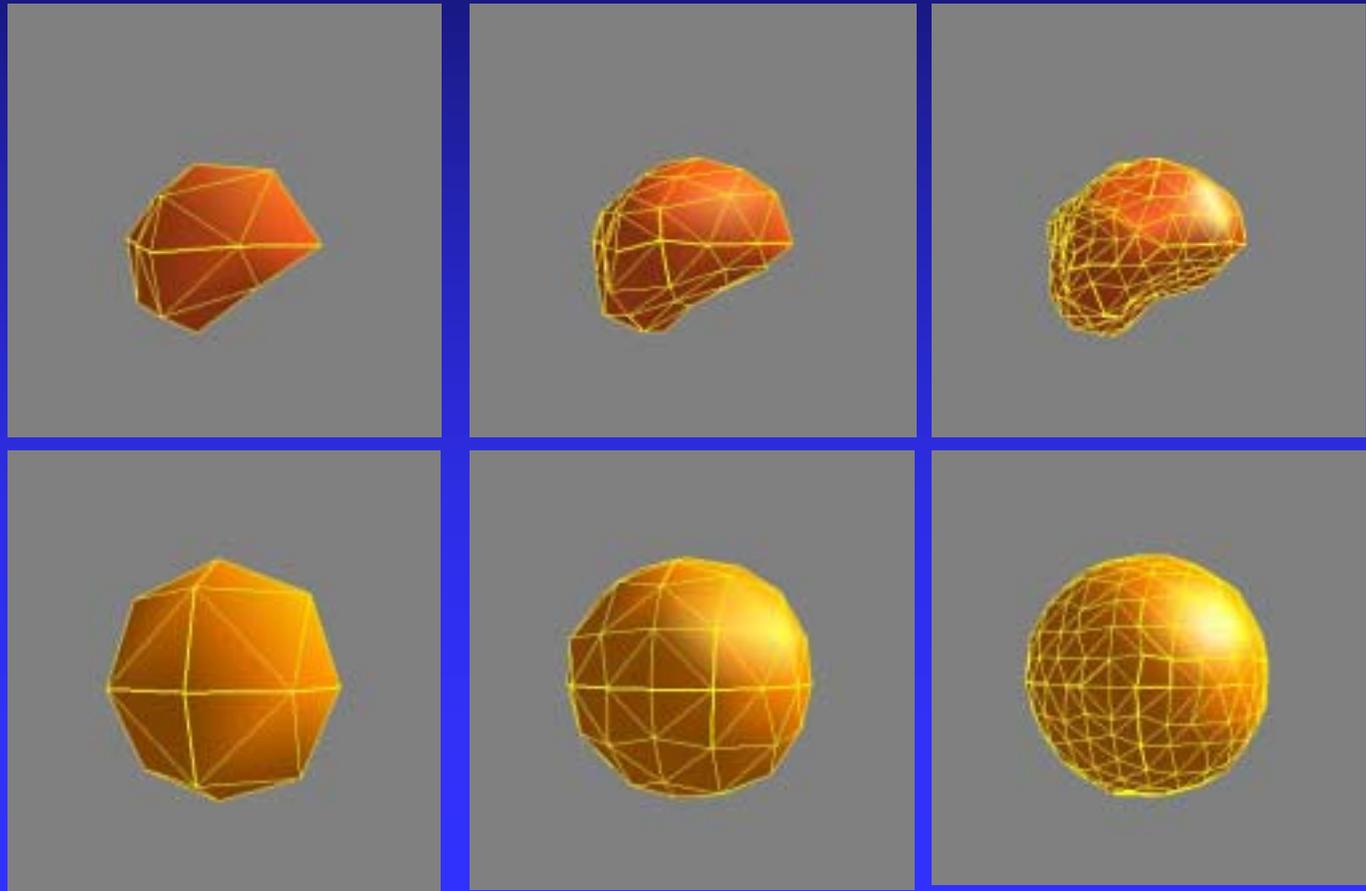
Parametrization: Spherical

1. Start by approximating the points with an octahedron.

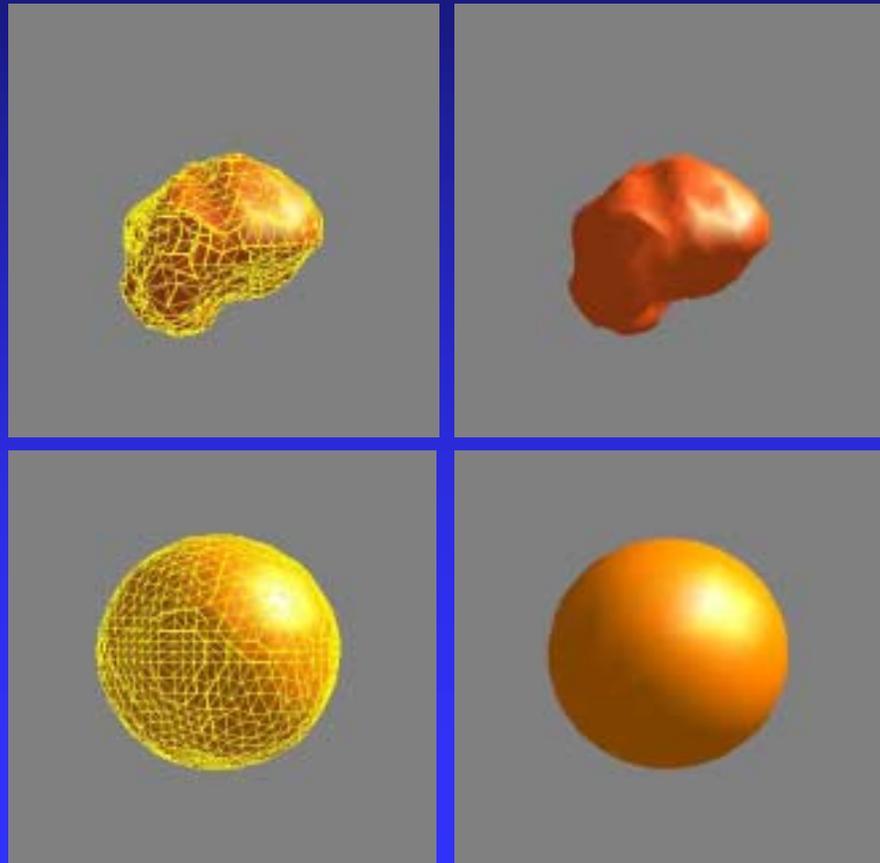
Similarly
approximate the
sphere with an
octahedron.



2. If error between a triangle and the shape is above a tolerance, subdivide the triangle. Similarly subdivide the corresponding triangle in sphere.

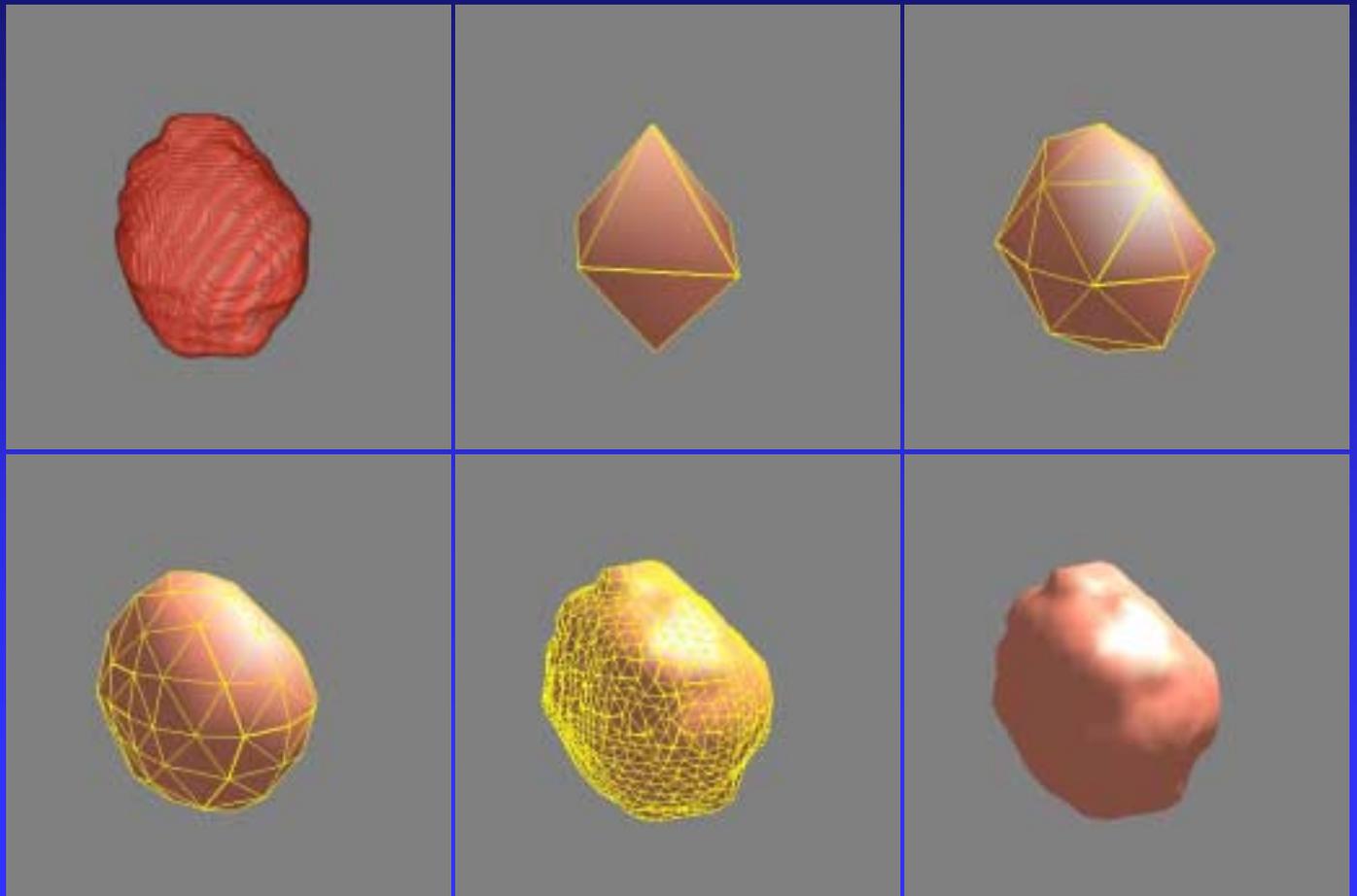


3. Repeat the subdivision until approximation error between all triangles and the shape reaches the required tolerance.

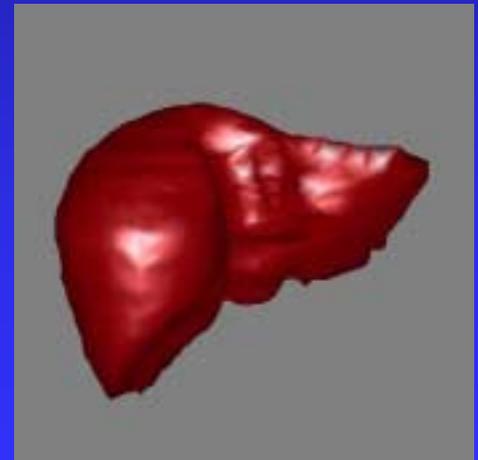
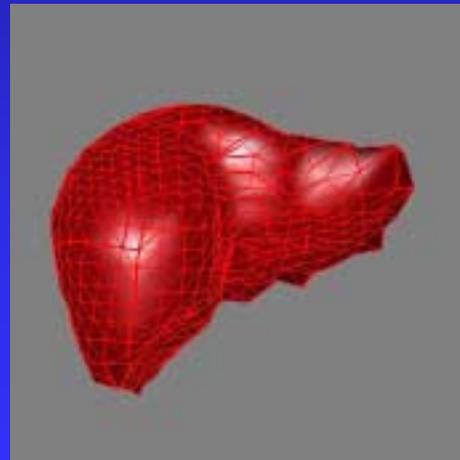
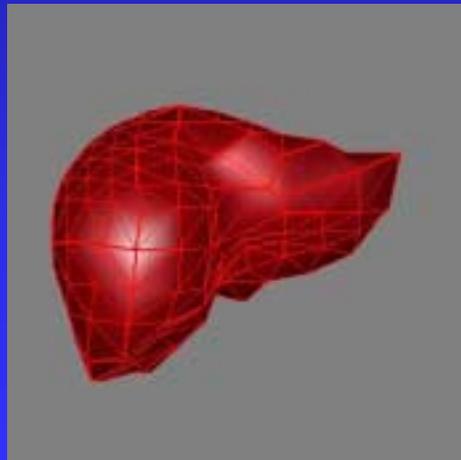
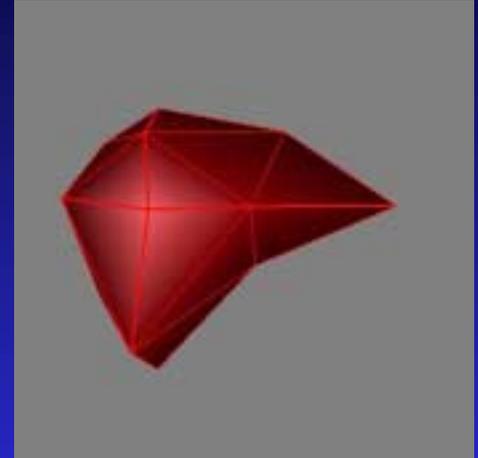
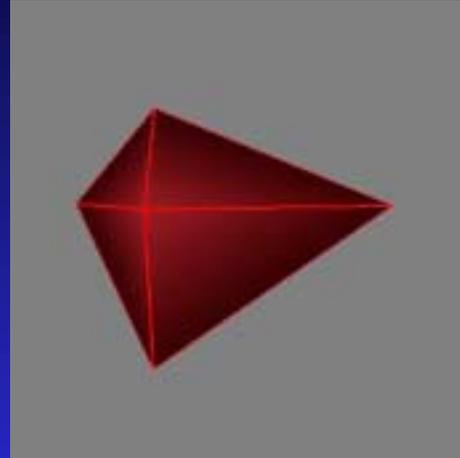
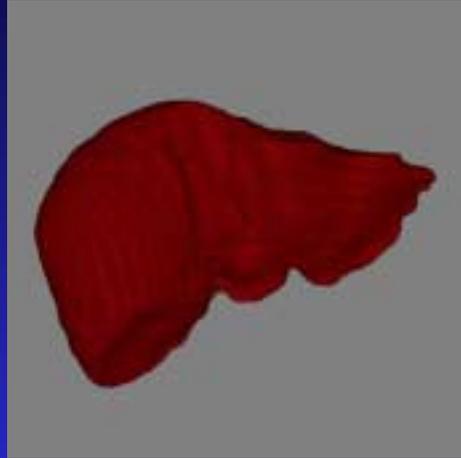


Subdivision Examples

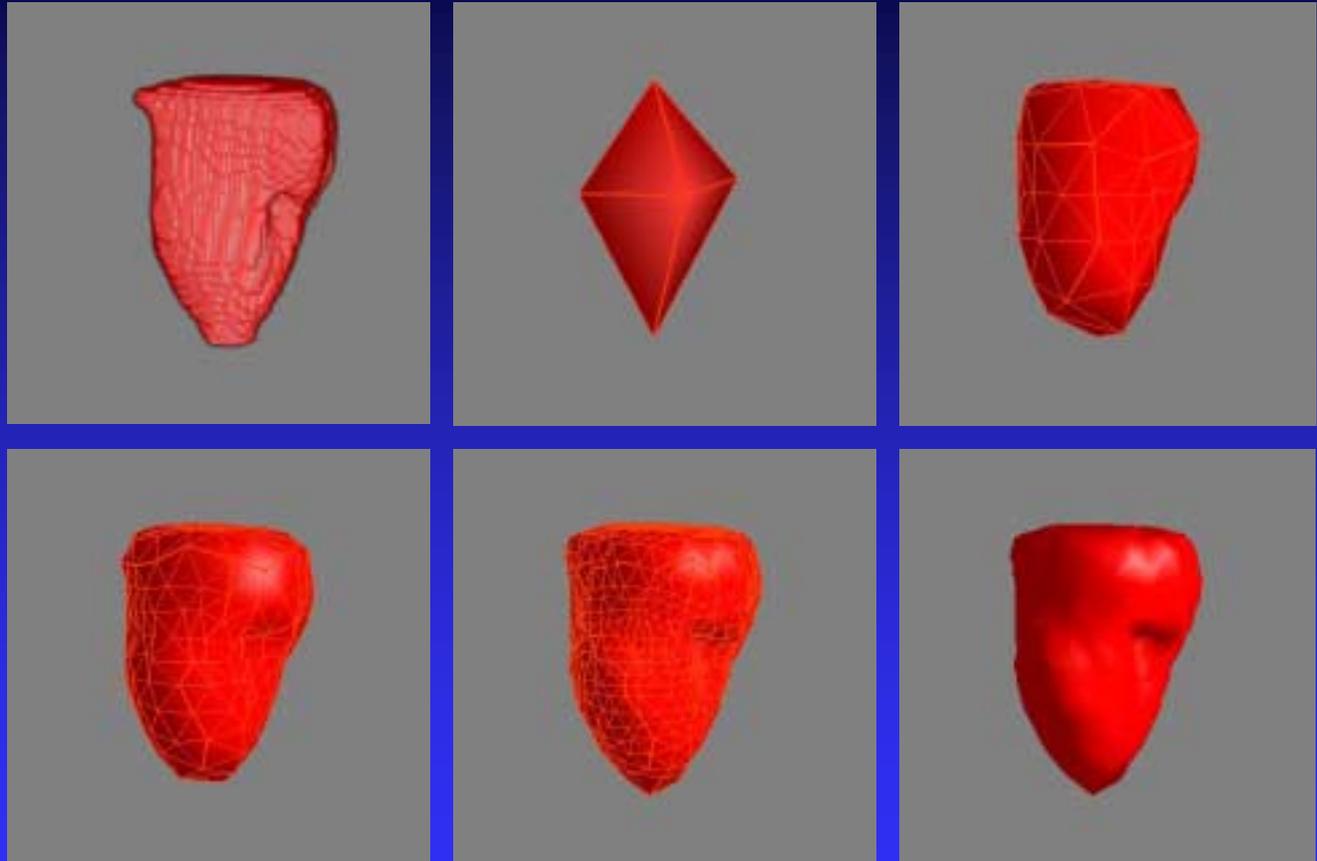
1. Liver tumor:



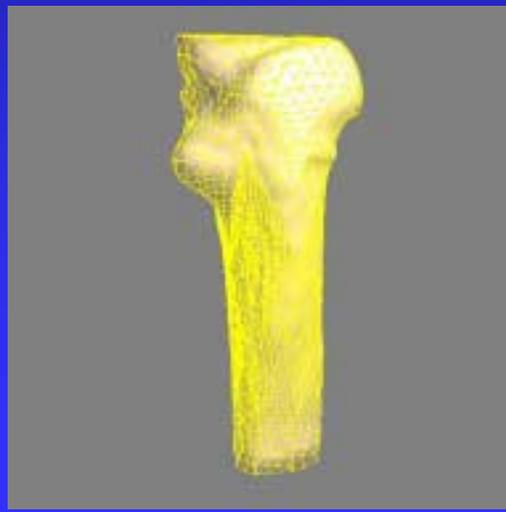
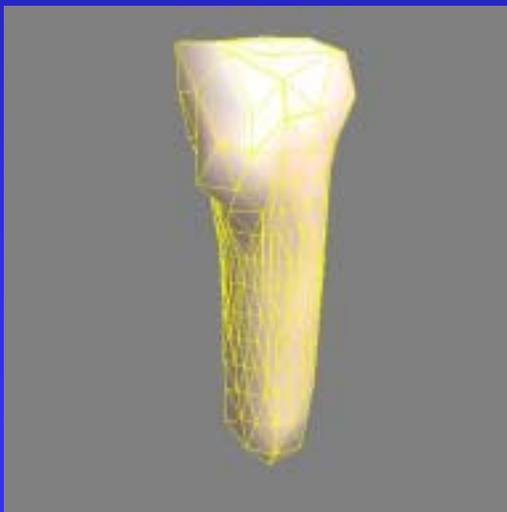
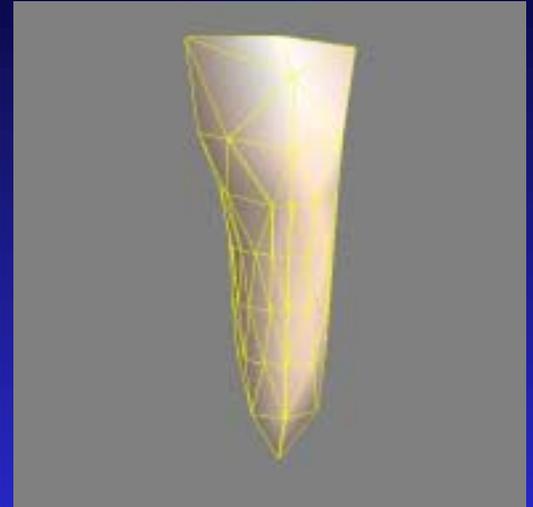
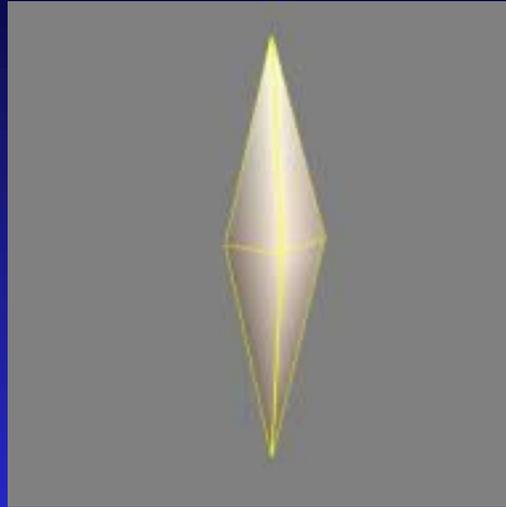
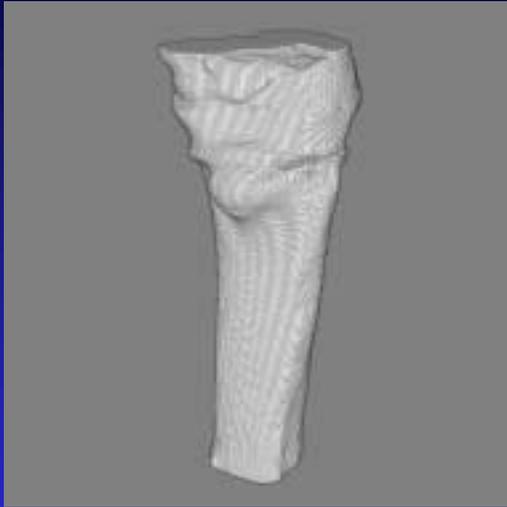
2. Liver:



