TRANSFINITE INTERPOLATION OF SURFACES WITH INTEGRAL CONSTRAINTS USING A DESIGN LANGUAGE

Fausto Bernardini
IBM T. J. Watson Research Center
PO Box 704, Yorktown Heights, NY 10598, U.S.A.
fausto@watson.ibm.com

Glauco Cenciotti & Alberto Paoluzzi
Dip. di Informatica e Automazione, Università Roma Tre
Via della Vasca Navale 79, 00146 Roma, Italy
{cenciott,paoluzzi}@dia.uniroma3.it

Abstract We discuss the problem of geometric design of curved ducts with variable cross-section by transfinite interpolation with integro-differential constraints. This work is a contribution to a research program including numerical simulation and preliminary design of an experimental apparatus for studying fluid-dynamics of air in internal combustion engines of new conception. The proposed geometric modeling approach is based on piecewise multivariate transfinite interpolation of assigned cross-sections, via combination of section-generating functions with univariate quintic Hermite's polynomials. The volume mapping produced by such transfinite interpolation is composed with a local section scaling extracted from a one-parameter family of affine transformations, where the diagonal coefficients depend on the ratio between the areas of the starting and current sections. An approriately chosen point sampling of the duct generated by the composition of volume mapping and section scaling is employed to generate a cell decomposition of the duct volume with tetrahedral elements. Such elements are used for numerical simulation of the fluid-dynamics problem.

Keywords: Solid modeling, transfinite interpolation, integration constraints, geometric programming

Introduction

In this paper we discuss the geometric design of a free-form duct, i.e. a curved duct with variable cross-section. The problem is being studied in order to allow fluid dynamics simulations of the air flow inside curved ducts, with the goal of minimizing pressure losses at the inlet and outlet of internal combustion engine of heavy trucks. Despite the difficulty in establishing an optimum trade-off between the conflicting requirements of aerodynamic efficiency, compactness, and discharge conditions suitable for efficient combustion, very few studies have been conducted on the fundamental physics of the flow evolving in such geometries [Gori, 2000].

In our approach, the duct geometry is generated by continuous transfinite piecewise interpolation of given sections, under suitable differential constraints (continuity of second-order partial derivatives on the surface and/or constant thickness of the duct envelope) as well as under integral constraints (constant area of interpolated cross-sections.) The preliminary choice of the form of the duct will be followed by the numerical simulation with various codes. Such simulation will allow to evaluate and possibly change the shapes of the cross sections which are used as input to the transfinite modeling described here.

Problem statement. Let two smooth parametric surfaces S_0 and S_1 be given, with

$$S_0, S_1 : [0,1] \times [0,1] \rightarrow E^3.$$

Let also two fields N_0 and N_1 of vectors normal to the surfaces S_0 and S_1 be assigned, with:

$$\begin{aligned} N_0 &: [0,1]^2 \rightarrow \Re^3 : N_0(u,v) = h(\partial_u S_0(u,v) \times \partial_v S_0(u,v)), \\ N_1 &: [0,1]^2 \rightarrow \Re^3 : N_1(u,v) = h(\partial_u S_1(u,v) \times \partial_v S_1(u,v)), \end{aligned}$$

with $h \in \Re$. and two fields B_0 and B_1 of vectors normal to the surfaces N_0 and N_1, respectively:

$$\begin{aligned} B_0 &: [0,1]^2 \rightarrow \Re^3 : B_0(u,v) = h(\partial_u N_0(u,v) \times \partial_v N_0(u,v)), \\ B_1 &: [0,1]^2 \rightarrow \Re^3 : B_1(u,v) = h(\partial_u N_1(u,v) \times \partial_v N_1(u,v)). \end{aligned}$$

Our goal is to generate the solid obtained by quintic Hermite's interpolation of surfaces S_0 and S_1 with extreme values of first and second derivatives given by N_0, N_1 and B_0, B_1, respectively. In other words, we want to generate the vector function V, depending on three real parameters $u, v, w \in [0,1]$, defined as:

$$V : [0,1] \times [0,1] \times [0,1] \rightarrow E^3, \qquad \text{such that:}$$

