Constructing Smooth Branching Surfaces from Cross Sections

P.D. Kaklis
joint work with N.C. Gabrielides and A.-A.I Ginnis

NATIONAL TECHNICAL UNIVERSITY OF ATHENS (NTUA)
School of Naval Architecture & Marine Engineering (sNAME)
contents

- the problem statement
- the *one-to-one* Problem
- the *one-to-many* Problem
- the *many-to-many* problem
the problem statement

Construct a $G^1$-continuous surface that interpolates ordered point sets belonging to the contours of sections of an object with parallel planes.
The Two instances of the problem

Reconstruction

Input Data
• The cardinality is large

Solution
• $C^0$ triangular surfaces
• fast Algorithms
• much work done on multiple contours per section
The Two instances of the problem

Design

Input Data

• The cardinality is small

Solution

• smooth surfaces consisting of four sided patches

• not so fast algorithms in comparison to the $C^0$ approach

• lack of algorithms permitting more than one contour per section
decomposing the problem – step A

A. The correspondence question: which contours of a plane are connected to which of the contours of the neighboring planes.

- The solution can be represented by a graph
- In the design context the correspondence graph is usually given
decomposing the problem – step B

The correspondence between two neighboring planes can be represented by a local subgraph of the following type:

- **edge**: “one-to-one”
- **tree**: “one-to-many”
- **graph**: “many-to-many”

B. The local surface construction problem: construct a smooth, $G^1$, surface, which:

- interpolates the contours described by the local sub-graph
- satisfies appropriate boundary conditions so that the union of all local solutions is $G^1$-continuous.
contents

- the problem statement
- the *one-to-one* Problem
- the *one-to-many* Problem
- the *many-to-many* problem
one-to-one problem: formulation

Construct a $G^1$-continuous surface interpolating the data sets (given data points and pre-estimated cross-tangent vectors on them) on $i$- and $(i+1)$-planes.
one-to-one problem
proposed solution

**Step A:** Parameterize the closed point sets on each plane
closed point sets parameterization

convex case:
Adopt method proposed in [Marsan & Dutta ’99]
▪ polar angle parameterization,
▪ aiming to minimize twist

non-convex case:
▪ transform the polygon onto its convex hull [Ekoule et al. ’91]
▪ adopt the parameterization of the transformed point set.
one-to-one problem
proposed solution

**Step B:** Construct the curves $C_k(u)$ and $T_k(u)$, $k=l, l+1$, using a shape-preserving interpolation technique [Kaklis & Karavelas ’97]
one-to-one problem
proposed solution

**Step B:** Construct the curves $C_k(u)$ and $T_k(u)$, $k=i,i+1$, using a shape-preserving interpolation technique [Kaklis & Karavelas '97]
Step C: Construct the tangent-vector ribbons $t_K(u)$

$C_K(u)$

$t_K(u) = T_K(u) - C_K(u)$

$T_K(u)$
one-to-one problem
proposed solution

**Step D:** Apply the skinning technique to obtain the final surface
contents

- the problem statement
- the one-to-one Problem
- the one-to-many Problem
- the many-to-many problem
one-to-many problem: formulation

Construct a $G^1$-continuous surface interpolating the data sets (given data points and cross-tangent vectors on them) on $i$- and $(i+1)$ - planes.
assumption on the polygonal contours

Each contour should contribute to their convex hull

(+ more conditions)
one-to-many problem: proposed solution

**Step A:** Construct the contours and the cross tangent distributions on them.
one-to-many problem: proposed solution

**Step A:** Construct the contours and the cross tangent distribution on them.
one-to-many problem:
proposed solution

Step A: Construct the contours and the cross tangent distribution on them.
one-to-many problem: proposed solution

**Step B:** Construct the surrounding curve and the associated tangent ribbon on the “many” plane.
one-to-many problem: proposed solution

**Step B:** Construct the surrounding curve and the associated tangent ribbon on the “many” plane.
one-to-many problem: proposed solution

**Step C:** Construct the surrounding skinning surface.
one-to-many problem: proposed solution

Step D: Construct appropriate trimming curves on the surrounding surface and ...
one-to-many problem: proposed solution

**Step D:** trim, leaving a 2xm sided hole to be filled.

(m is the number of contours on the “many” plane)
one-to-many problem: proposed solution

Step E: Hole filling with rectangular surface patches using an enhanced version of [Hahn, 89] method.

E1: Construct the “guide” curves.
one-to-many problem: proposed solution

Step E: Hole filling with rectangular surface patches.

E1: Construct the “guide” curves.
one-to-many problem: proposed solution

**Step E:** Hole filling with rectangular surface patches.

**E2:** Hole filling with Gordon-Coons patches.
one-to-many problem: proposed solution

Step E: Hole filling with rectangular surface patches.

E2: Hole filling with Gordon-Coons patches
one-to-many problem: proposed solution

**Step E:** Hole filling with rectangular surface patches.

**E2:** Hole filling with Gordon-Coons patches
contents

- the problem statement
- the one-to-one Problem
- the one-to-many Problem
- the many-to-many problem
many-to-many problem: formulation

Construct a $G^1$-continuous surface interpolating the data sets (given data points and cross-tangent vectors on them) on $i$- and $(i+1)$ - planes
many-to-many problem proposed solution

**Step A:** Construct the contours and the cross tangent distribution on them
many-to-many problem proposed solution

**Step B:** Construct the surrounding curves and the cross tangent distribution on them on both planes.
many-to-many problem proposed solution

**Step C:** Construct the surrounding (skinning) surface.
many-to-many problem proposed solution

**Step D:** Define a separating zone between the i and (i+1)-plane on the surrounding surface and obtain two *one-to-many* problems.
many-to-many problem proposed solution

