Approximate implicitization in Computer Aided Geometric Design

Jan B. Thomassen

SINTEF ICT

Nice, June 2, 2006

Work done with: Tor Dokken and Vibeke Skytt

Supported by: Aim@Shape (EU Network of Excellence)

...and, before that: GAIA II (EU project – André)

Background

- Representation of curves and surfaces in CAD-systems
- The intersection problem
- Recursive subdivision

Approximate implicitization

- Implicitization as a linear algebra problem
- Convergence rates

Examples

- Separation of surfaces
- Simple case: Cylinder-plane test
- Simple case: Monotonicity

Summary

Representation of curves and surfaces in CAD-systems

Spline curves:

$$\mathbf{p}(t) = \sum_{i=0}^{n} \mathbf{c}_i N_{i,n}(t)$$

Spline surfaces:

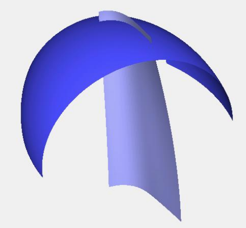
$$\mathbf{p}(u, v) = \sum_{i=0}^{n_1} \sum_{j=0}^{n_2} \mathbf{c}_{ij} N_{i,n_1}(u) N_{j,n_2}(v)$$

- Parametric
- Piecewise polynomial
- B-spline basis: $N_{i,n}(t)$

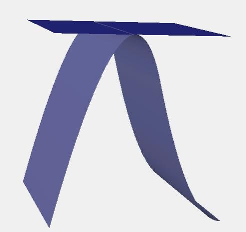
The intersection problem

Given: Two surfaces $\mathbf{p}_1(u_1, v_1)$ and $\mathbf{p}_2(u_2, v_2)$, and a numerical tolerance ϵ.

Find: All parameters (u_1, v_1, u_2, v_2) such that $|\mathbf{p}_1(u_1, v_1) - \mathbf{p}_2(u_2, v_2)| < \epsilon$.



Example: Transversal intersection (easy)



Example: Near tangential intersection (hard)

Recursive subdivision

Algorithm:

- Input: Two surfaces, a tolerance
  - Can we rule out intersection?
    - Yes $\Rightarrow$ OK/Stop
    - No $\Rightarrow$ Continue
  - Do we have simple case
    - Yes $\Rightarrow$ OK/Stop
    - No $\Rightarrow$ Continue
  - Subdivide and proceed with each subproblem
- Output: Topology of intersection curves

Implicit representations can help in *both* ruling out intersections and detecting simple cases.

Implicitization as a linear algebra problem

We have a parametric surface of bi-degree (n_1, n_2):

$$\mathbf{p}(u,v) = \sum_{ij} \mathbf{c}_{ij} B_{i,n_1}(u) B_{j,n_2}(v)$$

We want an implicit surface of total degree d:

$$q(\mathbf{x}) = \sum_{ijkl} b_{ijkl} B_{ijkl,d}(\mathbf{x}) = 0$$

Composition gives the equation:

$$q(\mathbf{p}(u,v)) \quad = \quad \mathbf{B}^T(u,v)\mathbf{D}\mathbf{b} \; = \; 0$$

Bernstein basis of degree n on an interval, $u \in [0,1]$:

$$B_{i,n}(u) \quad \equiv \quad \binom{n}{i} u^i (1-u)^{n-i}$$

Bernstein basis of total degree d on a tetrahedron, $\mathbf{x} = (u,v,w,z)$, $u+v+w+z=1$:

$$B_{ijkl,d}(\mathbf{x}) \quad \equiv \quad \frac{d!}{i!j!k!l!} u^i v^j w^k z^l$$

- Partitions of unity: $\sum_i B_{i,n} = 1$, $\sum_{ijkl} B_{ijkl,d} = 1$

Solving $\mathbf{Db} = 0$, $\mathbf{b} \neq 0$

The matrix equation $\mathbf{Db} = 0$, $\mathbf{b} \neq 0$, can be solved by SVD of $\mathbf{D}$,

$$\mathbf{D} = \mathbf{U\Sigma V}^T, \qquad \mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_N \\ & & \mathbf{0} & \end{pmatrix}$$

We choose $\mathbf{b} = \mathbf{v}_N$, where $V = (\mathbf{v}_1, \cdots, \mathbf{v}_N)$

- Exact implicitization: $\sigma_N = 0$, and $\mathbf{Db} = 0$
- Approximate implicitization: σ_N is "small", and $|q(\mathbf{p}(u, v))| \leq \sigma_N$

Convergence rates

A function $g(t)$ approximates a function $f(t)$ on $[a, b]$ with convergence rate k if $|f(t) - g(t)| \leq Ch^k$, where C is a constant and $h = b - a$.