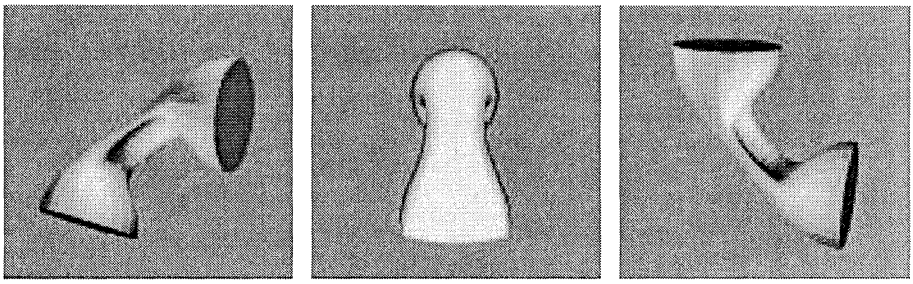

Figure 1. Some views of the unconstrained interpolated duct.

$$
\begin{aligned}
V(u,v,0) &= S_0(u,v), & V(u,v,1) &= S_1(u,v),\\
\partial_w V(u,v,0) &= N_0(u,v), & \partial_w V(u,v,1) &= N_1(u,v),\\
\partial_{ww} V(u,v,0) &= B_0(u,v), & \partial_{ww} V(u,v,1) &= B_1(u,v),
\end{aligned}
$$

under the constraint that the area of given "cross-sections" (for w fixed) be constant and equal to the area of initial section:

$$
\text{Area}(w) = \int_{V(D\times\{w\})} dS = \int_{S_0(D)} dS = \text{Area}(0), \qquad \text{for each } w \in [0,1].
$$

where $D = [0,1] \times [0,1]$. In the following we will refer to the function V above as *volume map*,

Approach. Of course, the more difficult part of the problem is given by the constraint of constant area for each cross-section of the solid generated by the volume map V. The volume map without such constraint would be constructed in a natural way as a quintic Hermite's transfinite blending of maps $S_0(u,v)$, $S_1(u,v)$, $N_0(u,v)$, $N_1(u,v)$, $B_0(u,v)$ and $B_1(u,v)$, according to the approach described in [Paoluzzi, 1999]. The solution described here utilizes methods of transfinite blending, i.e., of interpolation in functional spaces. Such methods are not new, and can be primarily referred to Coons' and Gordon's function interpolations [Coons, 1967; Gordon, 1968].

Our approach has been implemented using the functional design language Plasm [Paoluzzi et al., 1995], developed at the University of Rome "La Sapienza" between the years 89 and 94 [Paoluzzi et al., 1995]. Such language is a geometric extension of a subset of the functional language FL [Backus, 78; Backus et al., 1989; Backus et al., 1990] by John Backus (the FORTRAN inventor) and others, that was developed at Almaden's IBM Research Division in the late eighties. PLaSM is characterized by a multidimensional approach [Ferrucci and Paoluzzi, 1991; Paoluzzi et al., 1993] to geometric data structures and algorithms, so it works with geometric objects of dimension 1,2,3 (curves,

surfaces, solids) as well as with manifolds of higher dimensions. The language defines an algebraic calculus on geometries and allows for implementing parametric representations of multivariate manifolds in a natural and compact way. PLaSM is currently undergoing a re-definition and extension [Cenciotti et al., 1999] in order to allow for strong typing based on classes and objects. A book [Paoluzzi et al., 2001] on the language is being published by J. Wiley.

1. Transfinite Interpolation

Parametric curves and surfaces are usually defined (see e.g. [Bartels et al., 1987]) as vector-valued functions, component-wise generated from some vector space of polynomial (or rational) functions over the field of real numbers. In this section curves, surfaces, and multivariate manifolds are viewed in a unified framework as vector-valued functions generated from the same vector spaces, but over the field of polynomial (or rational) functions itself. This choice implies that the *coefficients* of the linear combination which uniquely represents a curved mapping in a given basis are not real-valued vectors, but function-valued vectors.

This approach is a strong generalization, which contains the standard parametric curves and surfaces as special cases. For example, the standard cubic Hermite interpolation for *curves*, where two extreme points and tangents are interpolated, can be in such extended approach applied to *surfaces*, where two extreme curves of points are interpolated with assigned derivative curves, or even to *volume* interpolation of two assigned surfaces with assigned normal fields. Notice that such an approach is not new, and is quite frequently used in CAD applications, mainly to ship and airplane design, since the times that Gordon-Coons patches were formulated [Coons, 1967; Gordon, 1968]. It is sometime called *function blending* [Gordon, 1968; Lancaster and Salkauskas, 1986], or *transfinite interpolation* [Gordon, 1969; Goldman, 1987].

