Threading splines through 3D channels

Ashish Myles and Jörg Peters [a]

[a] *University of Florida*

Abstract

Given a polygonal channel between obstacles in the plane or in space, we present an algorithm for generating a parametric spline curve with few pieces that traverses the channel and stays inside. While the problem without emphasis on few pieces has trivial solutions, the problem for a limited budget of pieces represents a nonlinear and continuous ('infinite') feasibility problem. Using tight, two-sided, piecewise linear bounds on the potential solution curves, we reformulate the problem as a finite, linear feasibility problem whose solution, by standard linear programming techniques, is a solution of the channel fitting problem. The algorithm allows the user to specify the degree and smoothness of the solution curve and to minimize an objective function, for example, to approximately minimize the curvature of the spline. We describe in detail how to formulate and solve the problem, as well as the problem of fitting parallel curves, for a spline in Bernstein-Bézier form.

1 Introduction

A *channel* is a fat curve of varying thickness that represents a contiguous opening between obstacle regions as depicted in Figures 1 and 14. The challenge is to solve the channel problem.

Channel Problem: Given a channel, construct a spline of prescribed smoothness and number of control points that traverses and stays strictly inside the channel.

Examples of channel problems are *automated navigation*, for example, piloting a submarine through a narrow underwater canal; *fitting piping*, for example, threading piping past existing structures while minimizing kinks to decrease resistance to flow; *computing free paths* for analyzing nano-structures; *designing curved circuitry*; parametrizing *cutter paths* so that the cutting tool avoids obstacles and sudden acceleration and deceleration; *graph drawing*, where graphs and networks have to be laid out for better visualization. Many of these tasks are too large or time-constrained for human intervention, say moving control points and visually checking for interference in the 2D projections.

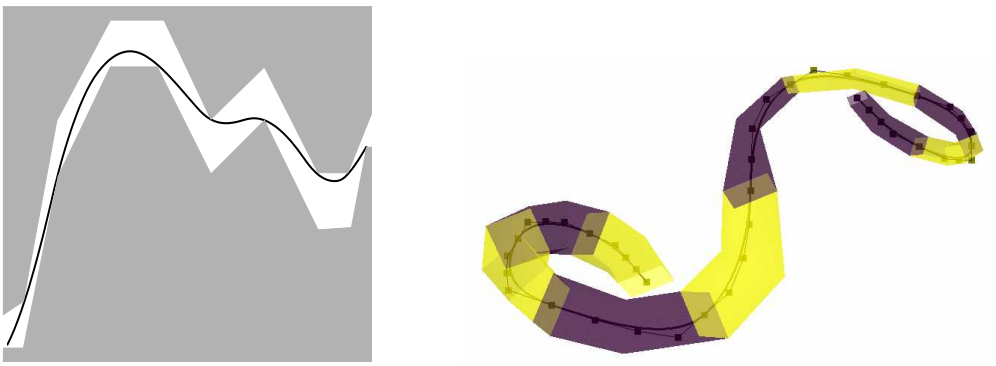

Fig. 1. (*left:*) A narrow **planar channel** traversed by an approximately curvature-minimizing planar spline. (*right:*) A helical **3D channel** (transparent, alternately dark and light gray rendered) that is traversed by a space curve whose Bézier control points are indicated as black squares.

The channel problem can be trivial or extremely difficult depending on the smoothness requirements and the number of pieces prescribed. On one hand, if the budget is no concern and the curvature distribution and optimal smoothness are not crucial, naive approaches work; for example, rounding off the piecewise linear channel center curve, or using one higher-degree polynomial piece per channel segment and Hermite-interpolating heuristically derived values at the channel breakpoints. On the other hand, if the clearances are tight and few spline pieces are needed, the exact constraints of the channel problem are nonlinear and continuous (also called infinite), in that we need non-interference for every parameter value. Off hand, without additional structure, such problems do not have a bounded complexity and are not tractable algorithmically. In our approach, we deal with tight clearances and few pieces by approximating the exact contraints. We approximate safely from one side so that our output is certified to be a solution. In the process, we will miss solutions, but only solutions that an *a posteriori* test, based on the same tight bounding constructs, would also not be able to certify as solutions.

