17. Parametric surfaces

Reading

Required:

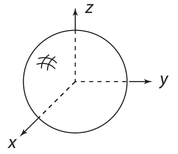
- Watt, 2.1.4, 3.4-3.5.

Optional

- Watt, 3.6.
- Bartels, Beatty, and Barsky. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling,* 1987.

Mathematical surface representations

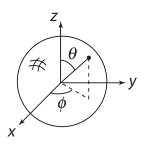
- Explicit $z=f(x,y)$ (a.k.a., a "height field")
  - what if the curve isn't a function, like a sphere?



- Implicit $g(x,y,z) = 0$

- Parametric $(x(u,v), y(u,v), z(u,v))$
  - For the sphere:

$x(u,v) = r \cos 2\pi v \sin \pi u$
$y(u,v) = r \sin 2\pi v \sin \pi u$
$z(u,v) = r \cos \pi u$



As with curves, we'll focus on parametric surfaces.

Surfaces of revolution

Idea: rotate a 2D **profile curve** around an axis.

What kinds of shapes can you model this way?

Constructing surfaces of revolution

Given: A curve **C**(*u*) in the *xy*-plane:

$$\mathbf{C}(u) = \begin{bmatrix} c_x(u) \\ c_y(u) \\ 0 \\ 1 \end{bmatrix}$$

Let $R_x(\theta)$ be a rotation about the *x*-axis.

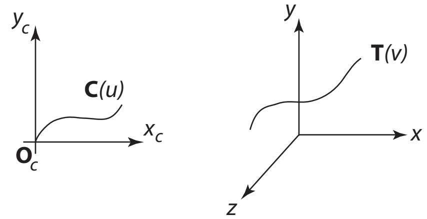
Find: A surface **S**(*u*,*v*) which is **C**(*u*) rotated about the *x*-axis.

Solution:

General sweep surfaces

The **surface of revolution** is a special case of a **swept surface**.

Idea: Trace out surface **S**(*u*,*v*) by moving a **profile curve C**(*u*) along a **trajectory curve T**(*v*).



More specifically:

- Suppose that **C**(*u*) lies in an (x_c,y_c) coordinate system with origin $\mathbf{O}_c$.
- For every point along **T**(*v*), lay **C**(*u*) so that $\mathbf{O}_c$ coincides with **T**(*v*).

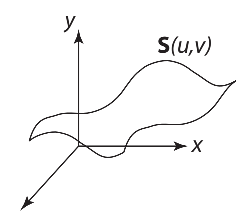
Orientation

The big issue:

- How to orient **C**(*u*) as it moves along **T**(*v*)?

Here are two options:

1. **Fixed** (or **static**): Just translate $\mathbf{O}_c$ along **T**(*v*).

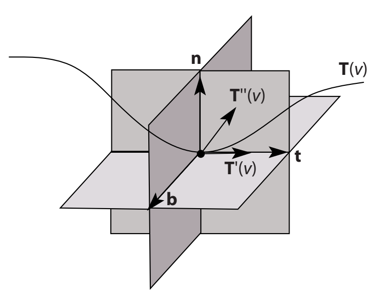


2. Moving. Use the **Frenet frame** of **T**(*v*).

- Allows smoothly varying orientation.
- Permits surfaces of revolution, for example.

Frenet frames

Motivation: Given a curve **T**(*v*), we want to attach a smoothly varying coordinate system.



To get a 3D coordinate system, we need 3 independent direction vectors.

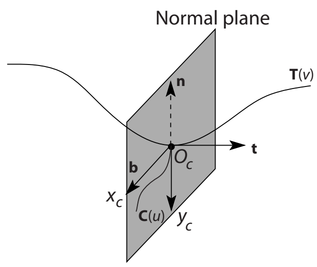
$$\mathbf{t}(v) = \text{normalize}[\mathbf{T}'(v)]$$
$$\mathbf{b}(v) = \text{normalize}[\mathbf{T}'(v) \times \mathbf{T}''(v)]$$
$$\mathbf{n}(v) = \mathbf{b}(v) \times \mathbf{t}(v)$$

As we move along *T*(*v*), the Frenet frame (*t*,*b*,*n*) varies smoothly.