Parametric maps $\Phi : U \to Y$ used in Computer Graphics and CAD usually belong to the space of rational (i.e. ratio of polynomial) functions of bounded integer degree n. Since the space $\mathcal{Z}_n$ of such functions is a finite-dimensional vector space over the field $\mathcal{Z}_n$ itself, then each $\Phi \in \mathcal{Z}_n$ can be expressed uniquely as a linear combination of $n+1$ basis functions $\phi_i \in \mathcal{Z}_n$ with coordinate functions $\chi_i \in \mathcal{Z}_n$, so that

$$\Phi = \chi_0\phi_0 + \cdots + \chi_n\phi_n.$$

Therefore a unique coordinate representation

$$\Phi = (\chi_0, \ldots, \chi_n)_{\mathcal{B}}$$

of the mapping is given after a basis $\mathcal{B} = \{\phi_0, \ldots, \phi_n\} \subset \mathcal{Z}_n$ has been chosen. The *power* basis, the *cardinal* (or *Lagrange*) basis, the *Hermite* basis, the

Bernstein/Bézier basis and the *B-spline* basis are the most common and useful choices for such a basis.

The coordinate functions χ_i may be easily generated, as will be explained in the following subsections, by using the "geometric handles" of the mapping, usually data points $p_i \in Y$, to be interpolated or approximated by the set $\Phi(U)$.

Only greek letters, either capitals or lower-case, will be used in the sequel to denote functions. Please notice that B and H are also greek upper-case letters for β and η, respectively.

1.1. Univariate case

Let consider the simple univariate case $\Phi : U \subset X \to Y$, where the dimension p of domain X is one. To generate the coordinate functions χ_i it is sufficient to transform each data point $p_i \in Y$ into a constant vector-valued function, so

$$\chi_i = \kappa(p_i), \quad \text{where} \quad \kappa(p_i) : U \to Y : u \mapsto p_i.$$

Using the functional notation with explicit variables, the constant function is such that

$$\kappa(p_i)(u) = p_i$$

for each parameter value $u \in U$.

1.2. Multivariate case

Let consider a multivariate mapping $\Phi : U \to Y$, where $U \subset X$ and X is a p-dimensional space. Since Φ depends on p parameters, in the following will be denoted as Φ^p.

With *transfinite blending* the p-dimensional map Φ^p of degree n is computed by linear combination of $n+1$ *coefficient* maps (each one depending on $p-1$ parameters) with the *univariate* basis of degree n. In other words:

$$\Phi^p = \Phi_0^{p-1}\phi_0 + \cdots \Phi_n^{p-1}\phi_n.$$

The coordinate representation of Φ with respect to the basis $\mathcal{B}^n = (\phi_0, \ldots, \phi_n)$ of degree n is so given by $n+1$ maps depending on $p-1$ parameters:

$$\Phi^p = \left(\Phi_0^{p-1}, \ldots, \Phi_n^{p-1}\right)_{\mathcal{B}^n}$$

and so on, inductively on the dimension of the component maps, until the basic 1-dimensional case is reached.

As an example of transfinite blending consider the generation of a bicubic Bézier surface mapping $B(u_1, u_2)$ as a combination of four Bézier cubic curve

maps $B_k(u_1)$, with $0 \leq k \leq 3$:

$$B(u_1, u_2) = \sum_{k=0}^{3} B_k(u_1)\, \beta_k^3(u_2)$$

where

$$\beta_k^3(u) = \binom{3}{k} u^k (1-u)^{3-k}, \qquad 0 \leq k \leq 3,$$

is the Bernstein/Bézier cubic basis. Analogously, a three-variate Bézier volume mapping $B(u_1, u_2, u_3)$, of degree n_3 on the last parameter, may be generated by univariate Bézier blending of surface maps $B_k(u_1, u_2)$, some of which possibly reduced to a curve map or even to a constant point map:

$$B(u_1, u_2, u_3) = \sum_{k=0}^{n_3} B_k(u_1, u_2)\, \beta_k^{n_3}(u_3)$$

The more interesting aspects of such approach are flexibility and simplicity. Unlike tensor-product method, there is no need that all component geometries have the same degree, or that all are generated using the same function basis. For example, a quintic Bézier surface map may be generated by blending both Bézier curve maps of lower (even zero) degree together with Hermite and Lagrange curve maps. Furthermore, it is much simpler to combine lower dimensional geometries (i.e. maps) than to meaningfully assembly the multi-index tensor of control data (i.e. points and vectors) to generate multivariate manifolds with tensor-product method.