Since continuous, nonlinear optimization is known to be difficult, we need only elaborate on the second claim that a *trial-and-error approach* with automatic refinement of the spline will likely
— generate many more spline control points than necessary, and hence increase the cost for downstream processing, and
— introduce unnecessary oscillations into the spline path.
Moreover, the cost of checking non-interference is not negligible and comes with the same trade-off: less sophisticated tests are faster but yield poorer results in the number of pieces and quality of the resulting curve. Figure 2 illustrates this point with an approach that would, at first sight, appear to be ‘reasonably ok in practice’. The steps are: generate points on the central axis of the channel, construct an interpolating spline curve, test for interference and refine or coarsen. There are many possible interpretations of this general recipe and possibly one that works better than the specializations we tried. The examples in Figure 2, use chordlength

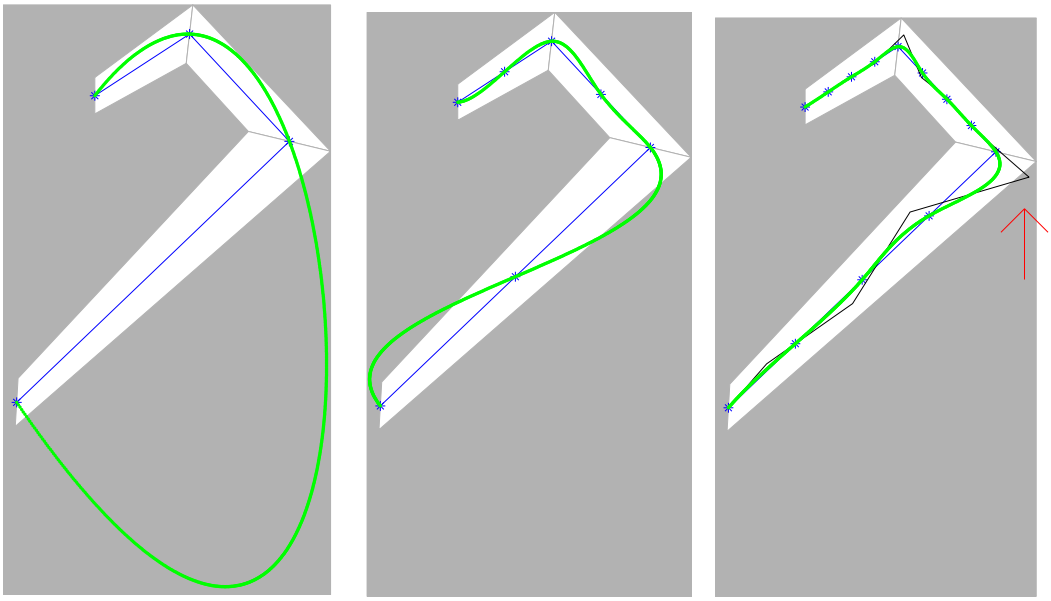

Fig. 2. A **naive approach** to the channel problem: spline interpolation of the channel center line with an increasing number of curve pieces. The rightmost channel shows additionally the cubic spline control polygon. The arrow indicates that the spline control polygon does not stay inside the channeland therefore still cannot certify that the curve lies inside the channel.

parametrization, Matlab's cubic `spline` routine and uniform refinement. After two refinements, the 12 polynomial pieces of the curve *visually* fit into the channel. But the spline control polygon does not lie inside. That means that a control-point based test, such as checking the convex hull, would report interference and trigger a third refinement. Also note the oscillations in the spline. The strategy proposed in this paper returns the sinle polynomial curve piece sketched in Figure 3 below and shown in detail in Figure 12, page 16. The solution process also *certifies* that the curve lies in the channel.

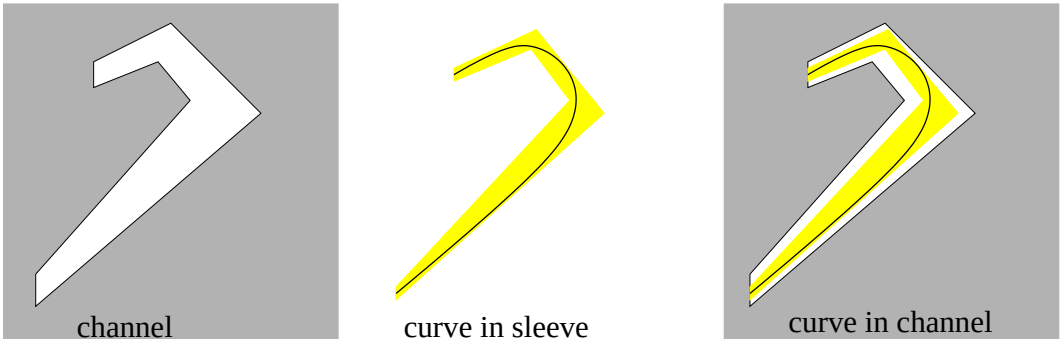

Fig. 3. The **basic idea**. Given the (simple 2D) channel (*left*), a piecewise linear enclosure or sleeve for a candidate curve is constructed (*middle*) and the problem is solved by fitting the sleeve into the channel (*right*).