Frenet swept surfaces

Orient the profile curve **C**(*u*) using the Frenet frame of the trajectory **T**(*v*):

- Put **C**(*u*) in the **normal plane** .
- Place $\mathbf{O}_c$ on **T**(*v*).
- Align x_c for **C**(*u*) with **b**.
- Align y_c for **C**(*u*) with -**n**.



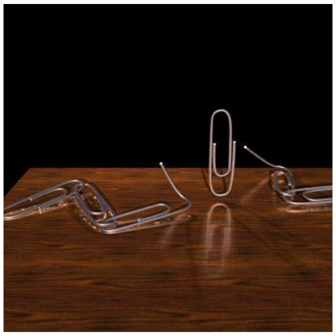
If *T*(*v*) is a circle, you get a surface of revolution exactly!

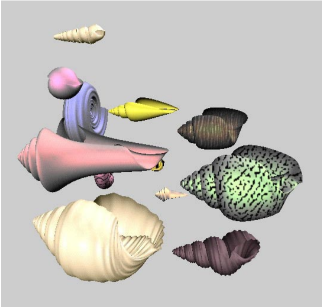
What happens at inflection points, I.e., where curvature goes to zero?

Variations

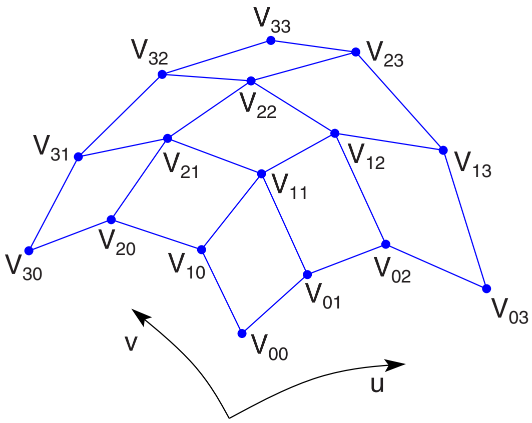
Several variations are possible:

- Scale **C**(*u*) as it moves, possibly using length of **T**(*v*) as a scale factor.
- Morph **C**(*u*) into some other curve $\tilde{\mathbf{C}}(u)$ as it moves along **T**(*v*).
- …





Tensor product Bézier surfaces

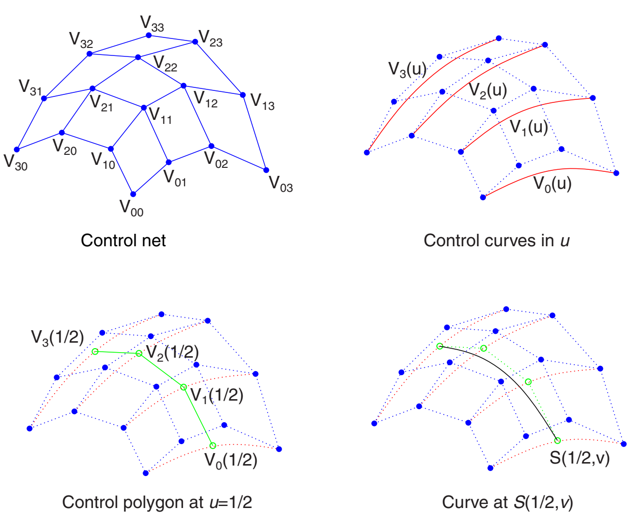


Given a grid of control points $\mathbf{V}_{ij}$, forming a **control net**, contruct a surface **S**(*u*,*v*) by:

- treating rows of **V** (the matrix consisting of the $\mathbf{V}_{ij}$) as control points for curves $\mathbf{V}_0(u),\ldots,\mathbf{V}_n(u)$.
- treating $\mathbf{V}_0(u),\ldots,\mathbf{V}_n(u)$ as control points for a curve parameterized by *v*.

Tensor product Bézier surfaces, cont.

Let's walk through the steps:



Which control points are interpolated by the surface?