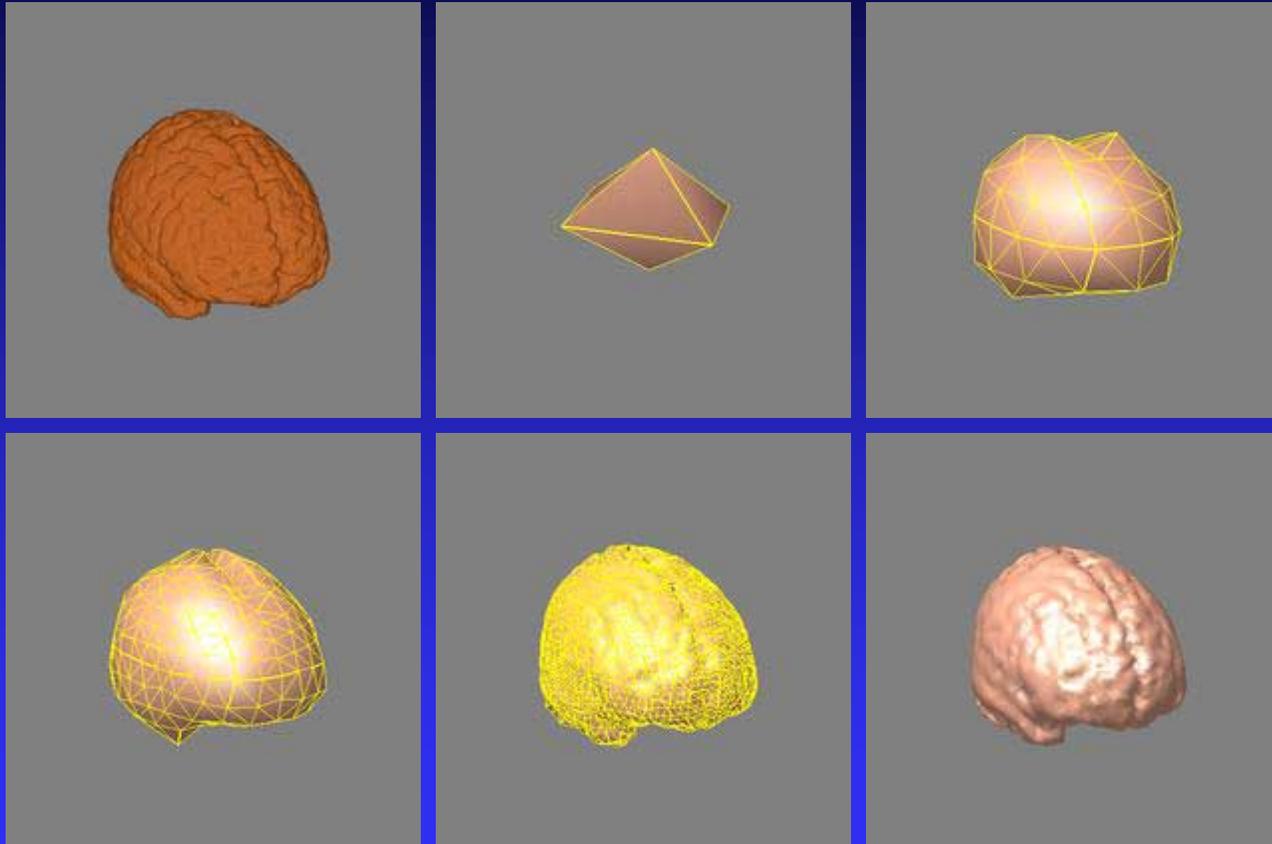
3. Left ventricular blood pool:



4. Femoral stem:



5. Brain:



Conclusions

- Recovering free-form shapes from irregularly spaced points requires parametrization of the points.
- In curve fitting, 1) find the spine of the points, and 2) using the projections of the points to the spine, compute parameters of the points.
- In surface fitting, 1) subdivide a sphere in parallel to the shape subdivision, and 2) from the correspondence between the shape and the sphere find parameters of the points.

Summary

- For shape design, we discussed
 - ◆ Bezier,
 - ◆ B-spline,
 - ◆ Rational Bezier,
 - ◆ NURBS,
 - ◆ D-NURBS, and
 - ◆ RaG.

Summary

- For shape recovery, we discussed
 - ◆ Thin-plate splines and
 - ◆ RaG curves and surfaces.
- Parametrizing irregularly spaced points for curve and surface fitting was also discussed.
- Programs for RaG curves and surfaces are included in your CD.
- Use of the programs will be demonstrated in the Computer Applications Lab (CAL).

**Thank you
for coming**

***Alyn Rockwood
Demetri Terzopoulos
Ardeshir Goshtasby***

Notes on Dynamic Non-Uniform Rational B-Splines (D-NURBS)

Demetri Terzopoulos

Courant Institute of Mathematical Sciences
New York University
New York, NY 10003

Abstract

Dynamic NURBS, or D-NURBS, are a physics-based generalization of non-uniform rational B-splines. NURBS have become a *de facto* standard in commercial modeling systems because of their power to represent both free-form and common analytic shapes. Traditionally, however, NURBS have been viewed as purely geometric primitives, which require the designer to interactively adjust many degrees of freedom (DOFs)—control points and associated weights—to achieve desired shapes. The conventional shape modification process can often be clumsy and laborious. D-NURBS are physics-based models that incorporate mass distributions, internal deformation energies, forces, and other physical quantities into the NURBS geometric substrate. Their dynamic behavior, resulting from the numerical integration of a set of nonlinear differential equations, produces physically meaningful, hence intuitive shape variation. Consequently, a modeler can interactively sculpt complex shapes to required specifications not only in the traditional indirect fashion, by adjusting control points, but also through direct physical manipulation, by applying simulated forces and local and global shape constraints. We use Lagrangian mechanics to formulate the equations of motion for D-NURBS curves, tensor-product surfaces, swung surfaces, and triangulated surfaces. We apply finite element analysis to reduce these equations to efficient algorithms that can be simulated at interactive rates using standard numerical techniques. We describe a prototype modeling environment based on D-NURBS, and demonstrate that D-NURBS can be effective tools in a wide range of CAGD applications such as shape blending, scattered data fitting, and interactive sculpting.

1 Introduction

In 1975 Versprille [33] proposed the Non-Uniform Rational B-Splines or NURBS. This shape representation for computer-aided geometric design (CAGD) generalized Riesenfeld's B-splines. NURBS quickly gained popularity and were incorporated into several commercial modeling systems [20]. The NURBS representation has several attractive properties. It offers a unified mathematical formulation for representing not only free-form curves and surfaces, but also standard analytic shapes such as conics, quadrics, and surfaces of revolution. The most frequently used NURBS design techniques are the specification of a control polygon, and interpolation or approximation of data points to generate the initial shape. For surfaces or solids, cross-sectional design including skinning, sweeping, and swinging operations is also popular. By adjusting the positions of control points, associated weights, and knots of the initial shape, one can design a large variety of shapes using NURBS [7, 18, 19, 20, 32].

1.1 Motivation of Physics-Based CAGD

NURBS have offered designers extraordinary flexibility for CAGD. However, traditional design methodology does not exploit the full potential of the underlying geometric formulations whose extraordinary flexi-

bility has some drawbacks:

- The designer is faced with the tedium of indirect shape manipulation through a bewildering variety of geometric parameters; i.e., by repositioning control points, adjusting weights, and modifying knot vectors. Despite the recent prevalence of sophisticated 3D interaction devices, the indirect geometric design process remains clumsy and time consuming in general.
- Shape design to required specifications by manual adjustment of available geometric degrees of freedom is often elusive, because relevant design tolerances are typically shape-oriented and not control point/weight oriented. The geometric “redundancy” of NURBS tends to make geometric shape refinement *ad hoc* and ambiguous; for instance, to adjust a shape should the designer move a control point, or change a weight, or move two control points,...?
- Typical design requirements may be stated in both quantitative and qualitative terms, such as “a fair and pleasing surface which approximates scattered data and interpolates a cross-section curve.” Such requirements impose both local and global constraints on shape. The incremental manipulation of local shape parameters to satisfy complex local and global shape constraints is at best cumbersome and often unproductive.

Physics-based modeling provides a means to overcome these drawbacks. Free-form deformable models, which were introduced to computer graphics in 1987 [30] and were further developed in recent years [29, 2, 3, 4, 34] are particularly relevant in the context of modeling with NURBS. Important advantages accrue from the deformable model approach [29]:

- The behavior of the deformable model is governed by physical laws. Through a computational physics simulation, the model responds dynamically to applied simulated forces in a natural and predictable way. Shapes can be sculpted interactively using a variety of force-based “tools.”
- The equilibrium state of the dynamic model is characterized by a minimum of the potential energy of the model subject to imposed constraints [28]. It is possible to formulate potential energy functionals that satisfy local and global design criteria, such as curve or surface (piecewise) smoothness, and to impose geometric constraints relevant to shape design.
- The physical model may be built upon a standard geometric foundation, such as free-form parametric curve and surface representations. This means that while shape design may proceed interactively or automatically at the physical level, existing geometric toolkits are concurrently applicable at the geometric level.

Thus, while traditional CAGD is based on geometric primitives and operations that often require the designer to painstakingly adjust geometric parameters in order to achieve desired shapes, physics-based CAGD treats these parameters as generalized coordinates which evolve automatically in response to simulated forces and geometric constraints according to the principles of Lagrangian mechanics. In this way, physics-based CAGD puts the laws of physics on the side of the designer. Physics-based CAGD can free designers from making nonintuitive decisions such as assigning weights to NURBS. In addition, with physics-based direct manipulation, non-expert users are able to concentrate on visual shape variation without necessarily comprehending the underlying mathematical formulation.

1.2 Dynamic NURBS: A Physics-Based Generalization of NURBS

Dynamic NURBS, or D-NURBS, are a dynamic generalization of conventional, geometric NURBS. They are physics-based models that incorporate mass distributions, internal deformation energies, and other physical quantities with the NURBS geometric substrate. Time is fundamental to the dynamic formulation. The

models are governed by dynamic differential equations which, when integrated numerically through time, continuously evolve the NURBS control points and weights in response to applied forces. The D-NURBS formulation supports interactive direct manipulation of NURBS objects, which results in physically meaningful hence intuitively predictable motion and shape variation.

Using D-NURBS, a modeler can interactively sculpt complex shapes not merely by kinematic adjustment of control points and weights, but dynamically as well—by applying simulated forces. Additional control over dynamic sculpting stems from the modification of physical parameters such as mass, damping, and elastic properties. Elastic functionals allow the imposition of qualitative “fairness” criteria through quantitative means. Linear or nonlinear constraints may be imposed either as hard constraints that must not be violated, or as soft constraints to be satisfied approximately. The latter may be interpreted intuitively as simple forces. Optimal shape design results when D-NURBS are allowed to achieve static equilibrium subject to shape constraints. All of these capabilities are subsumed under an elegant formulation grounded in physics.

D-NURBS are derived through the systematic use of Lagrangian mechanics and finite element analysis. D-NURBS control points and associated weights are generalized coordinates in the Lagrangian equations of motion. From a physics-based modeling point of view, the existence of weights makes the NURBS geometry substantially more challenging than B-spline geometry. Since the NURBS rational basis functions are functionally dependent on the weights, D-NURBS dynamics are generally nonlinear, and the mass, damping, and stiffness matrices must be recomputed at each simulation time step.¹ Fortunately, this does not preclude interactive performance on current graphics workstations, at least for the size of surface models that appear in our demonstrations. Because D-NURBS allow fully continuous mass and damping distributions, we obtain banded mass and damping matrices. To compute the integral expressions for the matrix entries in an efficient manner, we apply numerical quadrature to the underlying NURBS basis functions.

Detailed information about D-NURBS is available in the following publications: [31, 23, 25, 24].

2 Formulation of D-NURBS

The shape parameters of geometric NURBS play the role of generalized (physical) coordinates in dynamic NURBS. We introduce time, mass, and deformation energy into the standard NURBS formulation and employ Lagrangian dynamics to arrive at the system of nonlinear ordinary differential equations that govern the shape and motion of D-NURBS. In particular, we formulate four different varieties: D-NURBS curves, tensor-product D-NURBS surfaces, swung D-NURBS surfaces, and triangular D-NURBS surfaces.

2.1 D-NURBS Curves

NURBS generalize the non-rational parametric form. They inherit many of the properties of non-rational B-splines, such as the strong convex hull property, variation diminishing property, local support, and invariance under standard geometric transformations. Moreover, they have some additional properties. NURBS can be used to satisfy different smoothness requirements. They include weights as extra degrees of freedom which influence local shape. Most importantly, NURBS offer a common mathematical framework for implicit and parametric polynomial forms. In principle, they can represent analytic functions such as conics and quadrics precisely, as well as free-form shapes.

A kinematic NURBS curve extends the geometric NURBS definition by explicitly incorporating time. The kinematic curve is a function of both the parametric variable u and time t :

$$\mathbf{c}(u, t) = \frac{\sum_{i=0}^n \mathbf{p}_i(t) w_i(t) B_{i,k}(u)}{\sum_{i=0}^n w_i(t) B_{i,k}(u)}. \quad (1)$$

¹Note, however, that for static weights, the matrices become time invariant and the computational cost is reduced significantly.

where the $B_{i,k}(u)$ are the usual recursively defined piecewise basis functions [8], $\mathbf{p}_i(t)$ are the $n + 1$ control points, and $w_i(t)$ are associated non-negative weights. Assuming basis functions of degree $k - 1$, the curve has $n + k + 1$ knots t_i in non-decreasing sequence: $t_0 \leq t_1 \leq \dots \leq t_{n+k}$. In many applications, the end knots are repeated with multiplicity k in order to interpolate the initial and final control points \mathbf{p}_0 and \mathbf{p}_n .

To simplify notation, we define the vector of generalized coordinates $\mathbf{p}(t)$ and weights $w_i(t)$ as

$$\mathbf{p}(t) = \left[\mathbf{p}_0^\top \quad w_0 \quad \cdots \quad \mathbf{p}_n^\top \quad w_n \right]^\top,$$

where $^\top$ denotes transposition. We then express the curve (1) as $\mathbf{c}(u, \mathbf{p})$ in order to emphasize its dependence on \mathbf{p} whose components are functions of time.

The velocity of the kinematic spline is

$$\dot{\mathbf{c}}(u, \mathbf{p}) = \mathbf{J}\dot{\mathbf{p}}, \quad (2)$$

where the overstruck dot denotes a time derivative and $\mathbf{J}(u, \mathbf{p})$ is the Jacobian matrix. Because \mathbf{c} is a 3-component vector-valued function and \mathbf{p} is a $4(n + 1)$ dimensional vector, \mathbf{J} is the $3 \times 4(n + 1)$ matrix

$$\mathbf{J} = \left[\cdots \left[\begin{array}{ccc} \frac{\partial \mathbf{c}_x}{\partial \mathbf{p}_{i,x}} & 0 & 0 \\ 0 & \frac{\partial \mathbf{c}_y}{\partial \mathbf{p}_{i,y}} & 0 \\ 0 & 0 & \frac{\partial \mathbf{c}_z}{\partial \mathbf{p}_{i,z}} \end{array} \right] \frac{\partial \mathbf{c}}{\partial w_i} \quad \cdots \right], \quad (3)$$

where

$$\begin{aligned} \frac{\partial \mathbf{c}_x}{\partial \mathbf{p}_{i,x}} &= \frac{\partial \mathbf{c}_y}{\partial \mathbf{p}_{i,y}} = \frac{\partial \mathbf{c}_z}{\partial \mathbf{p}_{i,z}} = \frac{w_i B_{i,k}}{\sum_{j=0}^n w_j B_{j,k}}; \\ \frac{\partial \mathbf{c}}{\partial w_i} &= \frac{\sum_{j=0}^n (\mathbf{p}_i - \mathbf{p}_j) w_j B_{i,k} B_{j,k}}{(\sum_{j=0}^n w_j B_{j,k})^2}. \end{aligned}$$

The subscripts x , y , and z denote the components of a 3-vector. Furthermore, we can express the curve as the product of the Jacobian matrix and the generalized coordinate vector:

$$\mathbf{c}(u, \mathbf{p}) = \mathbf{J}\mathbf{p}. \quad (4)$$

The proof of (4) can be found elsewhere [31].

2.2 Tensor-Product D-NURBS Surfaces

In analogy to the kinematic curve of (1), a tensor-product D-NURBS surface

$$\mathbf{s}(u, v, t) = \frac{\sum_{i=0}^m \sum_{j=0}^n \mathbf{p}_{i,j}(t) w_{i,j}(t) B_{i,k}(u) B_{j,l}(v)}{\sum_{i=0}^m \sum_{j=0}^n w_{i,j}(t) B_{i,k}(u) B_{j,l}(v)} \quad (5)$$

generalizes the geometric NURBS surface. The $(m + 1)(n + 1)$ control points $\mathbf{p}_{i,j}(t)$ and weights $w_{i,j}(t)$, which are functions of time, comprise the D-NURBS generalized coordinates. Assuming basis functions along the two parametric axes of degree $k - 1$ and $l - 1$, respectively, the number of knots is $(m + k + 1)(n + l + 1)$. The non-decreasing knot sequence is $t_0 \leq t_1 \leq \dots \leq t_{m+k}$ along the u -axis and $s_0 \leq s_1 \leq \dots \leq s_{n+l}$ along the v -axis. The parametric domain is $t_{k-1} \leq u \leq t_{m+1}$ and $s_{l-1} \leq v \leq s_{n+1}$. If the end knots have multiplicity k and l in the u and v axis respectively, the surface patch will interpolate the four corners of the boundary control points.