1.3. Quintic Hermite's polynomials

We give here the parametric representation of a quintic Hermite's curve $\mathbf{c}(u)$ which interpolates two extreme points $\mathbf{p}_1$ and $\mathbf{p}_2$, with two extreme tangent vectors $\mathbf{t}_1$ and $\mathbf{t}_2$ and two extreme normal vectors $\mathbf{n}_1$ and $\mathbf{n}_2$:

$$\mathbf{c}(u) = \begin{bmatrix} u^5 & u^4 & u^3 & u^2 & u & 1 \end{bmatrix} \mathbf{H}_5\, \mathbf{g}_5$$

where

$$\mathbf{H}_5 = \begin{bmatrix} -6 & 6 & -3 & -3 & -\frac{1}{2} & \frac{1}{2} \\ 15 & -15 & 8 & 7 & \frac{3}{2} & -1 \\ -10 & 10 & -6 & -4 & -\frac{3}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{g}_5 = \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{t}_1 \\ \mathbf{t}_2 \\ \mathbf{n}_1 \\ \mathbf{n}_2 \end{bmatrix},$$

and where the matrix $\mathbf{H}_5$ can be easily derived from the constraint set:

$$\mathbf{c}(0) = \mathbf{p}_1,\ \mathbf{c}(1) = \mathbf{p}_2,\ \mathbf{c}'(0) = \mathbf{t}_1,\ \mathbf{c}'(1) = \mathbf{t}_2,\ \mathbf{c}''(0) = \mathbf{n}_1,\ \mathbf{c}''(1) = \mathbf{n}_2.$$

Hence the Hermite's basis of quintic polynomials is:

$$\left[\ h_0^5(u)\quad h_1^5(u)\quad \ldots\quad h_5^5(u)\ \right] = \left[\ u^5\quad u^4\quad u^3\quad u^2\quad u\quad 1\ \right]\mathbf{H}_5$$

with

$$\begin{aligned} h_0^5(u) &= -6u^5 + 15u^4 - 10u^3 + 1 \\ h_1^5(u) &= 6u^5 - 15u^4 + 10u^3 \\ h_2^5(u) &= -3u^5 + 8u^4 - 6u^3 + u \\ h_3^5(u) &= -3u^5 + 7u^4 - 4u^3 \\ h_4^5(u) &= -\frac{1}{2}u^5 + \frac{3}{2}u^4 - \frac{3}{2}u^3 + \frac{1}{2}u^2 \\ h_5^5(u) &= \frac{1}{2}u^5 - u^4 - \frac{1}{2}u^3 \end{aligned} \tag{1}$$

The basic *volume map* is so given by:

$$V(u,v,w) = \left[\ h_0^5(w)\quad h_1^5(w)\quad \ldots\quad h_4^5(w)\quad h_5^5(w)\ \right]\begin{bmatrix} S_0(u,v) \\ S_1(u,v) \\ N_0(u,v) \\ N_1(u,v) \\ B_0(u,v) \\ B_1(u,v) \end{bmatrix}$$

with

$$S_0, S_1, N_0, N_1, B_0, B_1 \in \mathcal{P}_2^n \subset \{\Re^2 \to \Re^3\}$$

$$V \in \mathcal{P}_3^n \subset \{\Re^3 \to \Re^3\}$$

Actually, the `PLaSM` implementation is a mapping of this kind:

$$M(u_1, \ldots, u_d) = \left[\ h_0^5(u_d)\quad h_1^5(u_d)\quad \ldots\quad h_5^5(u_d)\ \right]\begin{bmatrix} S_0(u_1, \ldots, u_{d-1}) \\ S_1(u_1, \ldots, u_{d-1}) \\ N_0(u_1, \ldots, u_{d-1}) \\ N_1(u_1, \ldots, u_{d-1}) \\ B_0(u_1, \ldots, u_{d-1}) \\ B_1(u_1, \ldots, u_{d-1}) \end{bmatrix}$$

with

$$S_0, S_1, N_0, N_1, B_0, B_1 \in \mathcal{P}_{d-1}^n \subset \{\Re^{d-1} \to \Re^q\}, \qquad d-1 \leq q$$

$$M \in \mathcal{P}_d^n \subset \{\Re^d \to \Re^q\}, \qquad d \leq q$$

so it works in the general case of d-variate manifolds.