Matrix form of Bézier surfaces

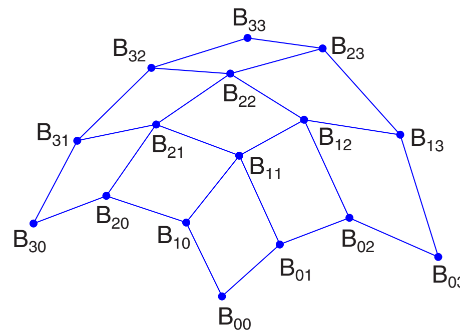
Tensor product surfaces can be written out explicitly:

$$\mathbf{S}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{n} \mathbf{V}_{ij} B_i^n(u) B_j^n(v)$$

$$= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} M_{Bézier} \mathbf{V} \; M_{Bézier}^T \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

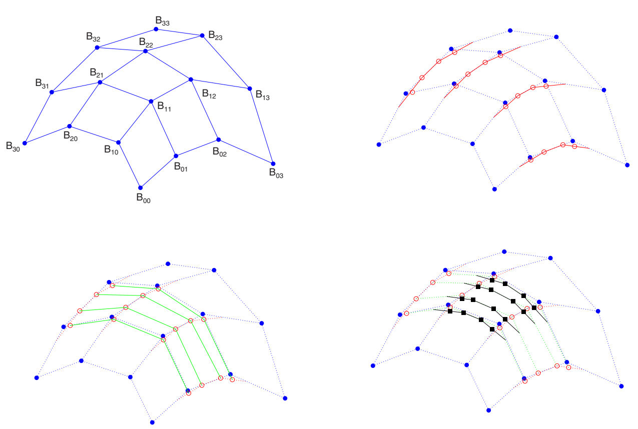
Tensor product B-spline surfaces

As with spline curves, we can piece together a sequence of Bézier surfaces to make a spline surface. If we enforce C^2 continuity and local control, we get B-spline curves:



- treat rows of *B* as control points to generate Bézier control points in *u*.
- treat Bézier control points in *u* as B-spline control points in *v*.
- treat B-spline control points in *v* to generate Bézier control points in *u*.

Tensor product B-spline surfaces, cont.



Which B-spline control points are interpolated by the surface?

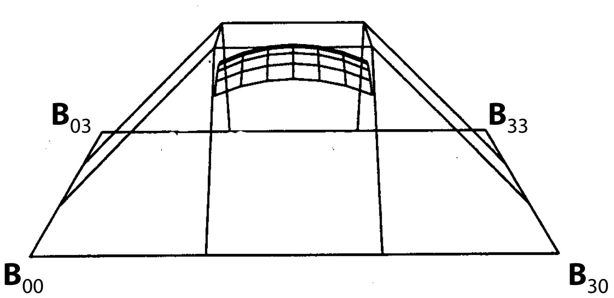
Matrix form of B-spline surfaces

Tensor product B-spline surfaces can be written out explicitly:

$$\mathbf{S}(u,v) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} M_{Bézier} M_{B\text{-}spline} \mathbf{B} \; M_{B\text{-}spline}^T M_{Bézier}^T \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

Tensor product B-splines, cont.

Another example:

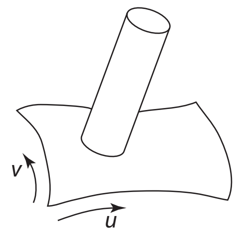


Trimmed NURBS surfaces

Uniform B-spline surfaces are a special case of NURBS surfaces.

Sometimes, we want to have control over which parts of a NURBS surface get drawn.

For example:



We can do this by **trimming** the *u*-*v* domain.

- Define a closed curve in the *u*-*v* domain (a **trim curve**)
- Do not draw the surface points inside of this curve.

It's really hard to maintain continuity in these regions, especially while animating.

Summary

What to take home:

- How to construct swept surfaces from a profile and trajectory curve:
  - with a fixed frame
  - with a Frenet frame
- How to construct tensor product Bézier surfaces
- How to construct tensor product B-spline surfaces