We concatenate these $N = 4(m + 1)(n + 1)$ coordinates into the vector:

$$\mathbf{p}(t) = \left[\mathbf{p}_{0,0}^\top \quad w_{0,0} \quad \cdots \quad \mathbf{p}_{i,j}^\top \quad w_{i,j} \quad \cdots \quad \mathbf{p}_{m,n}^\top \quad w_{m,n} \right]^\top.$$

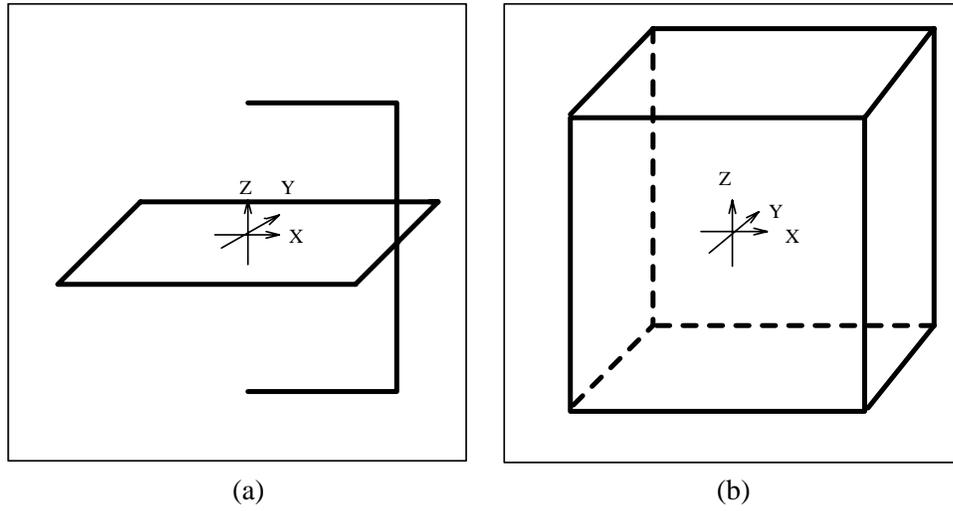


Figure 1: Construction of a cubical NURBS swung surface. (a) NURBS profile curve on x-z plane, NURBS trajectory curve on x-y plane. (b) Cube surface wireframe.

Two subscripts are now associated with the generalized coordinates, reflecting the surface parameters u and v . For concreteness, we order the components in these vectors such that the second subscript varies faster than the first, although this convention does not affect the derived results.

Similar to (2) and (4), we have

$$\dot{\mathbf{s}}(u, v, \mathbf{p}) = \mathbf{J}\dot{\mathbf{p}}, \quad \mathbf{s}(u, v, \mathbf{p}) = \mathbf{J}\mathbf{p}. \quad (6)$$

where $\mathbf{J}(u, v, \mathbf{p})$ is the $3 \times N$ Jacobian matrix of the D-NURBS surface with respect to \mathbf{p} . However, the contents of the Jacobian \mathbf{J} differ from those in the curve case. To arrive at an explicit expression for \mathbf{J} , let $\mathbf{B}_{i,j}(u, v, \mathbf{p})$, for $i = 0, \dots, m$, and $j = 0, \dots, n$, be a 3×3 diagonal matrix whose entries are

$$N_{i,j}(u, v, \mathbf{p}) = \frac{\partial \mathbf{s}}{\partial \mathbf{p}_{i,j}} = \frac{w_{i,j} B_{i,k}(u) B_{j,l}(v)}{\sum_{c=0}^m \sum_{d=0}^n w_{c,d} B_{c,k}(u) B_{d,l}(v)}$$

and let the 3-vector

$$\mathbf{w}_{i,j}(u, v, \mathbf{p}) = \frac{\partial \mathbf{s}}{\partial w_{i,j}} = \frac{\sum_{c=0}^m \sum_{d=0}^n (\mathbf{p}_{i,j} - \mathbf{p}_{c,d}) w_{c,d} B_{c,k}(u) B_{d,l}(v) B_{i,k}(u) B_{j,l}(v)}{(\sum_{c=0}^m \sum_{d=0}^n w_{c,d} B_{c,k}(u) B_{d,l}(v))^2}.$$

Hence,

$$\mathbf{J}(u, v, \mathbf{p}) = \begin{bmatrix} \mathbf{B}_{0,0} & \mathbf{w}_{0,0} & \cdots & \mathbf{B}_{m,n} & \mathbf{w}_{m,n} \end{bmatrix}.$$

Note that \mathbf{J} is now a $3 \times 4(m+1)(n+1)$ matrix.

2.3 Swung D-NURBS Surfaces

Many objects of interest, especially manufactured objects, exhibit symmetries. Often it is convenient to model symmetric objects through cross-sectional design by specifying profile curves [9]. Woodward [35] introduced the swinging operator by extending the spherical cross-product with a scaling factor, and applied it to generate surfaces with B-spline profile curves. Piegl [20] carried the swinging idea over to NURBS curves. He proposed NURBS swung surfaces, a special type of NURBS surfaces formed by swinging one planar NURBS profile curve along a second NURBS trajectory curve. For example, Fig. 1 illustrates the design of a cubical NURBS swung surface from two NURBS profile curves.

The NURBS swung surface retains a considerable breadth of geometric coverage. It can represent common geometric primitives such as spheres, tori, cubes, quadrics, surfaces of revolution, etc. The NURBS swung surface is efficient compared to a general NURBS surface, inasmuch as it can represent a broad class of shapes with essentially as few degrees of freedom as it takes to specify the two generator curves. Several geometric shape design systems include some form of swinging (or sweeping) among their repertoire of techniques [27].

Geometrically, a dynamic swung surface is generated from two planar kinematic NURBS profile curves through the swinging operation [20] (Fig 1). Let the two generator curves $\mathbf{c}_1(u, \mathbf{a})$ and $\mathbf{c}_2(v, \mathbf{b})$ be of the form (1). The swung surface is then defined as

$$\mathbf{s}(u, v, t) = \left[\alpha(t)\mathbf{c}_{1,x}\mathbf{c}_{2,x} \quad \alpha(t)\mathbf{c}_{1,x}\mathbf{c}_{2,y} \quad \mathbf{c}_{1,z} \right]^\top \quad (7)$$

where α is an arbitrary scalar. The second subscript denotes the component of a 3-vector.

Assume that \mathbf{c}_1 has basis functions of degree $k-1$ and that it has $m+1$ control points $\mathbf{a}_i(t)$ and weights $w_i^a(t)$. Similarly, \mathbf{c}_2 has basis functions of degree $l-1$ and that it has $n+1$ control points $\mathbf{b}_j(t)$ and weights $w_j^b(t)$. Therefore,

$$\mathbf{a}(t) = [\mathbf{a}_0^\top, w_0^a, \dots, \mathbf{a}_m^\top, w_m^a]^\top$$

and

$$\mathbf{b}(t) = [\mathbf{b}_0^\top, w_0^b, \dots, \mathbf{b}_n^\top, w_n^b]^\top$$

are the generalized coordinate vectors of the profile curves. We collect these into the generalized coordinate vector

$$\mathbf{p} = \left[\alpha \quad \mathbf{a}^\top \quad \mathbf{b}^\top \right]^\top.$$

This vector has dimensionality $M = 1 + 4(m+1) + 4(n+1)$. Thus the model has $O(n+m)$ degrees of freedom, compared to $O(nm)$ for general NURBS surfaces.

The velocity of the swung surface is

$$\dot{\mathbf{s}}(u, v, \mathbf{p}) = \mathbf{L}\dot{\mathbf{p}} \quad (8)$$

where $\mathbf{L}(u, v, \mathbf{p})$ is the Jacobian matrix with respect to the generalized coordinate vector \mathbf{p} . Hence, \mathbf{L} comprises the vectors $\partial\mathbf{s}/\partial\alpha$, $\partial\mathbf{s}/\partial\mathbf{a}$, and $\partial\mathbf{s}/\partial\mathbf{b}$. The expression of the $3 \times M$ matrix \mathbf{L} can be explicitly formulated [23]. Unlike \mathbf{J} in (4), \mathbf{L} cannot serve as the basis function matrix of the swung surface. Instead, we have

$$\mathbf{s}(u, v, \mathbf{p}) = \mathbf{H}\mathbf{p}, \quad (9)$$

where \mathbf{H} is the $3 \times M$ basis function matrix [23].

2.4 Triangular D-NURBS Surfaces

The main drawback of tensor-product NURBS is that the surface patches are rectangular. Consequently, the designer is forced to model multisided irregular shapes using degenerate patches with deteriorated inter-patch continuity. Thus, the associated smoothness constraints increase the complexity of the design task in general. In contrast, triangular B-splines [5] and NURBS can represent complex non-rectangular shapes over arbitrary triangulated domains with low degree piecewise polynomials that nonetheless maintain relatively high-order continuity. They can express smooth non-rectangular shapes without degeneracy. They can also model discontinuities by varying the knot distribution.

Let $T = \{\Delta(\mathbf{i}) = [\mathbf{r}, \mathbf{s}, \mathbf{t}] | \mathbf{i} = (i_0, i_1, i_2) \in Z_+^3\}$ be an arbitrary triangulation of the planar parametric domain, where i_0 , i_1 , and i_2 denote indices of \mathbf{r} , \mathbf{s} , and \mathbf{t} in the vertex array of the triangulation, respectively. For each vertex \mathbf{v} in the triangulated domain, we associate a knot sequence (also called a cloud of knots) $[\mathbf{v} = \mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_n]$ (which are inside the shaded circles in Fig. 2). Next, we define a convex hull

$$V_{i,\beta} = \{\mathbf{r}_0, \dots, \mathbf{r}_{\beta_0}, \mathbf{s}_0, \dots, \mathbf{s}_{\beta_1}, \mathbf{t}_0, \dots, \mathbf{t}_{\beta_2}\},$$

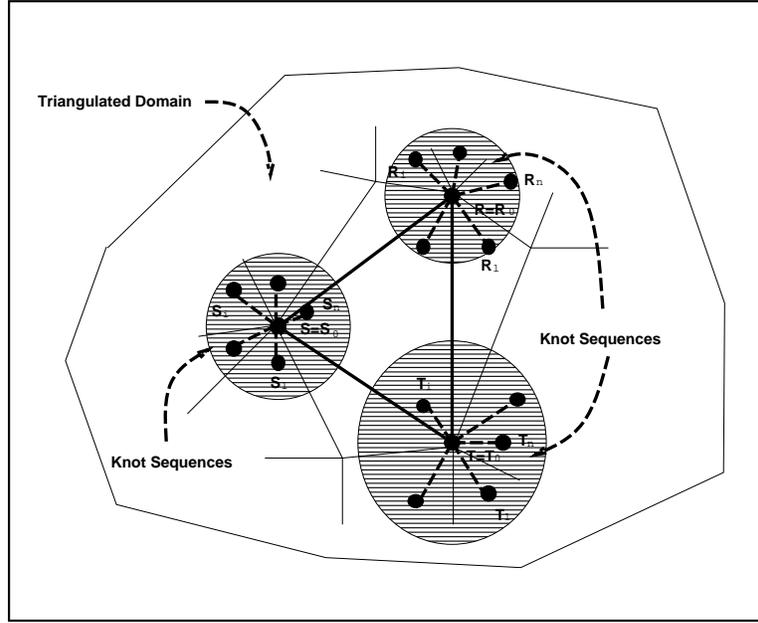


Figure 2: Knot vectors associated with each triangle in the domain triangulation.

where subscript i is a triangle index, and $\beta = (\beta_0, \beta_1, \beta_2)$ is a triplet such that $|\beta| = \beta_0 + \beta_1 + \beta_2 = n$. The bivariate simplex spline $M(\mathbf{u}|V_{i,\beta})$ with degree n over $V_{i,\beta}$ can be defined recursively (the details are found elsewhere [5]), where $\mathbf{u} = (u, v)$ defines the triangulated parametric domain of the surface. We then define a bivariate B-spline basis function as

$$N_{i,\beta}(\mathbf{u}) = d(\mathbf{r}_{\beta_0}, \mathbf{s}_{\beta_1}, \mathbf{t}_{\beta_2})M(\mathbf{u}|V_{i,\beta}), \quad (10)$$

where $d(\mathbf{r}_{\beta_0}, \mathbf{s}_{\beta_1}, \mathbf{t}_{\beta_2})$ is twice the area of $\Delta(\mathbf{r}_{\beta_0}, \mathbf{s}_{\beta_1}, \mathbf{t}_{\beta_2})$. Like the ordinary tensor-product D-NURBS, we define triangular D-NURBS as the combination of a set of piecewise rational functions by explicitly incorporating time and physical behavior. The surface is a function of both the parametric variable \mathbf{u} and time t :

$$\mathbf{s}(\mathbf{u}, t) = \frac{\sum_i \sum_{|\beta|=n} \mathbf{p}_{i,\beta}(t) w_{i,\beta}(t) N_{i,\beta}(\mathbf{u})}{\sum_i \sum_{|\beta|=n} w_{i,\beta}(t) N_{i,\beta}(\mathbf{u})}. \quad (11)$$

We define the vector of generalized coordinates (control points) $\mathbf{p}_{i,\beta}$ and (weights) $w_{i,\beta}$ as

$$\mathbf{p} = [\dots, \mathbf{p}_{i,\beta}^\top, w_{i,\beta}, \dots]^\top.$$

We then express (11) as $\mathbf{s}(\mathbf{u}, \mathbf{p})$ in order to emphasize its dependence on \mathbf{p} whose components are functions of time.

Thus, the velocity of the triangular D-NURBS is

$$\dot{\mathbf{s}}(\mathbf{u}, \mathbf{p}) = \mathbf{J}\dot{\mathbf{p}}, \quad (12)$$

where the overstruck dot denotes a time derivative and the Jacobian matrix $\mathbf{J}(\mathbf{u}, \mathbf{p})$ is the concatenation of the vectors $\partial \mathbf{s} / \partial \mathbf{p}_{i,\beta}$ and $\partial \mathbf{s} / \partial w_{i,\beta}$. Assuming m triangles in the parametric domain, β traverses $k = (n+2)! / (n!2!)$ possible triplets whose components sum to n . Because \mathbf{s} is a 3-vector and \mathbf{p} is an $M = 4mk$ dimensional vector, \mathbf{J} is a $3 \times M$ matrix, which may be written as

$$\mathbf{J} = \left[\dots, \begin{bmatrix} R_{i,\beta} & 0 & 0 \\ 0 & R_{i,\beta} & 0 \\ 0 & 0 & R_{i,\beta} \end{bmatrix}, \mathbf{w}_{i,\beta}, \dots \right] \quad (13)$$

where

$$R_{i,\beta}(\mathbf{u}, \mathbf{p}) = \frac{\partial \mathbf{s}_x}{\partial \mathbf{p}_{i,\beta,x}} = \frac{\partial \mathbf{s}_y}{\partial \mathbf{p}_{i,\beta,y}} = \frac{\partial \mathbf{s}_z}{\partial \mathbf{p}_{i,\beta,z}} = \frac{w_{i,\beta} N_{i,\beta}(\mathbf{u})}{\sum_j \sum_{|\alpha|=n} w_{j,\alpha} N_{j,\alpha}(\mathbf{u})}$$

and

$$\mathbf{w}_{i,\beta}(\mathbf{u}, \mathbf{p}) = \frac{\partial \mathbf{s}}{\partial w_{i,\beta}} = \frac{(\mathbf{p}_{i,\beta} - \mathbf{s}) N_{i,\beta}(\mathbf{u})}{\sum_j \sum_{|\alpha|=n} w_{j,\alpha} N_{j,\alpha}(\mathbf{u})}$$

The subscripts x , y , and z denote derivatives of the components of a 3-vector. Moreover, we can express the surface as the product of the Jacobian matrix and the generalized coordinate vector:

$$\mathbf{s}(\mathbf{u}, \mathbf{p}) = \mathbf{J}\mathbf{p}. \quad (14)$$

The proof of (14) is the same as that for the tensor-product D-NURBS [31].

2.5 D-NURBS Equations of Motion

The equations of motion of our D-NURBS are derived from the work-energy version of Lagrangian dynamics [11]. Applying the Lagrangian formulation to D-NURBS curves, tensor-product surfaces, swung surfaces, and triangulated surfaces, we obtain the second-order nonlinear equations of motion

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}_p + \mathbf{g}_p, \quad (15)$$

where the mass matrix $\mathbf{M}(\mathbf{p})$, the damping matrix $\mathbf{D}(\mathbf{p})$, and the stiffness matrix $\mathbf{K}(\mathbf{p})$ can all be formulated explicitly [31, 23, 25]. The $N \times N$ mass and damping matrices are

$$\mathbf{M}(\mathbf{p}) = \iint \mu \mathbf{J}^\top \mathbf{J} \, du \, dv; \quad \mathbf{D}(\mathbf{p}) = \iint \gamma \mathbf{J}^\top \mathbf{J} \, du \, dv \quad (16)$$

where $\mu(u, v)$ is the prescribed mass density function over the parametric domain of the surface and $\gamma(u, v)$ is the prescribed damping density function. To define an elastic potential energy for the surface, we adopt the *thin-plate under tension* energy model [28, 3, 34, 12, 31].² This yields the $N \times N$ stiffness matrix

$$\mathbf{K}(\mathbf{p}) = \iint \left(\alpha_{1,1} \mathbf{J}_u^\top \mathbf{J}_u + \alpha_{2,2} \mathbf{J}_v^\top \mathbf{J}_v + \beta_{1,1} \mathbf{J}_{uu}^\top \mathbf{J}_{uu} + \beta_{1,2} \mathbf{J}_{uv}^\top \mathbf{J}_{uv} + \beta_{2,2} \mathbf{J}_{vv}^\top \mathbf{J}_{vv} \right) \, du \, dv, \quad (17)$$

where the subscripts on \mathbf{J} denote parametric partial derivatives. The $\alpha_{i,j}(u, v)$ and $\beta_{i,j}(u, v)$ are elasticity functions which control tension and rigidity, respectively, in the two parametric coordinate directions. Other energies are applicable, including the nonquadratic, curvature-based energies [30, 17]. The generalized force $\mathbf{f}_p(\mathbf{p}) = \iint \mathbf{J}^\top \mathbf{f}(u, v, t) \, du \, dv$ is obtained through the principle of virtual work [11] done by the applied force distribution $\mathbf{f}(u, v, t)$. Because of the geometric nonlinearity, generalized inertial forces $\mathbf{g}(\mathbf{p})$ are also associated with the models (see our journal articles for the details [31, 23]).

3 Finite Element Implementation

The evolution of \mathbf{p} , determined by (15) with time-varying matrices, cannot be solved analytically in general. Instead, we pursue an efficient numerical implementation using finite-element techniques [13].

Standard finite element codes explicitly assemble the global matrices that appear in the discrete equations of motion [13]. We use an iterative matrix solver to avoid the cost of assembling the global \mathbf{M} , \mathbf{D} , and \mathbf{K} . In this way, we work with the individual element matrices and construct finite element data structures that permit the parallel computation of element matrices.

²See the second part of the course notes by the author on Generalized Splines.

3.1 Element Data Structures

We define an element data structure which contains the geometric specification of the surface patch element along with its physical properties. A complete D-NURBS surface is then implemented as a data structure which consists of an ordered array of elements with additional information. The element structure includes pointers to appropriate components of the global vector \mathbf{p} (control points and weights). Neighboring elements will share some generalized coordinates. The shared variables will have multiple pointers impinging on them. We also allocate in each element an elemental mass, damping, and stiffness matrix, and include in the element data structure the quantities needed to compute these matrices. These quantities include the mass $\mu(u, v)$, damping $\gamma(u, v)$, and elasticity $\alpha_{i,j}(u, v)$, $\beta_{i,j}(u, v)$ density functions, which may be represented as analytic functions or as parametric arrays of sample values.

3.2 Calculation of Element Matrices

The integral expressions for the mass, damping, and stiffness matrices associated with each element are evaluated numerically using Gaussian quadrature [22]. We shall explain the computation of the element mass matrix; the computation of the damping and stiffness matrices follow suit. Assuming the parametric domain of the element is Ω , the expression for entry m_{ij} of the mass matrix takes the integral form

$$m_{ij} = \int_{\Omega} \mu(u, v) f_{ij}(u, v) du dv,$$

where f_{ij} are entries of the mass matrix. Given integers N_g , we can find Gauss weights a_g , and abscissas u_g, v_g in the two parametric directions of Ω such that m_{ij} can be approximated by [22]

$$m_{ij} \approx \sum_{g=1}^{N_g} a_g \mu(u_g, v_g) f_{ij}(u_g, v_g).$$

We apply the de Boor algorithm [6] or the recursive algorithm of multivariate simplex B-splines [15] to evaluate $f_{ij}(u_g, v_g)$. In general, Gaussian quadrature evaluates the integral exactly with N weights and abscissas for polynomials of degree $2N - 1$ or less. In our system we choose N_g to be integers between 4 and 7. Our experiments indicate that matrices computed in this way lead to stable, convergent solutions. Note that because of the irregular knot distribution for the case of triangular D-NURBS, many f_{ij} 's are zero over the triangular subdomains of Ω . We can further subdivide the subdomains in order to decrease the numerical quadrature error [25].

3.3 Discrete Dynamics Equations

To integrate (15) in an interactive modeling environment, it is important to provide the modeler with visual feedback about the evolving state of the dynamic model. Rather than using costly time integration methods that take the largest possible time steps, it is more crucial to provide a smooth animation by maintaining the continuity of the dynamics from one step to the next. Hence, less costly yet stable time integration methods that take modest time steps are desirable.