2. Constrained Volume Map

In our approach the duct internal volume is generated by two surface interpolation steps, using three given surfaces. The desired volume map is so obtained by the union of a first solid map interpolating the base section surface with a middle section surface, and a second solid map interpolating the middle section surface with the rotated and translated base section surface (see Figures 2b and 2c). An example of the final constrained duct interior is illustrated in Figures 4a, 4b and 4c, showing it from different points of view.

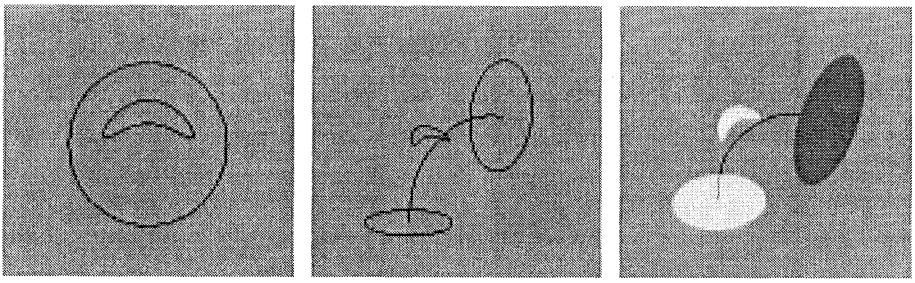

Figure 2. Input maps. (a) 2D curves; (b) embedding in 3D; (c) surfaces to be interpolated.

2.1. Constant-area constraint

In the first phase of computation an initial sequence of interpolated sections is generated. Each section corresponds to one of the w values of a user-specified uniform discretization. Successively, a proper affine transformation is applied to each section to satisfy the constant cross-section constraint.

First we get the solid $V([0,1]^3)$ generated by the basic *volume map* applied to the standard 3-cube, so to obtain an unconstrained duct. Then a family

$$\mathbf{Z} : [0,1] \to Aff^3 : w \mapsto \mathbf{Z}(w)$$

of affine transformations, depending on a parameter $w \in [0,1]$, is properly applied to each cross-section. Each $\mathbf{Z}(w)$ is an *uniform dilatation* in the x, y directions of a local frame $\{\mathbf{e}_x(w), \mathbf{e}_y(w), \mathbf{e}_z(w)\}$, with

$$\begin{aligned}
\mathbf{e}_x(w) &= \frac{\mathbf{q}_x(w)}{\|\mathbf{q}_x(w)\|}, & \mathbf{q}_x(w) &= V(\epsilon,0,w) - V(0,0,w),\\
\mathbf{e}_y(w) &= \frac{\mathbf{q}_y(w)}{\|\mathbf{q}_y(w)\|}, & \mathbf{q}_y(w) &= V(0,\epsilon,w) - V(0,0,w),\\
\mathbf{e}_z(w) &= \frac{\mathbf{q}_z(w)}{\|\mathbf{q}_z(w)\|}, & \mathbf{q}_z(w) &= V(0,0,\epsilon+w) - V(0,0,w),
\end{aligned}$$

and $\epsilon \to 0$. Notice that the local frame $\{\mathbf{e}_x(w), \mathbf{e}_y(w), \mathbf{e}_z(w)\}$ is extracted from the *tangent 3-manifold* to $V([0,1]^3)$ at $V(0,0,w)$.

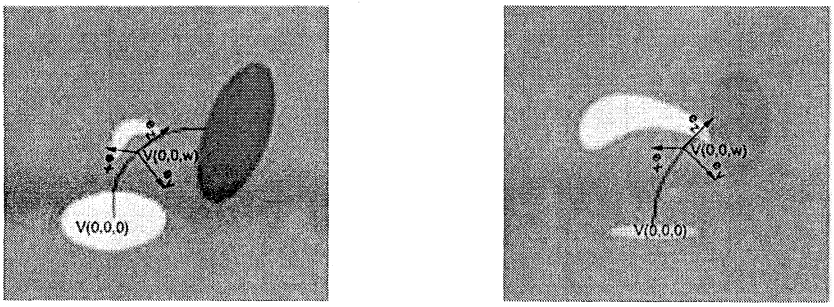

Figure 3. Basis on the tangent 3-manifold at $V(0,0,w)$ with the corresponding cross-sections $B(w)$ and $(\mathbf{Z}(w)(B(w))$.

