parametric surface patches
implicit representation

• implicit surface representation

\[ f(P) = 0 \]

• e.g. sphere: \( f(P) = 0 \Rightarrow x^2 + y^2 + z^2 - r^2 = 0 \)
parametric representation

• parametric surface representation

\[ \mathbf{P}(u) = ( f_x(u), f_y(u), f_z(u) ) \]

• e.g. sphere: \( \mathbf{P}(u) = ( r\cos \phi \sin \theta, r\sin \phi \sin \theta, r\cos \theta ) \)
parametric representation

- goals when defining $f$
  - smoothness, efficiency, local control
  - same as curves
- (1) combine curves to obtain surfaces
  - easy for simple solids, but not general
- (2) extend parametric curves to surface patches
  - general formulation
  - used heavily in CAD
surfaces from curves

- example: extrude curve $\mathbf{C}(u)$ in XY-plane along Z-axis
  \[ \mathbf{P}(u, v) = (C_x(u), C_y(u), v) \]

- example: revolve curve $\mathbf{C}(u)$ in YZ-plane about Z-axis
  \[ \mathbf{P}(u, v) = (C_y(u) \cos(v), C_y(u) \sin(v), C_z(u)) \]
patches
patches

• spline curves: 1D blending functions

\[ P(t) = \sum_i b_i(t)P_i \]

• surface patches: 2D blending functions
  ○ cross product of 1D blending functions

\[ P(u, v) = \sum_{ij} b_{ij}(u, v)P_{ij} \]

\[ b_{ij}(u, v) = b_i(u)b_j(v) \]
patches

- bicubic Bezier patches
patches

- joining is hard
other patch functions

- just like curves
  - uniform, non-uniform
- non-uniform rational B-splines (NURBS)
  - ratios of B-splines
  - invariance under perspective
  - can represent conic sections exactly
  - often used in 3D
rendering parametric surfaces
rendering parametric surfaces

- tessellation: approximate surfaces with triangles/quads
  - meshes are efficient to draw in hardware and software
  - more faces to provide better approximation
- uniform tessellation: split \((u, v)\) intervals uniformly
  - fast to compute and simple to implement
  - generates many segments
- adaptive tessellation: split recursively until good enough
  - Bezier patches can use De Casteljau
uniform tesselation

• split in $K \times K$ points uniformly at
  $$(u_{k_1}, v_{k_2}) = (1/k_1, 1/k_2)$$
  ◦ tesselate into quads by creating a $K \times K$ quad grid
  ◦ set vertices to $P(u_{k_1}, v_{k_2})$
  ◦ join vertices to avoid holes if function wraps (e.g. cylinder)

• normals
  ◦ evaluate analytically if possible (e.g. sphere)
  ◦ or evaluate by computing partial derivatives
    $$\mathbf{n} = \partial P / \partial u \times \partial P / \partial v$$
  ◦ or smooth mesh by averaging face normals over vertices