The state of the dynamic NURBS at time $t + \Delta t$ is integrated using prior states at time t and $t - \Delta t$. To maintain the stability of the integration scheme, we use an implicit time integration method, which employs the time integration formula

$$\left(2\mathbf{M} + \Delta t\mathbf{D} + 2\Delta t^2\mathbf{K}\right) \mathbf{p}^{(t+\Delta t)} = 2\Delta t^2(\mathbf{f}_p + \mathbf{g}_p) + 4\mathbf{M}\mathbf{p}^{(t)} - (2\mathbf{M} - \Delta t\mathbf{D})\mathbf{p}^{(t-\Delta t)} \quad (18)$$

where the superscripts denote evaluation of the quantities at the indicated times. The matrices and forces are evaluated at time t .

We employ the conjugate gradient method to obtain an iterative solution [22]. To achieve interactive simulation rates, we limit the number of conjugate gradient iterations per time step to 10. We have observed that 2 iterations typically suffice to converge to a residual of less than 10^{-3} . More than 2 iterations tend to be necessary when the physical parameters (mass, damping, tension, stiffness, applied forces) are changed significantly during dynamic simulation. Hence, our implementation permits the real-time simulation of dynamic NURBS surfaces on common graphics workstations.

The equations of motion allow realistic dynamics such as would be desirable for physics-based computer graphics animation. It is possible, however, to make simplifications that further reduce the computational cost of (18) to interactively sculpt larger surfaces. For example, in CAGD applications such as data fitting where the modeler is interested only in the final equilibrium configuration of the model, it makes sense to simplify (15) by setting the mass density function $\mu(u, v)$ to zero, so that the inertial terms vanish.

4 Physics-Based Shape Design

In the physics-based shape design approach, design requirements may be satisfied through the use of energies, forces, and constraints. The designer may apply time-varying forces to sculpt shapes interactively or to optimally approximate data. Certain aesthetic constraints such as “fairness” are expressible in terms of elastic energies that give rise to specific stiffness matrices \mathbf{K} . Other constraints include position or normal specification at surface points, and continuity requirements between adjacent surface patches. By building D-NURBS upon the standard NURBS geometry, we allow the modeler to continue to use the whole spectrum of advanced geometric design tools that have become prevalent, among them, the imposition of geometric constraints that the final shape must satisfy.

4.1 Applied Forces

Sculpting tools may be implemented as applied forces. The force $\mathbf{f}(u, v, t)$ represents the net effect of all applied forces. Typical force functions are spring forces, repulsion forces, gravitational forces, inflation forces, etc. [30].

For example, consider connecting a material point (u_0, v_0) of a D-NURBS surface to a point \mathbf{d}_0 in space with an ideal Hookean spring of stiffness k . The net applied spring force is

$$\mathbf{f}(u, v, t) = \iint k(\mathbf{d}_0 - \mathbf{s}(u, v, t))\delta(u - u_0, v - v_0) du dv, \quad (19)$$

where the δ is the unit delta function. Equation (19) implies that $\mathbf{f}(u_0, v_0, t) = k(\mathbf{d}_0 - \mathbf{s}(u_0, v_0, t))$ and vanishes elsewhere on the surface, but we can generalize it by replacing the δ function with a smooth kernel (e.g., a unit Gaussian) to spread the applied force over a greater portion of the surface. Furthermore, the points (u_0, v_0) and \mathbf{d}_0 need not be constant, in general. We can control either or both using a mouse to obtain an interactive spring force.

4.2 Constraints

In practical applications, design requirements may be posed as a set of physical parameters or as geometric constraints. Nonlinear constraints can be enforced through Lagrange multiplier techniques [16]. This approach increases the number of degrees of freedom, hence the computational cost, by adding unknowns λ , known as Lagrange multipliers, which determine the magnitudes of the constraint forces. The augmented Lagrangian method [16] combines the Lagrange multipliers with the simpler penalty method [21]. The Baumgarte stabilization method [1] solves constrained equations of motion through linear feedback control [14, 31]. These techniques are appropriate for D-NURBS with nonlinear constraints.

Linear geometric constraints such as point, curve, and surface normal constraints can be easily incorporated into dynamic swung surface by reducing the matrices and vectors in (15) to a minimal unconstrained set of generalized coordinates. They can then be implemented by applying the same numerical solver on an unconstrained subset of \mathbf{p} [31].

D-NURBS have an interesting idiosyncrasy due to the weights. While the control point components of \mathbf{p} may take arbitrary finite values in \mathfrak{R} , negative weights may cause the denominator to vanish at some evaluation points, causing the matrices to diverge. Although not forbidden, negative weights are not useful. We enforce positivity of weights at each simulation time step by simply projecting any weight value that has drifted below a small positive threshold back to this lower bound. Alternatively, we can give the designer the option of constraining the weights near certain desired target values w_i^0 by including in the surface energy the penalty term $c \sum (w_i - w_i^0)$, where c controls the tightness of the constraint.

5 Modeling Applications

This section describes our D-NURBS modeling environment and presents several applications relating to solid rounding, optimal surface fitting, and interactive sculpting.

5.1 Interactive Modeling Environment

We have developed a prototype modeling environment based on the tensor-product and swung D-NURBS model. The system is written in C and it currently runs under Iris Explorer on Silicon Graphics workstations. It may be combined with existing Explorer modules for data input and surface visualization. Our parallelized iterative numerical algorithm takes advantage of an SGI Iris 4D/380VGX multiprocessor. To date, our D-NURBS modules implement 3D curve and surface objects with basis function orders of 2, 3, or 4 (i.e., from linear to cubic D-NURBS) with linear geometric constraints.

We have also developed prototype modeling software based on dynamic triangular B-splines which is a special case of triangular D-NURBS by fixing all weights to be unity (an advanced system based upon dynamic triangular NURBS is under construction). We have adopted the data structure, file, and rendering formats of existing geometric triangular B-spline software [10]. To implement the Lagrangian dynamics model on top of this software, we have had to implement a new algorithm for simultaneously evaluating non-zero basis functions and their derivatives up to second order at arbitrary domain points for finite element assembly and dynamic simulation.

Users can sculpt surface shapes in conventional geometric ways, such as by sketching control polygons of arbitrary profile curves, repositioning control points, and adjusting associated weights, or according to the physics-based paradigm through the use of forces. They can satisfy design requirements by adjusting the internal physical parameters such as the mass, damping, and stiffness densities, along with force gain factors. Linear constraints such as the freezing of control points have been associated with physics-based toolkits in our prototype system. Local geometric constraints can be used to achieve real-time local manipulation for interactive sculpting of complex objects.

5.2 Optimal Surface Fitting

D-NURBS are applicable to the optimal fitting of regular or scattered data [26]. The most general and often most useful case occurs with scattered data, when there are fewer or more data points than unknowns—i.e., when the solution is underdetermined or overdetermined by the data. In this case, D-NURBS can yield “optimal” solutions by minimizing the thin-plate under tension deformation energy [28]. The surfaces are optimal in the sense that they provide the smoothest curve or surface (as measured by the deformation energy) which interpolates or approximates the data.

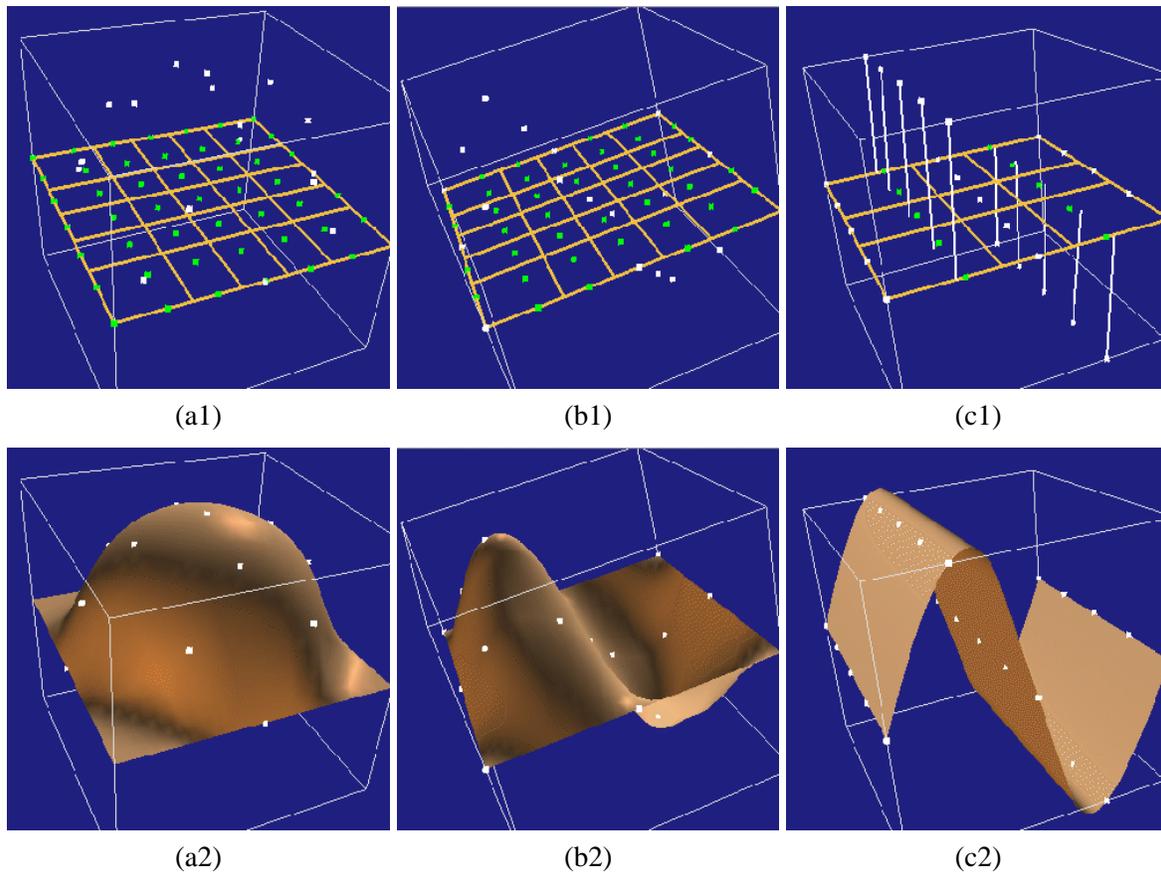


Figure 3: Optimal surface fitting: D-NURBS surfaces fit to sampled data from (a) a hemisphere, (b) a convex/concave surface, (c) a sinusoidal surface. (a–c1) D-NURBS patch outline with control points (white) and data points (red) shown. (a–c2) D-NURBS surface at equilibrium fitted to scattered data points. Red line segments in (c2) represent springs with fixed attachment points on surface.

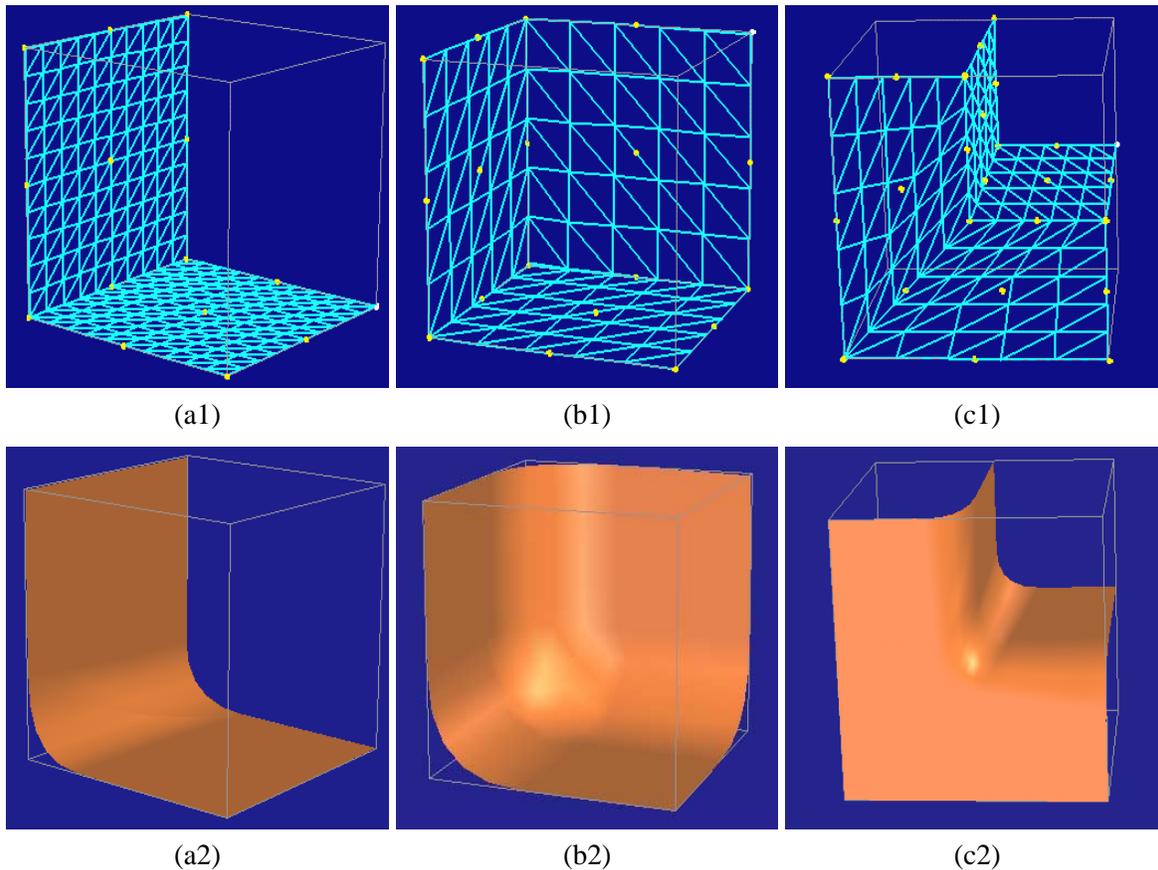


Figure 4: Solid rounding with triangular D-NURBS: Rounding of (a) an edge, (b) a trihedral corner, (c) a bevel joint. (a1-c1) Initial wireframe surfaces. (a2-c2) Final rounded, shaded surfaces.

The data point interpolation problem amounts to a linear constraint problem when the weights are fixed, and it is amenable to the constraint techniques presented in Section 4.2. The optimal approximation problem can be approached in physical terms, by coupling the D-NURBS to the data through Hookean spring forces (19). We interpret \mathbf{d}_0 in (19) as the data point (generally in \mathbb{R}^3) and (u_0, v_0) as the D-NURBS parametric coordinates associated with the data point (which may be the nearest material point to the data point). The spring constant c determines the closeness of fit to the data point.

We present three examples of surface fitting using tensor-product D-NURBS coupled to data points through spring forces. Fig. 3(a) shows 19 data points sampled from a hemisphere and their interpolation with a quadratic D-NURBS surface with 49 control points. Fig. 3(b) shows 19 data points and the reconstruction of the implied convex/concave surface by a quadratic D-NURBS with 49 control points. The spring forces associated with the data points are applied to the nearest points on the surface. In Fig. 3(c) we reconstruct a wave shape from 25 sample points using springs with fixed attachments to a quadratic tensor-product D-NURBS surface with 25 control points.

5.3 Rounding

The rounding operation is usually attempted geometrically by enforcing continuity requirements on the fillet which interpolates between two or more surfaces. By contrast, the D-NURBS can produce a smooth fillet by minimizing its internal deformation energy subject to position and normal constraints. The dynamic simulation automatically produces the desired final shape as it achieves static equilibrium.

Fig. 4(a) demonstrates the rounding of a sharp edge represented by a quadratic triangular D-NURBS

surface with 36 control points. The sharp edge can be represented exactly with multiple control points. By restricting the control polygon to be a continuous net, we reduced the number of control points to 21. The initial wireframe surface is shown in Fig. 4(a1). After initiating the physical simulation, the sharp edges are rounded as the final shape equilibrates into the minimal energy state shown by the shaded surface in Fig. 4(a2).

Fig. 4(b) illustrates the rounding of a trihedral corner of a cube. The corner is represented using a quadratic triangular D-NURBS with 78 control points. The initial wireframe is shown in Fig. 4(b1). The rounding operation is applied in the vicinity of three sharp edges. The sharp edges and corner are rounded with position and normal constraints along the far boundaries of the faces of the shaded surface shown in Fig. 4(b2).

Fig. 4(c) shows a rounding example involving a bevel joint. The bevel joint is a quadratic triangular D-NURBS with 108 control points. The initial right-angle joint and the final rounded surface are shown in Fig. 4(c1–2).

5.4 Interactive Sculpting

In the physics-based modeling approach, not only can designers manipulate the individual degrees of freedom with conventional geometric methods, but they can also move the object or refine its shape with interactive sculpting forces.

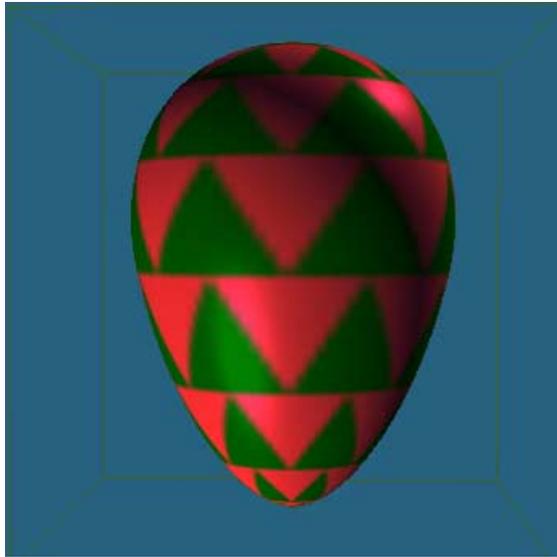
The physics-based modeling approach is ideal for interactive sculpting of surfaces. It provides direct manipulation of the dynamic surface to refine the shape of the surface through the application of interactive sculpting tools in the form of forces. Fig. 5(a) illustrates the results of four interactive sculpting sessions using swung D-NURBS surfaces and simple spring forces. A sphere was generated using two quadratic curves with 4 and 7 control points and was sculpted into the ovoid shown in Fig. 5(a). A torus whose two profile curves are quadratic with 7 and 7 control points, respectively, has been deformed into the shape in Fig. 5(b). A hat shape was created from two curves with 9 and 6 control points and was then deformed by spring forces into the shape in Fig 5(d). Finally, we generated a wine glass shape using two curves with 7 and 5 control points and sculpted it into the more pleasing shape shown in Fig 5(c).

6 Conclusion

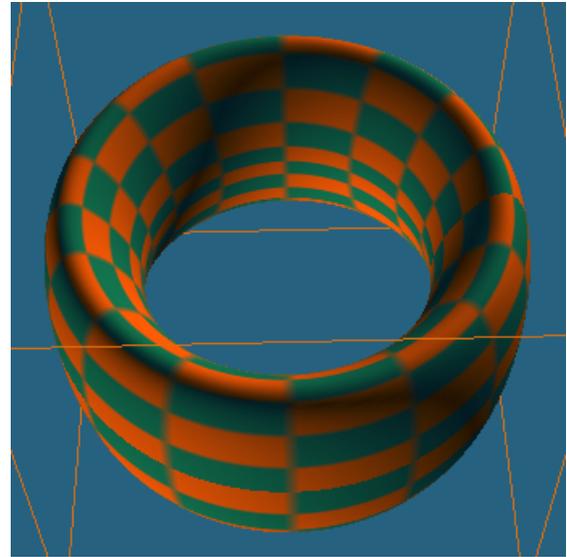
We have described D-NURBS, a dynamic generalization of geometric NURBS. D-NURBS were derived systematically through the application of Lagrangian mechanics and implemented using concepts from finite element analysis and efficient numerical methods. The mathematical development comprised four varieties: D-NURBS curves, tensor-product D-NURBS surfaces, swung D-NURBS surfaces, and triangular D-NURBS surfaces.

We also presented a new physics-based design paradigm based on D-NURBS which generalizes well established geometric design. This paradigm was the basis of a D-NURBS interactive modeling environment. The physics-based framework furnishes designers not only the standard geometric toolkits but powerful force-based sculpting tools as well. It provides mechanisms for automatically adjusting unknown parameters to support user manipulation and satisfy design requirements.