Each $\mathbf{Z}(w)$, $w \in [0,1]$, is applied to the point set $B(w) = V([0,1]^2 \times \{w\})$. See Figures 3a and 3b to clarify this point.

The family of maps $\mathbf{Z}(w)$ is defined by composition of elementary affine transformations depending on the parameter w. In particular, as usual in graphics, translations $\mathbf{T}(\mathbf{t}(w))$ and inverse $\mathbf{T}(-\mathbf{t}(w))$, rotations $\mathbf{R}(w)$ and inverse $\mathbf{R}^T(w)$, and a scaling $\mathbf{S}(s(w), s(w), 1)$ are composed together:

$$\mathbf{Z}(w) = \mathbf{T}(-\mathbf{t}(w)) \circ \mathbf{R}^T(w) \circ \mathbf{S}(s(w), s(w), 1) \circ \mathbf{R}(w) \circ \mathbf{T}(\mathbf{t}(w))$$

with

$$\mathbf{R}(w) = \begin{bmatrix} \mathbf{e}_x(w) & \mathbf{e}_y(w) & \mathbf{e}_z(w) \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{e}_x(w)^T \\ \mathbf{e}_y(w)^T \\ \mathbf{e}(w)_z^T \end{bmatrix}$$

$$\mathbf{t}(w) = V(0,0,0) - V(0,0,w),$$

and where

$$\mathbf{S}(w) = \begin{bmatrix} s(w) & 0 & 0 \\ 0 & s(w) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

with

$$s(w) = \sqrt{\frac{\text{Area}(0)}{\text{Area}(w)}} = \sqrt{\frac{\int_{S_0([0,1]^2)} dS}{\int_{V([0,1]^2 \times \{w\})} dS}}$$

The surface integral is computed using a general polynomial-integrating algorithm described in [Bernardini, 1991]. The algorithm allows an efficient

computation of integrals of polynomial functions over polyhedral domains of any dimension. This is easily computed in PLaSM by a single primitive

```
INTEGRAL:pol_complex:<i1,i2,...,id>
```

which returns the value of the domain integral of the monomial

$$I^{i_1 \cdots i_d}_{x_1 \cdots x_d} = \int_{\texttt{pol\_complex}} x_1^{i_1} x_2^{i_2} \ldots, x_d^{i_d} \, dV.$$

The domain may have non full dimensionality, e.g. a piecewise-linear curve or surface in $\Re^3$. In particular, when the expression is

```
INTEGRAL:pol_complex:0
```

the volume of the input complex is computed. When the input is the piecewise linear approximation of a curve or surface its lenght or surface area are computed.

2.2. Maps composition

The *constrained volume map* V^* is now obtained by composition of the *volume map* V previously discussed with a family of affine transformations depending on one parameter. In particular, we have that

$$V^* : [0,1]^3 \rightarrow \Re^3$$

is easily defined as:

$$V^*(u,v,w) = (\mathbf{Z}(w) \circ V)(u,v,w)$$

with V quintic transfinite Hermite's interpolation of input maps, and $\mathbf{Z} : [0,1] \rightarrow \mathcal{A}ff^3$, the family of affine transformations previously given.

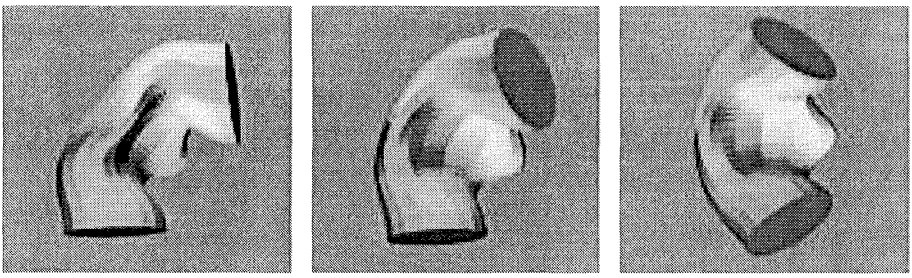

Figure 4. Some views of the final constrained duct.