**Step D:** Define a separating zone between the $i$ and $(i+1)$-plane on the surrounding surface and obtain two one-to-many problems.
many-to-many problem proposed solution

*Step E:* Construct the proposed solution for the *one-to-many* problems
many-to-many problem proposed solution

**Step E:** Construct the proposed solution for the *one-to-many* problems
many-to-many problem proposed solution

**Step E:** Construct the proposed solution for the *one-to-many* problems
current work

• modify the CH based method for constructing the surrounding curve so that it touches all contour curves

• improve the surface quality by optimizing characteristic parameters of the method, e.g. hole center, versus geometrically / physically based criteria
Smooth Branching Surfaces from Contours: Open Problems

P. Kaklis
joint work with M. Karavelas

National Technical University of Athens (NTUA)
School of Naval Architecture and Marine Engineering (Sname)
University of Crete
Department of Applied Mathematics

October 7, 2010
surrounding-curve construction: failure configurations

topological instability
point-sets that touch the convex hull may, under a small perturbation, seize to touch the convex hull and thus fail to participate in the construction of the surrounding curve.

ignoring point-sets near the convex hull
point-sets near, but not on, the convex hull cannot participate in the surrounding curve and thus cannot be taken into account when constructing the surrounding surface and eventually the final interpolatory surface.
remedy deficiencies with the aid of the Voronoi diagram

One way to, at least partially, remedy this deficiency is to **extend the surrounding curve definition by including point-sets that are near the convex hull**. There seems to be no unique notion of “nearness”, and in what follows we shall describe two ways of measuring proximity.

**limitations**

Our discussion below is limited to convex contour curves

In the surrounding curve context, the objects in the set $S$ are the convex objects bounded by the contour curves $C_{i+1,j}$. The **Voronoi diagram** $\mathcal{V}(S)$, or equivalently its dual **Delaunay graph** $\mathcal{D}(S)$, captures a lot of proximity information, which we exploit in order to extend the definition of surrounding curve.
Let us denote by $S_0$ the set of objects in $S$ that contribute to their convex hull $H(S)$.

Let $S_1$ be the set of objects in $S \setminus S_0$ that belong to a triangle in $\mathcal{D}(S)$, the other two vertices of which belong to $S_0$. In other words, we focus on non-convex-hull objects that belong to triangles adjacent to convex hull edges in $\mathcal{D}(S)$.

If the objects in $S_0$ can be thought of as the 0-level objects when approaching $S$ from infinity, $S_1$ are the next level (1-level) objects that we encounter after the objects in $S_0$. 
2 angles for measuring proximity

1. Let $S_\lambda$ be some object in $S_1$ and let $T$ be a triangle in $D(S)$ that connects $S_\lambda$ with two objects $S_\mu$ and $S_\nu$ in $S_0$.

2. Let $C_T$ be the Voronoi circle corresponding to $T$, and let $\Delta_{\lambda\mu\nu}$ denote the triangle defined by the three points of tangency of $C_T$ with the above objects, denoted as $\Sigma_\lambda$, $\Sigma_\mu$ and $\Sigma_\nu$.

1st criterion

Let $\alpha_{\lambda,\mu\nu}$ be the angle of $\Delta_{\lambda\mu\nu}$ at $\Sigma_\lambda$, and $K_{\lambda,\mu\nu}$ the corresponding cone, the apex of which is $\Sigma_\lambda$. 
Let $R_{\lambda\mu}$ and $R_{\lambda\nu}$ be the rays emanating from $\Sigma_\lambda$, belonging to $K_{\lambda,\mu\nu}$, that are tangent to $S_\mu$ and $S_\nu$. We shall denote by $\tilde{K}_{\lambda,\mu\nu}$ the cone defined by $R_{\lambda\mu}$ and $R_{\lambda\nu}$.

2nd criterion

Let $\tilde{\alpha}_{\lambda,\mu\nu}$ the angle of $\tilde{K}_{\lambda,\mu\nu}$ at its apex $\Sigma_\lambda$. Note that $\tilde{K}_{\lambda,\mu\nu}$ is essentially the region of space visible from $S_\lambda$ through $S_\mu$ and $S_\mu$.

The two angles $\alpha_{\lambda,\mu\nu}$ and $\tilde{\alpha}_{\lambda,\mu\nu}$ can be used to measure the proximity of $S_\lambda$ to $H(S)$. Note that as $S_\lambda$ approaches $H(S)$, both $\alpha_{\lambda,\mu\nu}$ and $\tilde{\alpha}_{\lambda,\mu\nu}$ tend to $\pi$. 
In view of this property we use these angles in order to decide whether or not an object in $S_1$ is to be added to the set of objects defining the surrounding curve.

The decision can be made via threshold values, which should be considered as design parameters. The designer can choose a value in the interval $[0, \pi]$ as the minimum acceptable value for either $\alpha_{\lambda,\mu\nu}$ or $\tilde{\alpha}_{\lambda,\mu\nu}$.

Setting this threshold to zero implies that all objects in $S_1$ are to participate in the surrounding curve, whereas choosing the threshold to be equal to $\pi$ essentially reduces to constructing the surrounding curve as in the first part of our presentation.
data_1, triangle criterion: $\geq 90^\circ$
data 1, cone criterion: \( \geq 90^\circ \)
data_2, triangle criterion: $\geq 90^\circ$
data_2, cone criterion: $\geq 90^\circ$
data_1, cone criterion: $\geq 150^\circ$
data_1, cone criterion: $\geq 120^\circ$
data_1, cone criterion: $\geq 90^\circ$
data_1, cone criterion: \( \geq 60^\circ \)
data_1, cone criterion: $\geq 30^\circ$
data_1, cone criterion: $\geq 0^\circ$