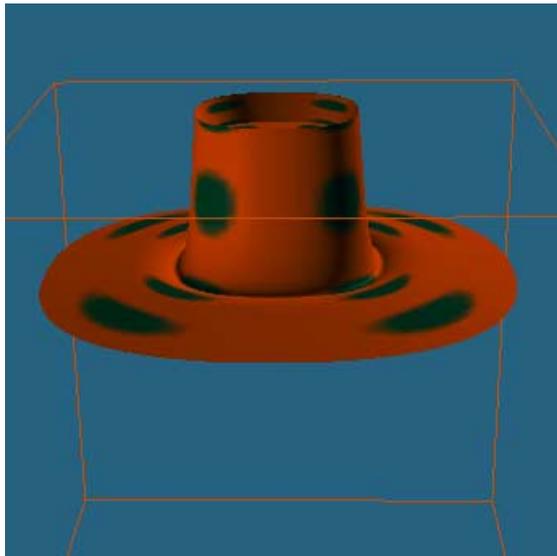
Since D-NURBS are built on the industry-standard NURBS geometric substrate, designers working with them can continue to make use of the existing array of geometric design toolkits. With the advent of high-performance graphics systems, however, the physics-based framework is poised for incorporation into commercial design systems to interactively model and sculpt complex shapes in real-time. Thus, D-NURBS can unify the features of the industry-standard geometry with the many demonstrated conveniences of interaction through physical dynamics.



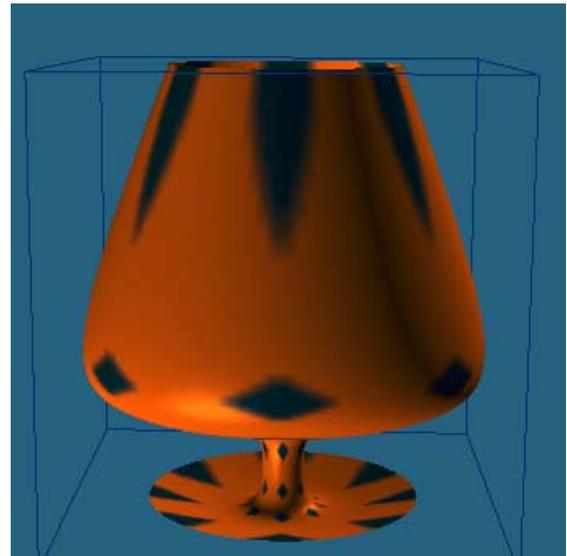
(a)



(b)



(c)



(d)

Figure 5: Interactive Sculpting of D-NURBS Swung Surfaces. Open and closed surfaces shown were sculpted interactively from prototype shapes noted in parentheses (a) Egg shape (sphere). (b) Deformed toroid (torus). (c) Hat (open surface). (d) Wine glass (cylinder).

References

- [1] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Comp. Meth. in Appl. Mech. and Eng.*, 1:1–16, 1972.
- [2] M.I.G. Bloor and M.J. Wilson. Representing PDE surfaces in terms of B-splines. *Computer-Aided Design*, 22(6):324–331, 1990.
- [3] G. Celniker and D. Gossard. Deformable curve and surface finite elements for free-form shape design. *Computer Graphics*, 25(4):257–266, 1991. (Proc. ACM Siggraph'91).
- [4] G. Celniker and W. Welch. Linear constraints for deformable B-spline surfaces. In *Proceedings, Symposium on Interactive 3D Graphics*, pages 165–170, 1992.
- [5] W. Dahmen, C. Micchelli, and H.-P. Seidel. Blossoming begets B-spline bases built better by B-patches. *Mathematics of Computation*, 59(199):97–115, 1992.
- [6] C. de Boor. On calculating with B-Splines. *Journal of Approximation Theory*, 6(1):50–62, 1972.
- [7] G. Farin. Trends in curve and surface design. *Computer-Aided Design*, 21(5):293–296, 1989.
- [8] G. Farin. *Curves and Surfaces for Computer aided Geometric Design: A Practical Guide*. Academic Press, second edition, 1990.
- [9] I.D. Faux and M.J. Pratt. *Computational Geometry for Design and Manufacture*. Ellis Horwood, Chichester,UK, 1979.
- [10] P. Fong and H.-P. Seidel. An implementation of triangular B-spline surfaces over arbitrary triangulations. *Computer Aided Geometric Design*, 3-4(10):267–275, 1993.
- [11] B.R. Gossick. *Hamilton's Principle and Physical Systems*. Academic Press, New York and London, 1967.
- [12] M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using Catmull-Clark surfaces. In *Computer Graphics Proceedings, Annual Conference Series, Proc. ACM Siggraph'93 (Anaheim, CA, Aug., 1993)*, pages 35–44, 1993.
- [13] H. Kardestuncer. *Finite Element Handbook*. McGraw-Hill, New York, 1987.
- [14] D. Metaxas and D. Terzopoulos. Dynamic deformation of solid primitives with constraints. *Computer Graphics*, 26(2):309–312, 1992. (Proc. ACM Siggraph'92).
- [15] C.A. Micchelli. On a numerically efficient method for computing with multivariate B-splines. In W. Schempp and K. Zeller, editors, *Multivariate Approximation Theory*, pages 211–248. Birkhauser, Basel, 1979.
- [16] M. Minoux. *Mathematical Programming*. Wiley, New York, 1986.
- [17] H.P. Moreton and C.H. Sequin. Functional optimization for fair surface design. *Computer Graphics*, 26(2):167–176, 1992. (Proc. ACM Siggraph'92).
- [18] L. Piegl. Modifying the shape of rational B-splines. part 1:curves. *Computer-Aided Design*, 21(8):509–518, 1989.
- [19] L. Piegl. Modifying the shape of rational B-splines. part 2:surfaces. *Computer-Aided Design*, 21(9):538–546, 1989.
- [20] L. Piegl. On NURBS: A survey. *IEEE Computer Graphics and Applications*, 11(1):55–71, Jan. 1991.
- [21] J. Platt. A generalization of dynamic constraints. *CVGIP: Graphical Models and Image Processing*, 54(6):516–525, 1992.
- [22] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, 1986.
- [23] H. Qin and D. Terzopoulos. Dynamic NURBS swung surfaces for physics-based shape design. *Computer Aided Design*, 27(2):111–127, 1995.
- [24] H. Qin and D. Terzopoulos. D-NURBS: A physics-based framework for geometric design. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):85–96, 1996.

- [25] H. Qin and D. Terzopoulos. Triangular NURBS and their dynamic generalizations. *Computer Aided Geometric Design*, 14:325–347, 1997.
- [26] L.L. Schumaker. Fitting surfaces to scattered data. In G.G. Lorentz, C.K. Chui, and L.L. Schumaker, editors, *Approximation Theory II*, pages 203–267. Academic Press, New York, 1976.
- [27] J. Snyder and J. Kajiya. Generative modeling: A symbolic system for geometric modeling. *Computer Graphics*, 26(2):369–378, 1992.
- [28] D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(4):413–424, 1986.
- [29] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, 1988.
- [30] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics*, 21(4):205–214, 1987.
- [31] D. Terzopoulos and H. Qin. Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2):103–136, 1994.
- [32] W. Tiller. Rational B-splines for curve and surface representation. *IEEE Computer Graphics and Applications*, 3(6):61–69, Sept. 1983.
- [33] K.J. Versprille. *Computer-Aided Design Applications of the Rational B-Spline Approximation form*. PhD thesis, Syracuse University, 1975.
- [34] W. Welch and A. Witkin. Variational surface modeling. *Computer Graphics*, 26(2):157–166, 1992. (Proc. ACM Siggraph'92).
- [35] C. Woodward. Cross-sectional design of B-spline surfaces. *Computers and Graphics*, 11(2):193–201, 1987.

Notes on Generalized Splines

Demetri Terzopoulos

Courant Institute of Mathematical Sciences
New York University
New York, NY 10003

Abstract

Generalized splines have proven useful in the reconstruction of surfaces from scattered data. The surface reconstruction problem is of concern in diverse fields, such as computer-aided design, computer vision, digital terrain mapping, geophysics, meteorology, etc. [12, 6]. These notes cover some of the mathematics of generalized splines, with emphasis on a relatively recent extension—discontinuity-preserving generalized splines [15].

1 Classical Smoothing Splines

Consider a set of N data points $\{(x_i, c_i)\}_{i=1}^N$. Each data point i comprises a position x_i on the infinite domain $x \in \mathfrak{R}$ and a value c_i . The approximation problem associated with classical *smoothing splines* [11, 10] involves the minimization of the functional

$$\mathcal{E}_m(v) = \mathcal{S}_m(v) + \mathcal{P}(v) = \frac{1}{2} \int_{\mathfrak{R}} \frac{\partial^m v(x)}{\partial x^m} dx + \frac{1}{2} \sum_{i=1}^N \alpha_i (v(x_i) - c_i)^2, \quad (1)$$

for non-negative weights α_i , where the order m of the partial derivative in the smoothness functional $\mathcal{S}_m(v)$ determines the continuity that any admissible functions $v(x)$ must possess.¹ The function $u(x)$ which minimizes \mathcal{E}_m is a piecewise polynomial of degree $2m - 1$.

2 Generalized Splines

For data $\{(\mathbf{x}_i, c_i)\}_{i=1}^N$ in a p -dimensional domain $\mathbf{x} = (x_1, \dots, x_p) \in \mathfrak{R}^p$, the natural generalization of the smoothness functional in (1) is

$$\begin{aligned} \mathcal{S}_m(v) &= \frac{1}{2} \sum_{i_1, \dots, i_m=1}^p \int_{\mathfrak{R}^p} \left(\frac{\partial^m v(\mathbf{x})}{\partial x_{i_1} \dots \partial x_{i_m}} \right)^2 d\mathbf{x} \\ &= \frac{1}{2} \int_{\mathfrak{R}^p} \sum_{|j|=m} \frac{m!}{j_1! \dots j_p!} \left(\frac{\partial^m v(\mathbf{x})}{\partial x_1^{j_1} \dots \partial x_p^{j_p}} \right)^2 d\mathbf{x}. \end{aligned} \quad (2)$$

Here, $j = (j_1, \dots, j_p)$ is a multi-index with $|j| = j_1 + \dots + j_p$. These (scalar) *generalized spline* functionals, which were studied by Duchon [4, 5] and Meinguet [8], have several important properties²

¹In the context of regularization theory [17] \mathcal{E}_m is known as a regularization functional and \mathcal{S}_m is known as a stabilization functional.

²The functionals define the natural semi-norms of a certain class of Sobolev spaces \mathcal{H} . These semi-norms are invariant under translation, rotation, and similarity transformation. The null-spaces \mathcal{N} of the semi-norms are simply the $M = \binom{p+m-1}{p}$ dimen-

3 Surface Splines

In the case $p = 2$, where we define $\mathbf{x} = (x_1, x_2) = (x, y)$ for notational convenience, the generalized spline functionals can be written as

$$\mathcal{S}_m(v) = \frac{1}{2} \iint_{\mathbb{R}^2} \sum_{i=0}^m \binom{m}{i} \left(\frac{\partial^m v}{\partial x^i \partial y^{m-i}} \right)^2 dx dy. \quad (3)$$

These functionals pertain to the problem of fitting surfaces to scattered data. This mathematical problem is of considerable concern in numerous application areas, notably to visible-surface reconstruction in computer vision [16].

Surface splines have interesting physical interpretations involving equilibria of elastic bodies with C^{m-1} intrinsic continuity [15]. The two lowest order cases are of particular interest. For $m = 1$ the functional reduces to

$$\mathcal{S}_1(v) = \frac{1}{2} \iint_{\mathbb{R}^2} (v_x^2 + v_y^2) dx dy, \quad (4)$$

where the subscripts denote a partial derivative with respect to x or y , is proportional to the small deflection strain energy of an area-minimizing membrane (e.g., rubber sheet) [3].³ Physically, the membrane spline characterizes a surface of C^0 continuity, a continuous surface which, however, need not have continuous first (and higher) order partial derivatives. For $m = 2$,

$$\mathcal{S}_2(v) = \frac{1}{2} \iint_{\mathbb{R}^2} (v_{xx}^2 + 2v_{xy}^2 + v_{yy}^2) dx dy. \quad (5)$$

This is known as the *thin plate spline* functional because it is proportional to the small deflection bending energy of a thin, flexible plate (with zero Poisson ratio) [3]. The thin plate spline is a C^1 surface, a continuous surface with continuous first partial derivatives, which need not have continuous derivatives of degree greater than one. As the natural extension to the common one-dimensional cubic spline, the thin plate spline is a popular surface interpolant [12, 6].

4 Thin-Plate Splines Under Tension

Schweikert [13] introduced splines under tension which can be made to imitate the behavior of cubic interpolating splines, while suppressing the extraneous inflection points that sometimes afflict cubic splines (see, also, [2]). They may be characterized as interpolatory functions $u(x)$ that minimize the functional $\int_{\mathbb{R}} (v_{xx}^2 + \tau v_x^2) dx$, where τ is a prespecified positive constant, called the tension parameter. The first term influences the “length” of the spline while the second term influences its “curvature.” Increasing the tension tends to eliminate extraneous loops and ripples by reducing the length of the spline.

The natural generalization of splines under tension to surfaces is a weighted convex combination of the thin-plate (5) and membrane (4) splines as follows [15]:

$$\mathcal{S}_{\rho\tau}(v) = \frac{\rho}{2} \iint_{\mathbb{R}^2} \left((1 - \tau)(v_{xx}^2 + 2v_{xy}^2 + v_{yy}^2) + \tau(v_x^2 + v_y^2) \right) dx dy, \quad (6)$$

where $0 \leq \tau \leq 1$ is the tension parameter and $0 \leq \rho \leq 1$ is a stiffness parameter. The thin-plate kernel affects surface “curvature” while the membrane kernel affects surface “area.”

sional spaces of all polynomials over \mathbb{R}^p of degree less than or equal to $m - 1$. Under certain conditions $\mathcal{E}(v)$ becomes a norm in \mathcal{H} , which guarantees existence, uniqueness, and stability of the minimizing function $u(\mathbf{x})$. A possible set of conditions is that the data points be \mathcal{N} -unisolvant, meaning that they define a unique polynomial in \mathcal{N} .

³Minimization of the true surface area leads to the famous Plateau’s problem.

5 Discontinuity-Preserving Generalized Splines

As introduced in [15], a simple but very consequential generalization is to allow ρ and τ in (6) to be functions $\rho(x, y)$ and $\tau(x, y)$:

$$\mathcal{S}_{\rho\tau}(v) = \frac{1}{2} \iint_{\mathbb{R}^2} \rho(x, y) \left((1 - \tau(x, y))(v_{xx}^2 + 2v_{xy}^2 + v_{yy}^2) + \tau(x, y)(v_x^2 + v_y^2) \right) dx dy. \quad (7)$$

This enables us to control the continuity of the resulting spline $u(x, y)$. In particular, we can explicitly introduce both zeroth and first order discontinuities at will over the bivariate domain. The smoothness of the spline $u(x, y)$ is controlled as follows: In continuous regions $\rho(x, y) = 1$ and $\tau(x, y) = 0$, so that the functional reduces to a thin-plate spline and generates a C^1 surface. Along orientation discontinuities $\rho(x, y) = 1$ and $\tau(x, y) = 1$; i.e. maximum tension is applied so that the stabilizer reduces locally to a membrane functional, thus maintaining only C^0 continuity and allowing the surface to crease freely. Along depth discontinuities $\rho(x, y) = 0$, thus inhibiting all continuity and thereby allowing the surface to fracture freely.

By analogy, a convenient way to control the continuity properties of the generalized spline of order n in (2) is to blend it with generalized splines of orders less than n as follows:

$$\mathcal{S}_n(v) = \frac{1}{2} \sum_{m=1}^n \sum_{|j|=m} \frac{m!}{j_1! \dots j_p!} \int_{\mathbb{R}^p} w_j(\mathbf{x}) \left(\frac{\partial^m v(\mathbf{x})}{\partial x_1^{j_1} \dots \partial x_p^{j_p}} \right)^2 d\mathbf{x}. \quad (8)$$

The smoothness of this discontinuity-preserving generalized spline is controlled by the vector $\mathbf{w}(\mathbf{x})$ of continuity control functions $w_j(\mathbf{x})$. For instance, a discontinuity of order $k < n$ is permitted to occur at \mathbf{x}_0 in the limit as $w_j(\mathbf{x}_0) \rightarrow 0$ for $|j| > k$. Note that (8) reduces to (7) for $n = 2$, $p = 2$ and the obvious choice of functions $\mathbf{w}(\mathbf{x})$.

6 Parametric, Discontinuity-Preserving Generalized Splines

All of the aforementioned spline functionals are scalar-valued or univariate. To create parametric splines, we simply extend (8) to the vector-valued case, generalized spline functionals defined on a vector of coordinate functions $\mathbf{v}(\mathbf{x}) = [v_1(\mathbf{x}), \dots, v_q(\mathbf{x})]$:

$$\mathcal{S}_n(\mathbf{v}) = \frac{1}{2} \sum_{m=1}^n \sum_{|j|=m} \frac{m!}{j_1! \dots j_p!} \int_{\mathbb{R}^p} w_j(\mathbf{x}) \left| \frac{\partial^m \mathbf{v}(\mathbf{x})}{\partial x_1^{j_1} \dots \partial x_p^{j_p}} \right|^2 d\mathbf{x}. \quad (9)$$

7 Euler-Lagrange Equations

Consider the classical smoothing splines defined by minimizing $\mathcal{E}_m(v)$ in (1), but this time for a continuous data function $c(x)$; i.e., for $\mathcal{P}(v) = 1/2 \int_{\mathbb{R}} \alpha(x)(v(x) - c(x))^2 dx$, where $\alpha(x)$ is a non-negative weighting function. It is well known that the necessary condition for the minimum of satisfies the Euler-Lagrange equation, $(-1)^m (\partial^{2m} v(x) / \partial x^{2m}) + \alpha(x)(v(x) - c(x)) = 0$, which expresses the vanishing of the first variation of $\mathcal{E}_m(v)$; i.e., $\delta_u \mathcal{E}_m(v) = 0$. For surface splines, the Euler-Lagrange equation for the membrane spline (4) involves the Laplacian operator $\Delta u = u_{xx} + u_{yy}$. For the thin-plate spline (5), the Euler-Lagrange equation involves the biharmonic operator $\Delta^2 u = u_{xxxx} + 2u_{xxyy} + u_{yyyy}$.

For the generalized splines (2), the Euler-Lagrange equation is

$$(-1)^m \Delta^m u(\mathbf{x}) + \alpha(\mathbf{x}) (u(\mathbf{x}) - c(\mathbf{x})) = 0, \quad (10)$$

where

$$\Delta^m = \sum_{|j|=m} \frac{m!}{j_1! \dots j_p!} \left(\frac{\partial^{2m}}{\partial x_1^{2j_1} \dots \partial x_p^{2j_p}} \right) \quad (11)$$

denotes the m th-order iterated Laplacian operator. For the discontinuity-preserving generalized splines (8), assuming that the control functions $\mathbf{w}(\mathbf{x})$ are differentiable to order p , the Euler–Lagrange equation is:

$$\sum_{m=0}^n (-1)^m \Delta_w^m u(\mathbf{x}) + \alpha(\mathbf{x}) (u(\mathbf{x}) - c(\mathbf{x})) = 0, \quad (12)$$

where

$$\Delta_w^m = \sum_{|j|=m} \frac{m!}{j_1! \dots j_p!} \frac{\partial^m}{\partial x_1^{j_1} \dots \partial x_p^{j_p}} \left(w_j(\mathbf{x}) \frac{\partial^m}{\partial x_1^{j_1} \dots \partial x_p^{j_p}} \right) \quad (13)$$

is a spatially weighted m th-order iterated Laplacian operator.

As a particular example of the discontinuity-preserving splines, the Euler–Lagrange equation for the thin-plate under tension spline (7) is

$$\frac{\partial^2}{\partial x^2} (\mu u_{xx}) + 2 \frac{\partial^2}{\partial x \partial y} (\mu u_{xy}) + \frac{\partial^2}{\partial y^2} (\mu u_{yy}) - \frac{\partial}{\partial x} (\eta u_x) - \frac{\partial}{\partial y} (\eta u_y) + \alpha(u - c) = 0, \quad (14)$$

where $\mu(x, y) = \rho(x, y)(1 - \tau(x, y))$ and $\eta(x, y) = \rho(x, y)\tau(x, y)$.

8 Global Basis Function Solution

A global method for computing generalized splines pursued by Duchon [4] and developed further in [8] and [18], involves a representation of the solution as a linear combination of N rotationally symmetric, global support basis functions, one centered at each data point. The Radial Basis Functions (RBFs) of choice are the fundamental solutions of the iterated Laplacian appearing in the Euler–Lagrange equations associated with the generalized splines. They are no more complicated than logarithms. Computing the solution requires first solving a system of linear equations for the unknown coefficients of the linear combination, then constructing the superposition of basis functions, restricted to a compact region of interest Ω in which the solution is continuous. The matrix of the linear system, whose size depends on the number of data points N , is positive definite, symmetric, and full. It can be solved by Cholesky factorization with back substitution.