2.3. Volume Elements

Another issue to analyze is how to sample the domain of the volume map. Different techniques may be used according to decomposition properties we

like to obtain. Among the approaches to triangulation given in literature, we focused our attention over two groups of algorithms: *a priori methods* and *adaptive methods.*

A priori methods. In this class of methods we include the ones that generate a discretization independently from the geometric model to map and approximate. The easiest method is to use some *uniform grid*, which tends to accumulate points where curvature is bigger. A better approach is given by an *incremental refining* of the domain, where the domain is subdivided in quasi-disjoined slices and each of them doubles the number of volume elements. This method is able to produce a better resolution where curvature is smaller.

Adaptive methods. In this class of methods we concentrate on *regular triangulations* decomposition techniques. They can be used to dynamically vary accumulation of points in subregions of a map domain using weights on points. Weights can be assigned according to properties of the mapping function in that regions (such as curvature, derivatives values, and so on). In this way it is possible to obtain a discretization that fits with desired properties of the geometric model in localized subregions of the initial domain. Edelsbrunner and Shah [Edelsbrunner and Shah, 1992] introduced an incremental algorithm for building a *regular triangulation* of a set S of weighted points. At each step they insert a new point in the triangulation, checking first for regularity conditions over present simplices and flipping facets after the insertion. In our case, we need not only regularity conditions among points and facets, but also to introduce strong regularity constraints among domain points, in order to interface FEM simulation codes. In the on-going new implementation of PLaSM with MzScheme [Felleisen et al., 1998] and C++, regular triangulations have been introduced in its geometric kernel, based on a set of libraries developed at Purdue University, and recently merged into the language environment. The main current issue is to find a suitable weighting function to obtain better discretizations of map domain subregions, taking advantages of regular triangulations power.

3. Example

The prototype problem set up to develop the approach described in this paper required the generation of a duct interpolating two extreme circular sections lying on two orthogonal planes, and passing through any intermediate cross-section, defined by a closed Bézier curve. In our approach, we generate the two segments of duct by interpolating from both the extreme sections to the intermediate one.

3.1. Curves

The curves generating the prescribed extreme and intermediate sections are defined in 2D. Curve `C0`(u) is a circle of radius 2 and starting angle $-\frac{\pi}{2}$; curve `L0`(u) is a degree 7 Bézier curve which is symmetric with respect to the y axis.

3.2. Sections

Function `sez0` defines a 2D stripe of constant thickness 0.5. Functions `sez1`, `sez12` and `sez2` define the initial, intermediate and final sections, respectively, appropriately embedded, translated and rotated in 3D space.

First segment. The volume map that generates the duct segment interpolating sections `sez1` and `sez12` is illustrated in the following. It is a Hermite map with constant normal vector fields at the two extremes.

Second segment. The volume map that generates the second section of the duct, from the intermediate to the final sections, is generated in the same way as the first section. Derivative continuity across the intermediate section is preserved by specifying identical interpolated normal fields on the two sides of the common cross-section.

Aggregation. The 3D `tube` is finally defined by stitching together the two parts `tube1` and `tube2`.

Transformation pipeline. Recall that V indicates the Hermite map that generates the solid. The scaling transformation $\mathbf{S}(w)$, $w \in [0,1]$ are defined w.r.t. the origin, and scale uniformly in the x and y directions, while leaving z untouched. To apply the scaling correctly to the section

$$B = V([0,1]^2 \times \{w\})$$

we need first to move the center of scaling to the origin and the section on the plane $z = 0$, then apply the scaling, and finally apply the inverse rigid transformation.

We assume that the centers of transformation are the points $V(0,0,w)$ for $w \in [0,1]$. For each section of the discretization we generate a triplet of orthonormal vectors, defining a local coordinate system.

Function `differences` generates the sequence of vectors difference of two sequences of vectors or points. In a similar way, function `intrinsic_frames` generates the squence of local frames, one for each cross section of the dicretization.

The transformation pipeline for each discrete section is generated by the function `affine_transformations`.

Constrained volume map. The pipeline of affine transformations that satisfy the constant cross-section area constraint if finally composed with the volume map. The details of such implementation go beyond the scope of the present paper. It is anyway interesting to notice that all the implementation amounts to three pages of PLaSM code, and that it was developed and tested in less than one week.