The convergence rates of approximate implicitization, $|q(\mathbf{p}(u, v))| \leq Ch^k$, depends on *choice of basis*, and *choice of degree*.

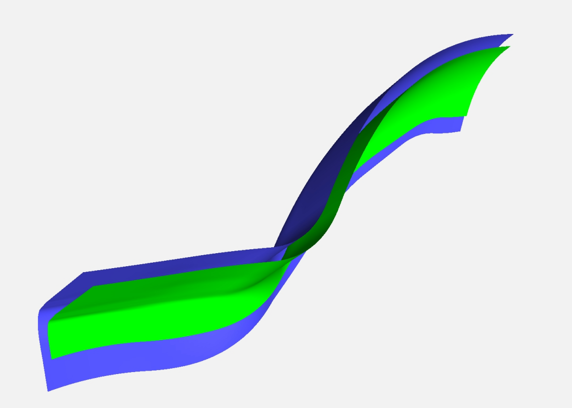
Algebraic degree	1	2	3	4	5	6
Convergence rate	2	5	9	14	20	27

Table: Convergence rates for curves in 2D

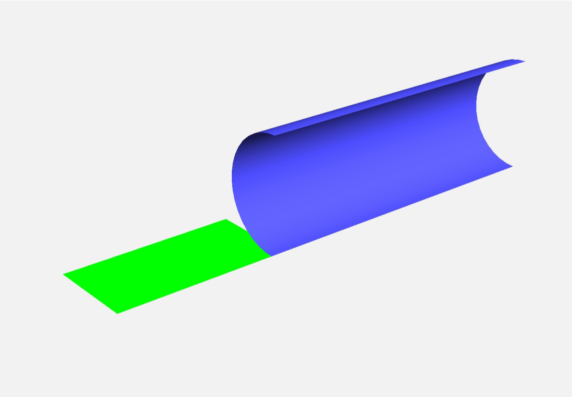
Algebraic degree	1	2	3	4	5	6
Convergence rate	2	3	5	7	10	12

Table: Convergence rates for surfaces in 3D

Separation of surfaces

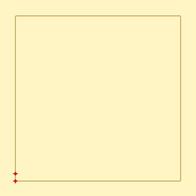


Simple case: Cylinder-plane test

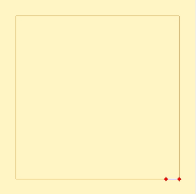


Parameter planes:

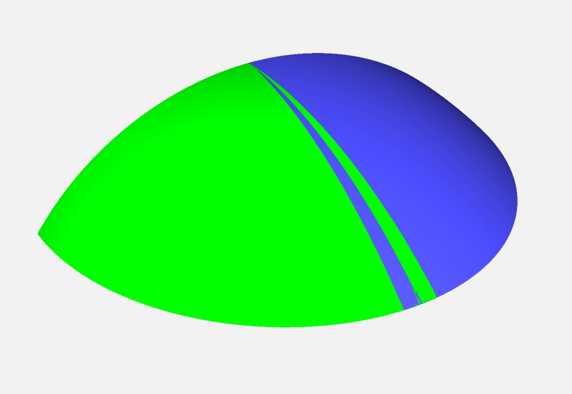
Half-cylinder:



Plane:

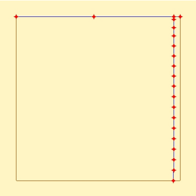


Simple case: Monotonicity

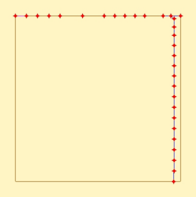


Parameter planes:

Sphere 1:



Sphere 2:



Summary

- In CAGD, surface-surface intersection problems are difficult
- Approximate implicitization is a mathematically and numerically well-defined procedure
- Implicitization is very useful in recursive subdivision algorithms for finding intersections