Consider the generalized spline with

$$\mathcal{E}(v(\mathbf{x})) = \mathcal{S}v(\mathbf{x})_m^2 + \sum_{i=1}^N \frac{1}{\lambda \sigma_i^2} (\mathcal{L}_i(v) - c_i)^2 \quad (15)$$

where $\{\mathcal{L}_i\}_{i=1}^N$ are N linearly independent evaluation functionals. If $\{p_j(\mathbf{x})\}_{j=1}^M$ are a basis for the $M = \binom{p+m-1}{p}$ dimensional space of polynomials of total degree less than m , a unique (bounded) solution $u(\mathbf{x})$ will exist provided $N \geq M$, and $\mathcal{L}_i \left(\sum_{j=1}^M a_j p_j \right) = 0$, for $i = 1, 2, \dots, N$, implies that all the a_j are 0 [8].

The solution has a representation

$$u(\mathbf{x}) = \sum_{i=1}^N a_i \xi_i(\mathbf{x}) + \sum_{j=1}^M b_j p_j(\mathbf{x}), \quad (16)$$

where

$$\xi_i(\mathbf{x}) = \mathcal{L}_{i(\mathbf{s})} E_m(r(\mathbf{x} - \mathbf{s})), \quad i = 1, 2, \dots, N, \quad (17)$$

where $r(\mathbf{x}) = |\mathbf{x}| = (\sum_{i=1}^d x_i^2)^{1/2}$ denotes the Euclidean norm of \mathbf{x} , and the subscript (s) indicates that the functional is applied to what follows considered as a function of \mathbf{s} [18]. $E_m(\mathbf{x})$ is the fundamental solution of the iterated Laplacian Δ^m in \mathfrak{R}^d ; i.e., it satisfies the equation $\Delta^m E_m(\mathbf{x}) = \delta(\mathbf{x})$, where δ is the Dirac delta distribution. It turns out to be the rotation invariant function defined on $\mathfrak{R}^d - \{0\}$ given by

$$E_m(\mathbf{x}) = \begin{cases} \frac{(-1)^{d/2+1}}{2^{2m-1}\pi^{d/2}(m-1)!(m-d/2)!} r^{2m-d} \ln r, & \text{if } 2m \geq d \text{ and } d \text{ even;} \\ \frac{(-1)^m \Gamma(d/2 - m)}{2^{2m}\pi^{d/2}(m-1)!} r^{2m-d}, & \text{otherwise,} \end{cases} \quad (18)$$

The coefficients $\mathbf{a} = [a_1, \dots, a_N]^\top$ and $\mathbf{b} = [b_1, \dots, b_M]^\top$ are determined by the linear systems

$$\begin{aligned} (\mathbf{K} + \lambda \Sigma^2) \mathbf{a} + \mathbf{T} \mathbf{b} &= \mathbf{c}, \\ \mathbf{T}^\top \mathbf{a} &= 0, \end{aligned} \quad (19)$$

where $\mathbf{c} = [c_1, \dots, c_N]^\top$, the $N \times N$ symmetric matrix $\mathbf{K} = [\mathcal{L}_{i(\mathbf{x})} \mathcal{L}_{j(\mathbf{s})} E_m(r(\mathbf{x} - \mathbf{s}))]$, the $N \times M$ matrix $\mathbf{T} = [\mathcal{L}_i p_j]$, and the $N \times N$ diagonal matrix $\Sigma = [\sigma_i]$ [18].

To construct the global solution, the Cholesky algorithm recursively computes a sequence of optimal approximations, as each data point is accessed in sequence. This has the advantage that the addition or removal of a single data point requires relatively little computation to determine the new global solution, given the current approximation. Although the global method may be attractive for problems involving a modest number of constraints, it becomes expensive to store and solve an $N \times N$ linear system with full matrix, if N exceeds order $10^2 - 10^3$, or so. In addition, the system tends to become ill-conditioned for large N .

9 Local Basis Function Solutions

An alternative approach to solving generalized spline problems is to apply the finite element or finite difference methods which make use of local basis functions. While the size of the resulting linear system is usually greater than N , its matrix is sparse, due to the local support of the basis functions [9].

As a concrete example, we discretize the thin-plate spline under tension spline (7). Uniformly discretizing a rectangular subdomain of $\Omega \in \mathfrak{R}^2$ using a uniform grid with spacing h , whose nodes $\{ih, jh\} \cap \Omega$ are indexed by (i, j) . We define the nodal variables $u_{i,j} = u(ih, jh)$ as the unknown depths of the surface at the N nodes (i, j) . Taken together, these nodal variables form the vector \mathbf{v} , to be determined by solving the discrete problem. Similarly, nodal parameters $\rho_{i,j} = \rho(ih, jh)$ and $\tau_{i,j} = \tau(ih, jh)$ represent the parameter functions of the surface. Letting $\mu_{i,j} = \rho_{i,j}(1 - \tau_{i,j})/h^2$ and $\eta_{i,j} = \rho_{i,j}\tau_{i,j}$, and applying finite differences

to (14), the nodal equation for the solution $u_{i,j}$ at an arbitrary node (i, j) is given by

$$\begin{aligned}
& \{ (u_{i,j} - 2u_{i-1,j} + u_{i-2,j}) \mu_{i-1,j} \\
& + (-2u_{i+1,j} + 4u_{i,j} - 2u_{i-1,j}) \mu_{i,j} \\
& + (u_{i+2,j} - 2u_{i+1,j} + u_{i,j}) \mu_{i+1,j} \\
& + (2u_{i,j} - 2u_{i-1,j} - 2u_{i,j-1} + 2u_{i-1,j-1}) \mu_{i-1,j-1} \\
& + (-2u_{i+1,j} + 2u_{i,j} + 2u_{i+1,j-1} - 2u_{i,j-1}) \mu_{i,j-1} \\
& + (-2u_{i,j+1} + 2u_{i-1,j+1} + 2u_{i,j} - 2u_{i-1,j}) \mu_{i-1,j} \\
& + (2u_{i+1,j+1} - 2u_{i,j+1} - 2u_{i+1,j} + 2u_{i,j}) \mu_{i,j} \\
& + (u_{i,j} - 2u_{i,j-1} + u_{i,j-2}) \mu_{i,j-1} \\
& + (-2u_{i,j+1} + 4u_{i,j} - 2u_{i,j-1}) \mu_{i,j} \\
& + (u_{i,j+2} - 2u_{i,j+1} + u_{i,j}) \mu_{i,j+1} \} \\
& + \{ (u_{i,j} - u_{i-1,j}) \eta_{i-1,j} + (u_{i,j} - u_{i+1,j}) \eta_{i,j} \\
& + (u_{i,j} - u_{i,j-1}) \eta_{i,j-1} + (u_{i,j} - u_{i,j+1}) \eta_{i,j} \} \\
& + \alpha_{i,j}(u_{i,j} - c_{i,j}) = 0.
\end{aligned} \tag{20}$$

From the above equation, it is clear that $\mathbf{A}\mathbf{u} = \mathbf{c}$ is a sparse system of dimensionality equal to the number of nodes in the grid.

In the closely related finite element solution, the approximation is a linear combination of the local support basis functions (typically low-order piecewise polynomials) of finite element spaces. The number of basis functions depends on the number of finite elements employed to tessellate the continuous domain. The choice of tessellation is very flexible. As in the finite difference approach, the finite element representation of generalized splines leads to a positive definite, symmetric system of linear equations that must be solved for the unknown coefficients. This suggests the use of either direct or iterative sparse matrix techniques to compute the solution. Iterative sparse matrix techniques need store only the nonzero matrix entries and relaxation methods operate on these entries in place. With this approach we can tractably handle millions of data points. Efficient multigrid relaxation methods have been applied successfully to the finite element and finite difference solution of generalized spline problems [16].

References

- [1] Ahlberg, J.H., Nilson, E.N., and Walsh, J.L., *The Theory of Splines and their Applications* Academic Press, New York, 1967.
- [2] Cline, A.K., Scalar- and planar-valued curve fitting using splines under tension, *Comm. ACM*, **17**, 218–220, 1974.
- [3] Courant, R., and Hilbert, D. *Methods of Mathematical Physics*, Vol. I Interscience, London, 1953.
- [4] Duchon, J. Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces, *R.A.I.R.O. Analyse Numérique*, **10**, 5–12, 1976.
- [5] Duchon, J. Splines minimizing rotation-invariant semi-norms in Sobolev spaces, *Constructive Theory of Functions of Several Variables*, A. Dodd and B. Eckmann (eds.), Springer-Verlag, Berlin, 85–100, 1977.
- [6] Franke, R., “Scattered data interpolation: tests of some methods,” *Math. Comp.*, **38**, 1982, 181–199.

- [7] Lancaster, P., and Salkauskas, K., *Curve and Surface Fitting: An Introduction*, Academic Press, New York, 1986.
- [8] Meinguet, J., Multivariate interpolation at arbitrary points made simple, *Jour. Applied Math. and Physics (ZAMP)*, **30**, 292–304, 1979.
- [9] Press, W., Flannery, B., Teukolsky, S., and Vetterling, W., *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, Cambridge, 1986.
- [10] Reinsch, G., Smoothing by spline functions, *Numer. Math.*, **10**, 177–183, 1967.
- [11] Schoenberg, I.J., Spline functions and the problem of graduation, *Proc. National Academy of Sciences*, **52**, 947–950, 1964.
- [12] Schumaker, L.L., “Fitting surfaces to scattered data,” *Approximation II*, G.G. Lorentz, C.K. Chui, L.L. Schumaker (eds), Academic Press, New York, 1976, 203–267.
- [13] Schweikert, D.G., An interpolation curve using a spline in tension, *J. Math. and Physics*, **45**, 312–317, 1966.
- [14] R. Szeliski, D. Terzopoulos, “From splines to fractals,” *Computer Graphics*, **23**, 3, 1989, 51–60.
- [15] D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(4):413–424, 1986.
- [16] D. Terzopoulos, “The computation of visible-surface representations,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-10**, 4, 1988, 417–438.
- [17] Tikhonov, A.N., and Arsenin, V.A., *Solutions of Ill-Posed Problems*, Winston and Sons, Washington, DC, 1977.
- [18] Wahba, G., and Wendelberger, J., Some new mathematical methods for variational objective analysis using splines and cross validation, *Monthly Weather Review*, **108** 1122–1143, 1980.

Fitting Parametric Curves to Dense and Noisy Points *

A. Ardeshir Goshtasby

Abstract—Given a large set of irregularly spaced points in the plane, an algorithm for partitioning the points into subsets and fitting a parametric curve to each subset is described. The points could be measurements from a physical phenomenon, and the objective in this process could be to find patterns among the points and describe the phenomenon analytically. The points could be measurements from a geometric model, and the objective could be to reconstruct the model by a combination of parametric curves. The algorithm proposed here can be used in various applications, especially where given points are dense and noisy.

1 Introduction

In many science and engineering problems there is a need to fit a curve or curves to an irregularly spaced set of points. Curve fitting has been studied extensively in Approximation Theory and Geometric Modeling, and there are numerous books on the subject [1,5,6,12,23]. Existing techniques typically find a single curve segment that approximates or interpolates the given points. Many techniques assume that the points are ordered and fit a curve to them by minimizing an error criterion [3,7,8,14,16,22,27,29,31,34]. If the points are ordered, piecewise polynomial curves can also be fitted to them [19,30]. Difficulties arise when the points are not ordered.

To fit curves to an irregularly spaced set of points: 1) the set should be partitioned into subsets, 2) the points in each subset should be ordered, and 3) a curve should be fitted to points in each subset. This paper will provide solutions to the first two problems; that is, partitioning a point set into subsets and ordering the points in each subset. Once the points in each subset are ordered, existing techniques can be used to find the curves.

Given a large set of irregularly spaced points in the plane, $\{\mathbf{p}_i = (x_i, y_i) : i = 1, \dots, N\}$, we would like to fit one or more parametric curves to the points, with the number of the curves to depend on the organization of the points and the resolution of the representation. When fitting a parametric curve to an irregularly spaced set of points, the main problem is to find the nodes of the curve. The nodes of a parametric curve determine the adjacency relation between the points and order the points. The curve will then approximate the points in the order specified. Methods to order sparse points [11,17,24] as well as dense points [25,26,32] have been developed. Existing methods, however, fit a single curve segment to an entire data set. Sometimes it is not desirable to fit a single curve segment to a large and complex point set, and it is necessary to represent the geometric structure present in the point set by many curve segments. In this paper it will be shown how to partition a point set into subsets and how to fit a parametric curve to each subset. A new method to order a set of dense and noisy points for curve fitting will also be presented.

*Presented at *4th Int'l Conf. Curves and Surfaces*, St. Malo, France, July 1–7, 1999.

In the proposed model, a radial field is centered at each point such that the strength of the field monotonically decreases as one moves away from the point. The sum of the fields has the averaging effect and reduces the effect of noise, and local maxima of the sum of the fields has the effect of tracing the spine of the points. Therefore, we will use the local maxima of the sum of the fields (the ridges of the obtained field surface) as an approximation to the curves to be determined. Based on the organization of the points, disjoint ridges may be obtained, each suggesting a curve. The ridges will be used to partition the points into subsets and fit a curve to each subset. In the following, the steps of this process are described in detail.

2 Approach

A desirable property of an approximating curve is for it to pass as close as possible to the given points while providing a certain smoothness appearance. For a dense point set, the curve cannot pass close to all the points, so it is desired that the curve trace the spine of the points. In the model proposed here, an initial estimation to a curve is obtained by taking points in the xy plane whose sum of inverse distances to the given points is locally maximum. That is, if the sum of inverse distances of point (x, y) to given points $\{(x_i, y_i) : i = 1, \dots, N\}$ is larger than the sum of inverse distances of points in the neighborhood of (x, y) to the given points, then point (x, y) is considered an initial estimation to a point on the curve. Therefore, by tracing points in the xy plane that locally maximize

$$f(x, y) = \sum_{i=1}^N [(x - x_i)^2 + (y - y_i)^2 + 1]^{-\frac{1}{2}}, \quad (1)$$

we find an approximation to the curves we want to find.

The function f can also be interpreted as follows: Suppose a radial field of strength 1 is centered at point (x_i, y_i) , $i = 1, \dots, N$, such that the strength of the field decreases with inverse distance as one moves away from the point. Then, the strength of the field at point (x, y) will be $[(x - x_i)^2 + (y - y_i)^2 + 1]^{-\frac{1}{2}}$, and the curves to be found can be considered points in the xy plane whose sum of field values are locally maximum.

Once a set of points is given, the function f becomes fixed, and the obtained ridges will have a fixed shape. In order to provide control over the shape or smoothness of obtained ridges, we revise formula (1) as follows: If instead of inverse distances defined by $[(x - x_i)^2 + (y - y_i)^2 + 1]^{-\frac{1}{2}}$, we use

$$[(x - x_i)^2 + (y - y_i)^2 + r^2]^{-\frac{1}{2}} \quad (2)$$

in equation (1), we obtain

$$g(x, y) = \sum_{i=1}^N [(x - x_i)^2 + (y - y_i)^2 + r^2]^{-\frac{1}{2}}. \quad (3)$$

The basis functions defined by (2) are known as inverse multiquadrics [13]. The parameter r of the basis functions can be varied to generate different surfaces [21]. Figure 1b shows the field surface obtained when using the points of Fig. 1a and inverse multiquadric basis functions with $r = 5$.

Instead of inverse multiquadric basis functions, other radial basis functions [2,4,10,28,33,35] also can be used to define function g . The choice of the basis functions influences the shape of

the obtained field surface, the shape of the obtained ridges, and, consequently, the shape of the obtained curves.

By tracing the local maxima of field surface g in the xy plane, we will obtain an approximation to the curves. Parameter r changes the shape of the basis functions and affects the shape of the field surface.

Local maxima of surface g can result in structures that contain branches and loops. The proposed model, therefore, can recover very complex patterns in dense and noisy point sets. Note also that the proposed method does not require any knowledge about the adjacency relation between the points. This method, in fact, provides the means to determine the adjacency relation between the points.

3 Implementation

Derivation of an analytic formula that represents the local maxima of surface g may not be possible. Digital approximation to the local maxima, however, is possible. This approximation is found in the form of digital contours and is used to partition the points into subsets. To digitally trace surface ridges, the surface is digitized into a digital image. The digitization process involves starting from $x = x_{min}$ and $y = y_{min}$ and incrementing x and y by some small increment δ until reaching $x = x_{max}$ and $y = y_{max}$. For each discrete (x, y) , the value for $g(x, y)$ is then found from formula (3). x_{min} and x_{max} could be the smallest and largest x coordinates, and y_{min} and y_{max} could be the smallest and largest y coordinates of the given points. Parameter δ is used as the increment for both x and y because radially symmetric basis functions are used to define g . This parameter determines the resolution of the obtained image. For a finer resolution, this parameter should be reduced, while for a coarser resolution this parameter should be increased. If this parameter is to be chosen automatically, it should be selected such that most given points map to unique pixels in the obtained image.

Digitizing surface g in this manner will result in a digital image whose pixel values show uniform samples from surface g . Figure 1b shows digitization of a field surface into an image of 256×256 pixels. To find the image ridges, pixels with locally maximum intensities are located. To find locally maximum image intensities, the gradient magnitude and the gradient direction [20] of the image at each pixel are determined. Gradient direction at a pixel is the direction at which change in intensity at the pixel is maximum, and gradient magnitude is the magnitude of the intensity change in the gradient direction at the pixel.

To find the ridges, we find each pixel A in the image where two pixels B and C that are adjacent to it and are at its opposite sides have intensities that are smaller than that at A . Assuming that the image obtained after digitizing surface g is represented by I , we mark the pixel at (i, j) as A if one of the following is true:

$$I(i - 1, j) < I(i, j) \quad \& \quad I(i + 1, j) < I(i, j); \quad (4)$$

$$I(i, j - 1) < I(i, j) \quad \& \quad I(i, j + 1) < I(i, j); \quad (5)$$

$$I(i - 1, j - 1) < I(i, j) \quad \& \quad I(i + 1, j + 1) < I(i, j); \quad (6)$$

$$I(i - 1, j + 1) < I(i, j) \quad \& \quad I(i + 1, j - 1) < I(i, j). \quad (7)$$

Using the image of Fig. 1b, we find that pixels in the contours shown in Fig. 1c are marked as A . We will call the contours obtained in this manner the *minor ridges* of the image. Next, we

find each pixel D whose value is not only larger than those of B and C adjacent to it and at its opposite sides, but which also has a gradient direction that is the same as the direction obtained by connecting pixels B and C . The gradient direction at a pixel is quantized with 45-degree steps to ensure that only directions that are possible to obtain when connecting pixels B and C in an image are obtained. The pixels marked as D are shown in Fig. 1d. We will call these contours the *major ridges* of the image. As can be observed, major ridges are a subset of minor ridges. We also see that major ridge points do not fall on small and noisy branches of the minor ridges but rather fall on contours that represent the spines of the points. If the minor ridges are cut at the branch points, and branches that do not contain a major ridge point are removed, and if the remaining contours are thinned, we obtain Fig. 1e. The obtained contours will be called the *local-maxima contours*, or simply the *contours*. These contours will be taken as approximations to the curves to be found. We will use them not only to partition the points into subsets but also to order the points in the subsets.

4 Node Estimation

The method outlined in the preceding section determines contours that are approximations to the curves to be found. These contours will be used to partition a point set into subsets and order the points in each subset.

Suppose a point set has produced m contours; then, a point is assigned to contour j ($1 \leq j \leq m$) if it is closest to a pixel in contour j than to a pixel in any other contour. In this manner, a point is assigned to one of m contours. This process, when completed, will partition a point set into m subsets by assigning the points into one of m contours. Figures 2a and 2b show the point subsets obtained in this manner from the point set of Fig. 1a.

To order points $\{\mathbf{q}_i : i = 1, \dots, n\}$ in subset j , for each point \mathbf{q}_i a point in contour j that is closest to it is determined. We call the obtained contour point the *projection* of point \mathbf{q}_i . After determining projections of all points in the subset to the contour, the contour is traced from one end to the other, and in the order the projections are visited, the associated points are ordered.

Since the contours are approximations to the curves to be found, the contour length from a projection to the start of the contour is divided by the length of the contour to obtain an arc-length estimation to the node of the point. If the contour is closed, an arbitrary point on the contour is taken as the start point. If the contour is open, one of the end points is taken as the start point.

The size of the image obtained by digitizing surface g determines the accuracy of the obtained nodes. If the surface g is very coarsely digitized, the obtained contours will be very short, and numerous points may produce the same node, especially when given points are dense. To provide a more accurate node estimation, the surface g should be digitized into an image large enough to produce unique nodes.