4. Conclusion

In this work we have proposed an analytical solution to the modeling of free-form ducts with cross-sections of constant area, based on the composition of transfinite volume maps with section scaling aiming to satisfy such integral constraint.

This application of the language PLaSM to the geometric modeling of ducts with variable cross-section allowed to evaluate the use of language in generating complex form features by a fully parameterized functional programming approach. The solid duct generation described in this paper can be in fact completely parameterized with respect to number, position and orientation of the given key-sections, and even to their shape. Furthermore, the transfinite solution given here can also generate an optimal decomposition of the duct interior with finite tetrahedral elements. Such a model can be used to simulate the fluid-dynamics problem.

We like to emphasize the fact that such geometric programming approach allows not only to define some geometric object in a fully parameterized way, but also to define compact new methods for geometric shape generation, even when subject to constraints of great mathematical complexity.

Acknowlegments

The problem discussed here has been posed by Roberto Tarditi of FIAT Research Center and by A. Palazzetti and B. Bibbione of FIAT-IVECO. Fabio Gori of the University of Rome "Tor Vergata" is heading a research project aiming to develop a theoretical and experimental approach for the development of ducts of new conception for internal combustion engines of industrial vehicles, where the modeling ideas here presented are being used. We would also like to thank Antonio Di Carlo and Isidoro Del Lungo for useful discussions about theory and practice of ducts folding.

References

Backus, J. (78). Can programming be liberated from the von neumann's style? a functional style and its algebra of programs. *Communications of the ACM*, 21(8).

Backus, J., Williams, J., and Wimmers, E. (1990). An introduction to the programming language fl. In Turner, D., editor, *Research Topics in Functional Programming*. Addison-Wesley.

Backus, J., Williams, J., Wimmers, E., Lucas, P., , and Aiken, A. (1989). Fl language manual, parts 1 and 2. RJ 7100 (67163), IBM Research Report.

Bartels, r., Beatty, J., and Barsky, B. (1987). *An Introduction to Splines for Use in Computer Graphics & Geometric Modeling*. Morgan Kaufmann.

Bernardini, F. (1991). Integration of polynomials over n-dimensional polyhedra. *Computer Aided Design*, 23(1).

Cenciotti, G., Morgia, C., and Paoluzzi, A. (1999). Object-oriented extension of a functional design language. Technical Report RT-DIA-45, Dip. Informatica e Automazione, Univ. Roma Tre.

Coons, S. (1967). Surfaces for computer-aided design of space forms. Technical Report MAC-TR-41, MIT, Cambridge.

Edelsbrunner, H. and Shah, N. (1992). Incremental topological flipping works for regular triangulations. In *8th Annual Symposium on Computational Geometry. ACM.*

Felleisen, M., Findler, R. B., Flatt, M., , and Krishnamurthi, S. (1998). The drscheme project: An overview. *ACM Sigplan Notices*, 33(6).

Ferrucci, V. and Paoluzzi, A. (1991). 'extrusion and boundary evaluation for multidimensional polyhedra. *Computer Aided Design*, 23(1).

Goldman, R. (1987). The role of surfaces in solid modeling. In G.E.Farin, editor, *Geometric Modeling: Algorithms and New Trends*. SIAM Publications.

Gordon, W. (1968). Blending function methods of bivariate and multivariate interpolation and approximation. Technical report, General Motors.

Gordon, W. (1969). Spline-blended surface interpolation through curve networks. *J. Math. Mech.*, 18.

Gori, F. (2000). 'Personal communication'. Dept. of Mechanical Engineering, University of Rome "Tor Vergata", June.

Lancaster, P. and Salkauskas, K. (1986). *Curve and Surface Fitting. An Introduction.* Academic Press.

Paoluzzi, A. (1999). Transfinite blending made easy. Technical Report RT-DIA-40, Dip. Informatica e Automazione, Univ. Roma Tre.

Paoluzzi, A., Baldazzi, C., Pascucci, V., , and Vicentino, M. (2001). *Geometric Programming for Computer Aided Design*. John Wiley & Sons. To appear.

Paoluzzi, A., Bernardini, F., Cattani, C., and Ferrucci, V. (1993). Dimension independent modeling with simplicial complexes. *ACM Transactions on Graphics*, 12(1).

Paoluzzi, A., Pascucci, V., and Vicentino, M. (1995). Geometric programming. a programming approach to geometric design. *ACM Transactions on Graphics*, 14(3).