Once the coordinates of given points and the associated nodes are known, a parametric curve can be fitted to the points by one of the existing methods [9,11,16,18,30]. Fitting rational Gaussian (RaG) curves [9] to the points shown in Fig. 1a with nodes as determined above, we obtain the curves shown in Fig. 1f. The curves are overlaid with the original points to show the quality of the curve fitting. Note that these curves were obtained using the points in Fig. 1a and not the contour points in Fig. 1e. The contour points were used only to partition a point set into subsets and to determine the nodes of the points.

5 Observations

To observe the behavior of the proposed curve-fitting method, results on three additional point sets are shown in Fig. 3. Figure 3a shows noisy points along an open contour, Fig. 3c shows a dense and noisy point set along the silhouette of a coffee mug, Fig. 3e shows irregularly spaced points along the silhouette of a model plane and one of its wings. We can see the geometric structures in these point sets and, if asked, can trace the structures manually without any difficulty. The algorithm proposed here is intended to do the same. The curves obtained are shown in Figs. 3b, 3d, and 3f.

The point sets shown in Fig. 3 did not contain geometric structures with branches and loops. If a point set contains branches and loops, the local-maxima contours will also contain branches and loops. A single curve segment, however, cannot represent branching structures. The solution we propose is to segment a complex contour into simple ones by cutting it at the branch points and fitting a curve to each branch.

6 Summary and Conclusions

A large number of techniques for fitting parametric curves to irregularly spaced points have been developed. These techniques fit a single curve to the given points and often require that the points be ordered. In science and engineering problems that deal with measurement data, the given points may not be ordered and they may contain noise. Moreover, it may not be appropriate to fit a single curve segment to all of the points. In this paper, a method to partition a point set into subsets and fit a parametric curve to each subset was described. The proposed method has the ability to take into consideration the noisiness and denseness of a point set when obtaining the curves.

Also introduced was a method to determine the nodes of a parametric curve that approximates a set of dense and noisy points. The proposed method provides the means to fit any parametric curve, including B-Splines and Non-Uniform Rational B-Splines, to irregularly spaced points. Although in this paper only inverse multiquadrics were used as basis functions to obtain a field surface from which the curve segments were determined, other radial basis functions [33] can be used in the same manner. Depending on the parametric curve formulation and the radial basis functions used, the number and the shapes of the curves fitting to a set of points may vary.

References

- [1] Beach, R. C., *An Introduction to the Curves and Surfaces of Computer-Aided Design*, Van Nostrand: New York, 1991.
- [2] Buhmann, M. D., Multivariate cardinal interpolation with radial basis functions, *Computer Aided Geometric Design*, vol. 6 (1990), 225–255.
- [3] Cohen, E. and C. L. O’Dell, A data dependent parametrization for spline approximation, *Mathematical Methods in Computer Aided Geometric Design*, T. Lyche and L. L. Schumaker (eds.), 1989, 155–166.
- [4] Dyn, N., Interpolation of scattered data by radial functions, *Topics in Multivariate Approximation*, Academic Press, 1987, 47–61.

- [5] Farin, G., *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, New York, 1988.
- [6] Faux, I. D. and M. J. Pratt, *Computational Geometry for Design and Manufacture*, Ellis Horwood, Chichester, 1979.
- [7] Foley, T. A. and G. M. Nielson, Knot selection for parametric spline interpolation, *Mathematical Methods in Computer Aided Geometric Design*, T. Lyche and L. L. Schumaker (eds.), 1989.
- [8] Fritsch, F. N. and R. E. Carlson, Monotone piecewise cubic interpolation, *SIAM J. Numerical Analysis*, vol. 17, no. 2 (1980), 238–246.
- [9] Goshtasby, A., Geometric modeling using rational Gaussian curves and surfaces, *Computer Aided Geometric Design*, vol. 27, no. 5 (1995), 363–375.
- [10] Goshtasby, A. and W. D. O’Neill, Surface fitting to scattered data by a sum of Gaussians, *Computer Aided Geometric Design*, vol. 10 (1993), 143–156.
- [11] Grossman, M., Parametric curve fitting, *Computer Journal*, vol. 14, no. 2 (1970), 169–172.
- [12] Hagen, H., *Curve and Surface Design*, SIAM, 1992.
- [13] Hardy, R. L., Theory and applications of the multiquadrics–biharmonic method, *Comput. Math. Appl.*, vol. 19, no. 8/9 (1990), 163–208.
- [14] Hartley, P. J. and C. J. Judd, Parametrization and shape of B-spline curves for CAD, *Computer Aided Design*, vol. 12, no. 5 (1980), 235–238.
- [15] Hölzle, G. E., Knot placement for piecewise polynomial approximation of curves, *Computer Aided Design*, vol. 15, no. 5 (1983), pp 295–296.
- [16] Hoschek, J., Approximate conversion of spline curves, *Computer Aided Geometric Design*, vol. 4 (1987), 59–66.
- [17] Hoschek, J., Intrinsic parametrization for approximation, *Computer Aided Geometric Design*, vol. 5 (1988), 27–31.
- [18] Hoschek, J., Spline approximation of offset curves, *Computer Aided Geometric Design*, vol. 5 (1988), 33–40.
- [19] Ichida, K. and T. Kiyono, Curve fitting by a one-pass method with a piecewise cubic polynomial, *ACM Trans. Math. Software*, vol. 3, no. 2 (1977), 164–174.
- [20] Jain, A. K., *Fundamentals of Digital Image Processing*, Prentice Hall, 1989, 347–349.
- [21] Kansa, E. J., Multiquadrics—A scattered data approximation scheme with applications to computational fluid dynamics—I, *Comput. Math. Appl.*, vol. 19, no. 8/9 (1990), 127–145.
- [22] Kusters, M., Curvature-dependent parametrization of curves and surfaces, *Computer Aided Design*, vol. 23, no. 8 (1991), 569–578.

- [23] Lancaster, P. and K. Šalkauskas, *Curve and Surface Fitting*, Academic Press, New York, 1986.
- [24] Lee, E. T. Y., Choosing nodes in parametric curve interpolation, *Computer Aided Design*, vol. 6 (1989), 363–370.
- [25] Lee, I.-K., Curve reconstruction from unorganized points, *Computer Aided Geometric Design*, vol. 17 (2000), 161–177.
- [26] Levin, D., The approximation power of moving least-squares, *Math. Comp.*, vol. 67 (1998), 1517–1531.
- [27] Marin, S. P., An approach to data parametrization in parametric cubic spline interpolation problem, *J. Approx. Theory*, vol. 41 (1984), 64–86.
- [28] Meinguet, J., An intrinsic approach to multivariate spline interpolation at arbitrary points, *Polynomial and Spline Approximation*, B. N. Sahney (ed.), D. Reidel Publishing, 1979, 163–190.
- [29] Mullineux, M., Approximating shapes using parametrized curves, *IMA J. Applied Mathematics*, vol. 29 (1982), 203–220.
- [30] Piegl, L., A technique for smoothing scattered data with conic sections, *Computers in Industry*, vol. 9 (1987), 223–237.
- [31] Plass, M. and M. Stone, Curve-fitting with piecewise parametric cubics, *Computer Graphics*, vol. 17, no. 3 (1983), 229–239.
- [32] Pottmann, H. and T. Randrup, Rotational and helical surface approximation for reverse engineering, *Computing*, vol. 60 (1998), 307–322.
- [33] Powell, M. J. D., Radial basis functions for multivariable interpolation: A review, *Algorithms for Approximation*, J. C. Mason and M. G. Cox (eds.), Clarendon Press, Oxford, 1987, 143–167.
- [34] Sarkar, B. and C-H Menq, Parameter optimization in approximating curves and surfaces in measurement data, *Computer Aided Geometric Design*, vol. 8 (1991), 267–290.
- [35] Schagen, I. P., The use of stochastic processes in interpolation and approximation, *Int. J. Computer Math.*, Section B, vol. 8 (1980), 63–76.

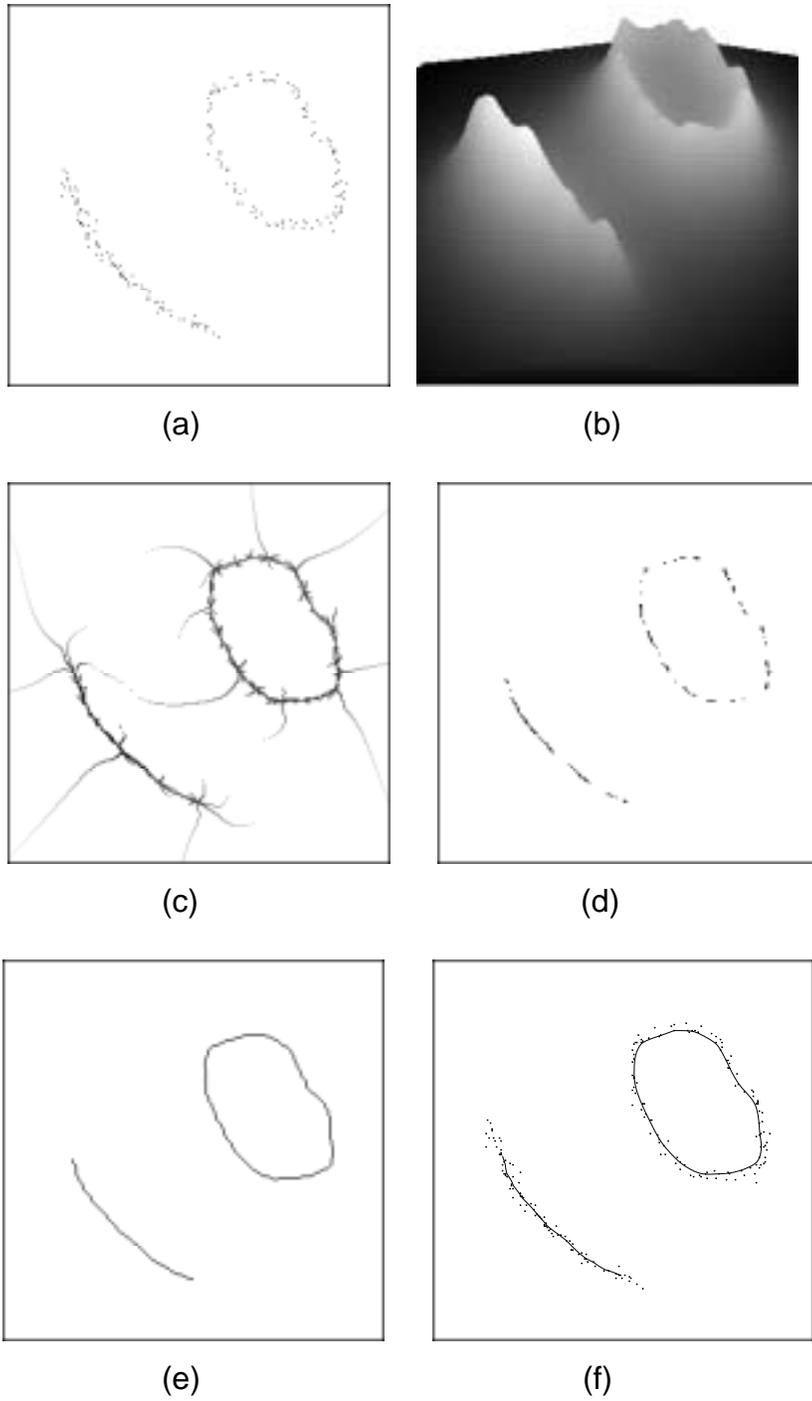


Fig. 1. (a) An irregularly spaced set of points. (b) A digitized field surface. (c) Contours representing the minor ridges. (d) Contours representing the major ridges. (e) Local-maxima contours. (f) RaG curve with standard deviation = 0.04 fitting points in (a).



(a)



(b)

Fig. 2. (a), (b) Two point subsets obtained from the point set of Fig. 1a.

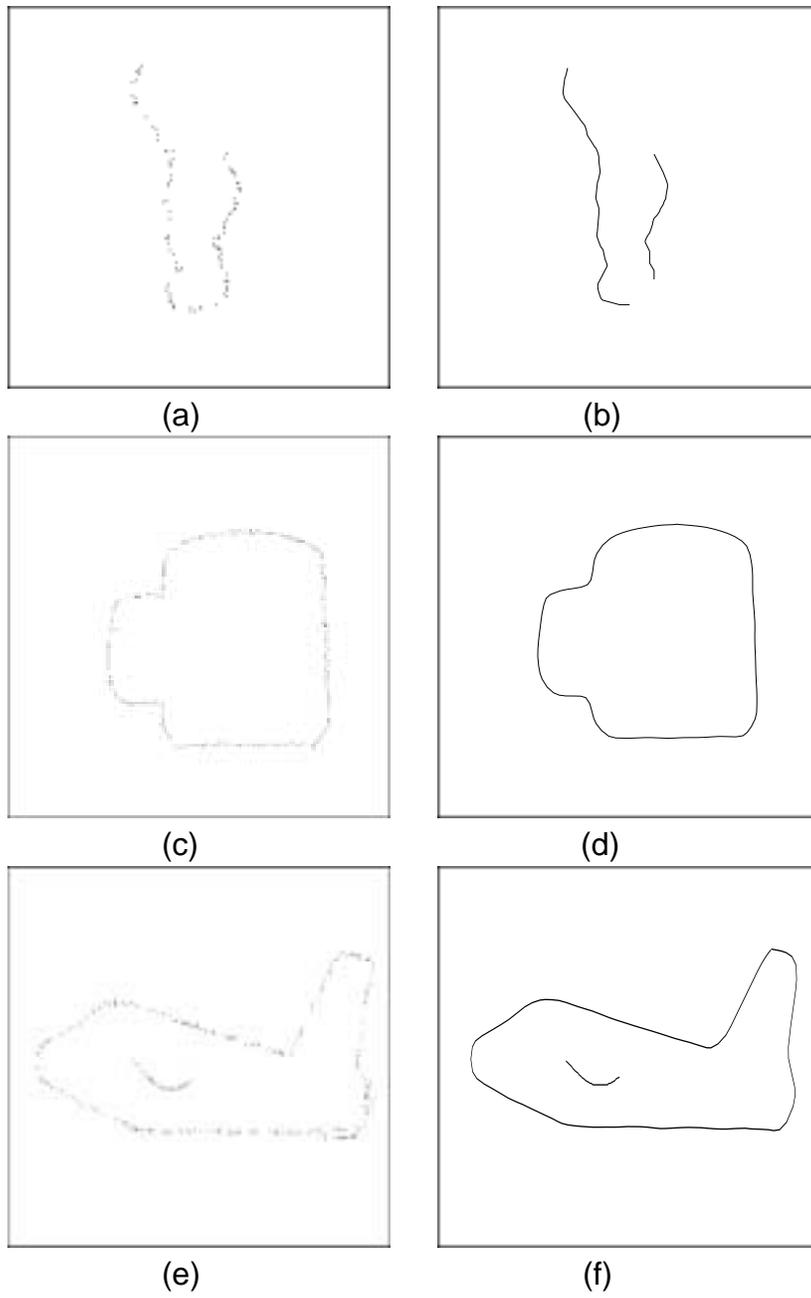


Fig. 3. A few curve-fitting examples.

Approximating Digital Shapes by Parametric Surfaces *

A. Ardeshir Goshtasby

Abstract—A method for parametrizing discrete points sampled from a smooth shape and fitting a rational Gaussian surface to the points with a required accuracy is presented. With the proposed method, a very complex shape can be represented by a single surface, and the surface can be rendered at a desired level of detail by adjusting a smoothness parameter.

1 Introduction

Given a set of scattered points in 3-D, $\{\mathbf{p}_i = (x_i, y_i, z_i) : i = 1, \dots, n\}$, we would like to determine a parametric surface $\mathbf{P}(u, v)$ that approximates the points with a required error tolerance:

$$\max_i \|\mathbf{P}(u_i, v_i) - \mathbf{p}_i\| < \varepsilon. \quad (1)$$

ε is the error tolerance and (u_i, v_i) are the parameters at \mathbf{p}_i .

We will consider a special case where the given points are voxels covering an object in a discretized 3-D space. We will call such a data set a *digital shape*. Digital shapes are typically obtained by segmenting tomographic images obtained by industrial or medical scanners. We assume that the given shape is closed and contains no holes.

The objective in this approximation is threefold. First, we want to approximate the points with a parametric surface where the number of control points in the surface is much smaller than the number of points in the shape, and the maximum distance between the shape points and the surface is within a required tolerance. Second, we want to have the ability to revise the obtained surface so that a reconstructed shape can be edited. Therefore, the surface formulation used should lend itself to easy editing. Third, the obtained surface should smooth noise among the given points, with the degree of smoothing adjustable by the user.

In the following sections, first, an algorithm to parametrize a set of points by mapping them to a sphere is described. Then, surface fitting using rational Gaussian (RaG) surfaces is discussed. Finally, examples of the proposed surface-fitting method using medical data are presented.

2 Definitions

In this section, terminologies used in the paper are defined.

Digital shape: A set of points covering a smooth shape in a discretized 3-D space.

Point: A voxel in a digital shape. Shape points will be denoted by \mathbf{p} 's.

*Presented at *5th Int'l Conf. Curves and Surface*, Oslo, Norway, June 28 - July 4, 2000.

Adjacent points: Two points, \mathbf{p}_i and \mathbf{p}_j , are considered adjacent if $1 \leq \|\mathbf{p}_i - \mathbf{p}_j\| \leq \sqrt{3}$. Two adjacent points are also called **neighbors**.

Connected points: Two points are said to be connected if a path can be formed between them by connecting adjacent points.

Path: A path between points \mathbf{p}_i and \mathbf{p}_j is a connected set of points starting from \mathbf{p}_i and ending at \mathbf{p}_j where no point is repeated and every point has exactly two neighbors except for \mathbf{p}_i and \mathbf{p}_j , which may have only one neighbor. A path is also called a **contour**.

Triangular mesh approximation of a shape: A triangular mesh that interpolates some points and approximates the rest in a shape. The mesh vertices will be denoted by \mathbf{P} 's. Note that \mathbf{P} 's are a subset of the \mathbf{p} 's.

Edge contour: A contour in a digital shape that is delimited by the end points of a mesh edge and lies in the plane passing through the edge and bisecting the angle between the two triangular faces that share the edge. Note that due to the digital nature of shape points, some points in an edge contour may not fall exactly in the bisecting plane; however, they will be closer to the plane than points in any other path connecting the edge end points.

Distance of point \mathbf{p}_i to edge $\mathbf{P}_j\mathbf{P}_k$: Assuming the plane passing through the point and normal to the edge intersects the edge at \mathbf{P}_l , if \mathbf{P}_l is between \mathbf{P}_j and \mathbf{P}_k , the distance will be $\|\mathbf{p}_i - \mathbf{P}_l\|$. Otherwise, if \mathbf{P}_l is closer to \mathbf{P}_j than to \mathbf{P}_k , the distance will be $\|\mathbf{p}_i - \mathbf{P}_j\|$, and if \mathbf{P}_l is closer to \mathbf{P}_k than to \mathbf{P}_j , the distance will be $\|\mathbf{p}_i - \mathbf{P}_k\|$.

Distance of an edge to an edge contour: Assuming \mathbf{p}_i is a contour point with distance d_i to the associating edge, we will take the maximum distance from points on the contour to the edge as the distance of the contour to the edge. That is, $D_e = \max_i \{d_i\}$.

Distance of point \mathbf{p}_i to triangular face $\mathbf{P}_j\mathbf{P}_k\mathbf{P}_l$: Assuming the line passing through \mathbf{p}_i and normal to the triangle intersects the triangle at \mathbf{P}_m , if \mathbf{P}_m is inside the triangle, the distance will be $\|\mathbf{p}_i - \mathbf{P}_m\|$. If \mathbf{P}_m is outside the triangle and assuming \mathbf{P}_n is the point on a triangle edge closest to \mathbf{P}_m , the distance will be $\|\mathbf{p}_i - \mathbf{P}_n\|$.

Triangular patch: A connected set of points in a digital shape delimited by three contours whose end points are the vertices of a triangle.

Distance of a triangular patch to the associating triangle: Assuming point \mathbf{p}_i belongs to the triangular patch and the distance between \mathbf{p}_i and the triangle is d_i , the distance to be determined is the maximum of such distances when all points in the patch are tested. That is, $D_t = \max_i \{d_i\}$.

Major axis of a digital shape: This is the axis defined by the largest eigenvector of the inertia matrix [2] of points defining the shape.

Subdividing edge $\mathbf{P}_j\mathbf{P}_k$: Replacing the edge with edges $\mathbf{P}_j\mathbf{P}_i$ and $\mathbf{P}_i\mathbf{P}_k$, where \mathbf{P}_i is the farthest point on the associating edge contour to the edge and distance of \mathbf{P}_i to the edge is larger than the given tolerance ε .

Subdividing a triangle: A triangle may be subdivided in four different ways:

1. If distances between all three edges of the triangle and the corresponding contours are larger than the specified tolerance, then by subdividing each edge into two and connecting the obtained points to each other and to the vertices of the triangle, four smaller triangles are obtained.
2. If distances between two of the edges and corresponding contours are larger than the specified tolerance, then two of the edges are subdivided. By connecting the newly obtained points to each other and to the vertices of the triangle, three new smaller triangles are obtained.

3. If the distance between only one of the edges and the corresponding contour is larger than the required tolerance, then only one of the edges is subdivided into two. By connecting the newly obtained point to the opposing triangle vertex, two smaller triangles are obtained.
4. If distances between all three edges and the corresponding contours do not reach the required tolerance, then the distance between the patch and the triangle is determined, and if it is larger than the required tolerance, the point in the patch farthest from the triangle is connected to the vertices of the triangle to produce three smaller triangles.

In this way, a triangle is subdivided into 2, 3, or 4 smaller triangles. Subdivision will take place in the order specified above. That is, only if subdivision by case 1 is not possible will subdivision by case 2 be considered, and subdivision by case 3 will be considered only when subdivision by case 2 is not possible, and so on.

Parametrizing points in an edge contour: By knowing the parameters at the end points of an edge, parameters along the edge are determined by linear interpolation. Parameters at a contour point are then set equal to the parameters at the edge point closest to it. At a coarse resolution, multiple contour points may map to the same edge point, thus producing the same parameters. However, as the subdivision proceeds, contour points will be more likely to map to unique edge points, producing unique parameters.

Parametrizing points in a triangular patch: By knowing parameters at the vertices of a triangle, parameters of points in the triangle and along its edges can be determined from the barycentric coordinates [5, pp. 289–291]. Parameters of a point in a triangular patch are set equal to the parameters of the point closest to it in the associating triangle. Initially, depending on the complexity of a shape, some points in a patch may receive the same parameters. However, as the subdivision proceeds the probability of such cases decreases, producing unique parameters for points in a patch.

3 The Subdivision Algorithm

Using the above definitions, we now describe an algorithm that approximates a digital shape by a triangular mesh with a required accuracy. We start by approximating the shape with an octahedron. Then, we subdivide the triangular faces into smaller triangles until error in the approximation reaches a required tolerance.

Algorithm 1: Subdivision of a digital shape to a triangular mesh

1. **Initialization:** Determine the major axis of the shape and approximate the shape with an octahedron whose major axis lies on the shape's major axis and whose vertices lie on the shape. Then, enter the triangular faces of the obtained octahedron into a list.
2. **Main Step:** Remove a triangle from the list. If the distance between the triangle and the corresponding patch is larger than the required tolerance, subdivide it and enter the newly obtained triangles into the list.
3. **Stopping Criterion:** If the list is empty, stop. Otherwise, go to the Main Step.

The reason for orienting the octahedron so that its major axis lies on the major axis of the shape is to maximize overlap between the shape and the octahedron and, thereby, minimize the distance between the approximating mesh and the shape. When subdivision is complete, the maximum distance between the shape and its approximating mesh is guaranteed to be smaller than the required tolerance.

Some of the properties of this approximation are:

1. A unique subdivision is obtained independent of the orientation or position of a shape. This is achieved by aligning the major axis of the octahedron with the major axis of the shape.
2. The process avoids subdivision into triangles with acute angles or long edges. This is achieved by subdividing the edges of the mesh first.
3. Compression rate depends on the complexity of the shape. During subdivision, large triangles are generated at smooth areas and small triangles are created at detailed areas. The process automatically adjusts triangle sizes to reproduce local details in a shape.

4 Parametrizing the Shape Points

To parametrize the mesh vertices, first, parameters at the octahedral vertices approximating the shape are determined. This is achieved by fitting an octahedron to a sphere, establishing correspondence between vertices in the shape approximation and in the sphere approximation, and assigning parameters of mesh vertices in the sphere approximation to parameters of mesh vertices in the shape approximation. As a triangle in the shape approximation is subdivided, the corresponding triangle in the sphere approximation is subdivided also and, again, parameters at newly obtained mesh vertices in the sphere are assigned to corresponding mesh vertices in the shape.

When an edge contour in the shape approximation is divided into two, in the sphere approximation, the corresponding arc is divided into two in such a way that the proportion of the lengths of newly obtained arcs are the same as the proportion of the lengths of contour segments in the shape. At any stage of the process, by knowing parameters of mesh vertices in the sphere, parameters of corresponding mesh vertices in the shape are known. Therefore, when the subdivision ends, spherical parameters of all mesh vertices approximating a shape will be known. Algorithm 1, therefore, provides the means to determine the parameters at vertices of the triangular mesh approximating a shape. By knowing parameters at three vertices of a triangle, parameters at points in the triangle can be determined using the barycentric coordinates, and by knowing the parameters of points in a triangle, parameters of points in the associating triangular patch can be determined from the correspondence between the two. In this manner, the spherical parameters of all points in a digital shape can be determined.

5 Approximating a Digital Shape by a Rational Gaussian Surface

Knowing the coordinates and the parameters of points in a digital shape, we can find a smooth parametric surface that approximates the shape. We will use the mesh vertices as the control

points of the parametric surface. Since the vertices are irregularly spaced, we will need a surface formulation that does not require a regular grid of control points. A RaG surface [3, 4] can have irregularly spaced control points and, therefore, will be used in this approximation. Assuming vertices of the triangular mesh obtained by Algorithm 1 are $\{\mathbf{P}_i : i = 1, \dots, N\}$ and the parameters associated with them are $\{(u_i, v_i) : i = 1, \dots, N\}$, a RaG surface that approximates the vertices can be written as [3, 4]

$$\mathbf{P}(u, v) = \sum_{i=1}^N \mathbf{P}_i g_i(u, v), \quad (2)$$

where $g_i(u, v)$ is the i th basis function of the surface defined by

$$g_i(u, v) = \frac{G_{\sigma_i}(u - u_i, v - v_i)}{\sum_{j=1}^N G_{\sigma_j}(u - u_j, v - v_j)}, \quad (3)$$

and $G_{\sigma_i}(u - u_i, v - v_i) = \exp\{[(u - u_i)^2 + (v - v_i)^2]/2\sigma_i^2\}$.

The standard deviations of Gaussians will be set in such a way to reproduce local shape details. The sizes of triangles obtained in Algorithm 1 contain information about local details in a shape. We will set the standard deviations of Gaussians proportional to the perimeters of the triangles.

If the surface is required to interpolate the vertices, we let $\mathbf{P}(u_i, v_i) = \mathbf{P}_i$ and compute the control points of the surface, $\{\mathbf{V}_i : i = 1, \dots, N\}$, from three systems of N linear equations:

$$\mathbf{P}_i = \sum_{j=1}^N \mathbf{V}_j g_j(u, v), \quad i = 1, \dots, N. \quad (4)$$

Because of the nature of the rational Gaussian bases, the obtained matrix of coefficients will be diagonally dominant. For very large standard deviations, however, the system will become unstable because it may not be possible to fit a surface with a desired smoothness to fit to points in a very detailed area.

In Algorithm 1, the decision to subdivide a triangle was based initially on distances between edges of the triangle and the associating edge contours, and then on the distance between the triangle and the associating patch. In order to fit a RaG surface to a shape, we redefine the error criteria in Algorithm 1 as follows:

1. Instead of determining the distance between an edge and its corresponding edge contour, we will determine the distance between an edge contour and the approximating/interpolating surface. Since parameters of all points in an edge contour are known, for each point \mathbf{p}_i in an edge contour, we can determine the corresponding point $\mathbf{P}(u_i, v_i)$ in the approximating/interpolating RaG surface. We then let the maximum distance between corresponding points in the contour and the surface be the distance between an edge contour and the RaG surface: $D_E = \max_i \|\mathbf{P}(u_i, v_i) - \mathbf{p}_i\|$.
2. Instead of determining the distance between a triangle and its associating patch, we will determine the distance between the patch and the approximating/interpolating RaG surface. This is possible because, by knowing parameters (u_i, v_i) at each point \mathbf{p}_i in the patch, we can determine the corresponding point $\mathbf{P}(u_i, v_i)$ in the surface. We will then define the distance between a triangular patch and its approximating/interpolating RaG surface to be the maximum distance between corresponding points in the patch and the surface: $D_T = \max_i \|\mathbf{P}(u_i, v_i) - \mathbf{p}_i\|$.

Replacing error measures D_e and D_t in Algorithm 1 with error measures D_E and D_T , respectively, we will obtain an algorithm that fits a RaG surface to a digital shape, ensuring that distance between the given shape and the approximating/interpolating surface is within the required tolerance. We will call this new algorithm, **Algorithm 2**.

Note that σ 's in the RaG formulation are in the same units as u 's and v 's. As the subdivision progresses, the sizes of triangles in the sphere subdivision become smaller. Roughly, the perimeter of a triangle reduces to half its size in each subdivision. Therefore, the σ to be assigned at a particular subdivision level will depend on the depth of the subdivision.

Suppose at level 0 an octahedron is fitted to the shape and another octahedron is fitted to a sphere. If we were to stop the approximation at level 0, we would set all σ 's to 1 to obtain a smooth approximation to the shape. As the σ 's are reduced, the obtained surface will resemble the approximating triangular mesh, and as the σ 's are increased the surface approaches a sphere. In the case of interpolation, for very large values of σ 's it may not be possible to obtain a surface that would pass through the points. Typically, proper values for the σ 's at level 0 are between 0.2 and 2.

At level 1, perimeters of triangles in the sphere are roughly half the perimeters of triangles at level 0. Therefore, σ 's at level 1 should be half the σ 's at level 0. Analogously, σ 's at level n should be 2^{-n} of σ 's at level 0. Denoting the σ assigned to vertices at the i th level by s_i , we will have $s_i = 2^{-i}s_0$. In this manner, σ at all vertices will depend on a single parameter s_0 . By adjusting this single parameter, the overall smoothness of the reconstructed surface can be controlled. At one extreme when s_0 is close to zero, the surface approaches the approximating triangular mesh. At the other extreme, when s_0 is very large, the shape will approach a sphere, and if an interpolating surface is required, beyond a certain point it may not be possible to obtain a surface that would fit the mesh vertices.

Note that since σ 's are associated with the control points of a RaG surface, and the control points are the vertices of the approximating triangular mesh, whenever an edge is divided into two, σ 's associated with the mesh vertices corresponding to the end points of the edge are also divided by two. A vertex may be shared by many triangles obtained at different levels. Assigning a σ to the vertex that is proportional to the triangle at the highest level will enable reproduction of details differently at different sides of a point. This can be explained by examining the spatial frequency characteristics of Gaussians.

From the signal processing point of view, as the standard deviation of Gaussians in the spatial domain increases (decreases), the Fourier transform of the Gaussians, which are also Gaussians, will become narrower (wider) in the frequency domain. This means, in areas where smaller σ 's are used, the obtained surface can reproduce high and low spatial frequencies in the surface, and in areas where only large σ 's are used, high spatial frequencies are not reproduced, thus creating a smooth surface. Narrower Gaussians enable reproduction of both low and high spatial frequencies. For any value of s_0 , relative details obtained in different areas in a reconstructed surface will depend on relative details of local areas in the original shape.

Also note that although some triangles at level n could have the same size as some triangles at levels $m < n$ in the xyz space, in the uv space the sizes of triangles at level n are smaller than those at level m . By appropriately reducing the σ 's at higher levels, we are in effect preserving information about the sizes of triangles at different levels. When a small portion of a sphere is mapped to a large portion of a shape, the σ 's assigned to different subdivision levels enable proper reproduction of details in the shape.

6 Examples

The digital shape depicted in Fig. 1a shows a femoral stem obtained by segmenting a volumetric X-ray Computer Tomography (CT) image. There were holes at the top and bottom of the original bone. The holes were covered with planar patches to obtain a closed shape. The process of subdividing the closed shape into triangles using Algorithm 1 with error tolerance of 0.5 units is shown in Figs. 1b–1e. It is assumed that the length of each side of elements (voxels) in the shape is 1 unit. Figure 1f shows the rendered femoral stem using the obtained triangular mesh. The triangulation process has placed larger triangles in smoother areas and smaller triangles in more detailed areas.

Since subdivision in the shape and in the sphere are performed in parallel, at any stage of the process, parameters at the vertices of the triangular mesh are known from the parameters of corresponding points in the sphere. Using Algorithm 2 with the same number of control points as the number of vertices obtained in the mesh approximation, we obtain Figs. 2a–2c when s_0 is set to 0.2, 0.5, and 1, respectively.

A second set of examples is shown in Figs. 3a–3c. The data set used in this experiment was obtained by segmenting a Magnetic Resonance (MR) image of a person’s head. The segmentation has extracted the skin of the head. Using Algorithm 2 with error tolerance equal to 0.5, we obtain the surface shown in Fig. 3a with $s_0 = 0.2$. Increasing s_0 to 0.5 we obtain the surface shown in Fig. 3b. Increasing s_0 further to 1, we obtain the surface shown in Fig. 3c.

Figures 2 and 3 show two examples of the proposed surface-fitting method. The number of control points obtained in an approximation is not a mere function of the error tolerance; it is also a function of the smoothness of the required surface. Preliminary results show that to achieve a high compression rate, when a large error tolerance is given a large s_0 should be used, and when a small error tolerance is given a small s_0 should be used.

7 Concluding Remarks

An algorithm to subdivide a digital shape into a triangular mesh, parametrize the mesh vertices as well as the shape points, and fit a rational Gaussian surface to the points was presented. Attempts to parametrize mesh vertices have been made before. Lee *et al.* [6] simplified a mesh to a base mesh, assigned parameters to the vertices of the base mesh, and determined parameters at the original mesh vertices through conformal mapping of the base mesh to the original mesh. Rogers and Fog [7] developed a nonlinear optimization method for determining the parameters of a mesh to be approximated by B-spline patches. Brechbühler *et al.* [1] developed an optimization method for mapping vertices of a simple polyhedron into a sphere and thereby parametrizing the polyhedral vertices.

Once the shape points or the mesh vertices are parametrized, a single RaG surface can be fitted to the points to reconstruct the shape. A RaG surface enables editing of a shape by moving its control points just like a NURBS surface. This representation is especially useful when a noisy data set is given and there is a need to smooth noise in the data. The RaG formulation has a smoothness parameter that can be varied to obtain surfaces at different levels of detail.

References

- [1] Ch. Brechbühler, G. Gerig, and O. Kübler, Parametrization of closed surfaces for 3-D shape description, *Computer Vision and Image Understanding*, vol. 61, no. 2 (1995), 154–170.
- [2] J. M. Galvez and M. Canton, Normalization and shape recognition of three-dimensional objects by 3-D moments, *Pattern Recognition*, vol. 26, no. 5 (1993), 667–682.
- [3] A. Goshtasby, Design and recovery of 2-D and 3-D shapes using rational Gaussian curves and surfaces, *Int'l J. Computer Vision*, vol. 10, no. 3 (1993), 233–256.
- [4] A. Goshtasby, Geometric modeling using rational Gaussian curves and surfaces, *Computer Aided Design*, vol. 27, no. 5 (1995), 363–375.
- [5] J. Hoschek and D. Lasser, *Computer Aided Geometric Design*, A. K. Peters, 1989.
- [6] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin, MAPS: Multiresolution adaptive parametrization of surfaces, *Computer Graphics Proceedings* (1998), 95–104.
- [7] D. F. Rogers and N. G. Fog, Constrained B-spline curve and surface fitting, *Computer Aided Geometric Design*, vol. 21 (1989), 641–648.

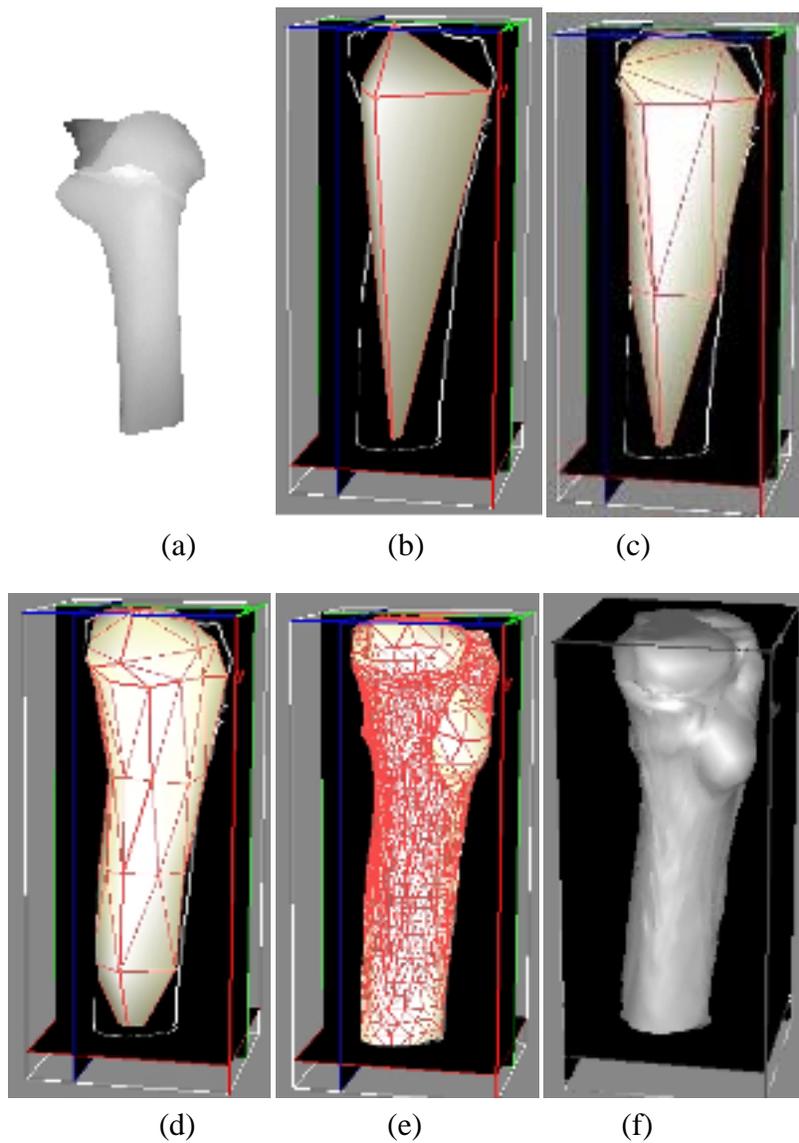


Fig. 1. Approximation of a digital shape by a triangular mesh

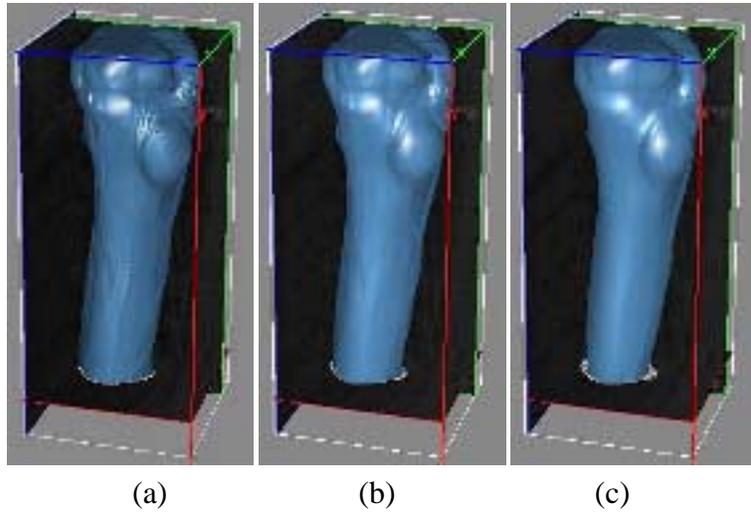


Fig. 2. Approximating a digital femoral stem by RaG surfaces

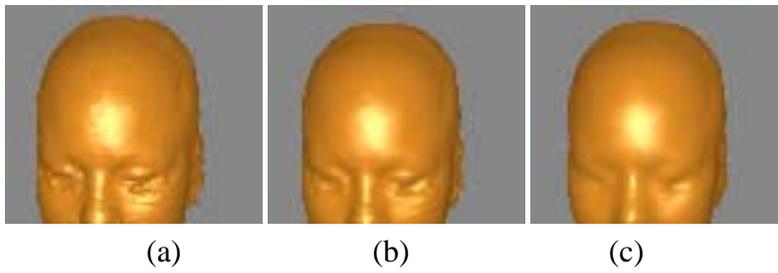


Fig. 3. Approximating a digital head by RaG surfaces