Our approach to solving the channel problem is to construct a (piecewise linear) sleeve around a candidate space curve; then it is sufficient to constrain this sleeve – rather than the original nonlinear curve – to stay within the channel as illustrated in Figure 3. Carefully formulated, this approximation results in a *linear feasibility problem* that is solvable as a Linear Program, say by the simplex method. (An alternative formulation, as semi-definite programming problem, will not be considered here.) A solution to the linear feasibility problem solves the original problem and, due to the tightness of the sleeve that we use, only solutions with very low clearance are missed. The missed solution curves are exactly those that also the *a posteriori* intersection test, based on piecewise linear enclosure of the curve with the same granularity, would miss. (Note that testing intersection with Newton's method would only be safe if we used interval arithmetic [7]. Efficient interval arithmetic, in turn, depends on tight piecewise linear enclosures.) That is, a solution curve that would be missed by the proposed approach would also be declared 'potentially intersecting' by an *a posteriori* test of the same granularity.

While the channel problem only asks for a feasible curve, we can add optimality and uniqueness by augmenting the feasibility problem with a linear (or quadratic) optimization function. The result is the standard linearly constrained minimization problem:

$$\min f(b) \text{ subject to linear equality and inequality constraints.}$$

The structure of the paper is as follows.
In Section 2, we review and select a sleeve.
In Section 3, we define spline, sleeves and parameterization.
In Section 4, we list the constraints in full detail.
In Section 5, we add an objective function.
In Section 6, we specialize the setup to 2D and parallel curve problems.
In Section 7, we illustrate the approach with several examples.

2 Bounding constructs and related work

Construction of a sleeve around the spline curve is equivalent to enclosing the spline curve with linear pieces. However, in the channel problem, the coefficients of the spline curve are unknown and sought as the solution of the feasibility problem.

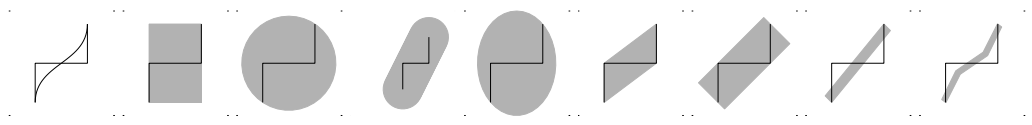

Fig. 4. **Enclosures based on control points**: (*from left to right*:) cubic curve with control polygon, axis-aligned box, bounding circle, Filip [5] (scaled by 1/2), bounding ellipse [17], convex hull and 8-dop [1,8,9], oriented bounding box [6], fat arc [15], 3-piece slefe [**?**].

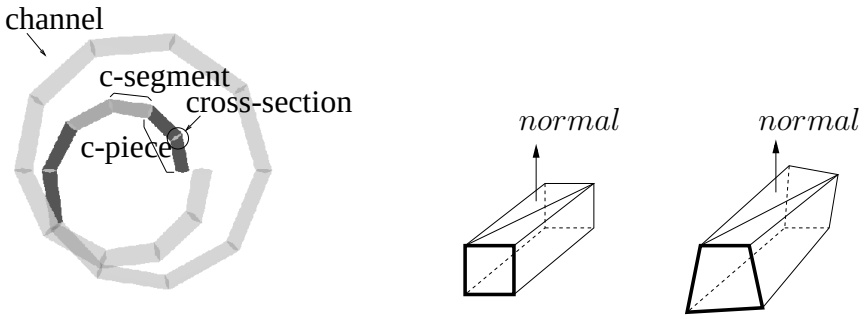

Fig. 5. **Components of a channel**. (*left*) An entire helix-shaped channel viewed from the top. (*right*) A close-up view of two c-segments with the front cross sections emphasized by thick lines.

Therefore one would like the enclosure to have three properties.

(i) The enclosure should depend linearly on the coefficients of the spline representation.
(ii) The enclosure should be as narrow as possible.
(iii) The enclosure should be refinable to adaptively adjust to where the channel is particularly narrow or tricky to navigate.

The literature offers many choices of enclosures or bounding constructs that could in principle be used to construct a sleeve (see Figure 4). However, requirement (i) rules out oriented bounding boxes and convex hull based approaches, or building a local coordinate system and bounding box in the normal direction [10]. In fact, Figure 12 illustrates that these enclosures are not sufficiently narrow even for simple channels. Requirement (ii) rules out the use of even looser bounding constructs such as bounding spheres and axis-aligned bounding boxes. The best match of linearity, tightness and refinability for the channel problem are the enclosures developed in [12,**?**]. However, even these require a smart formulation to reduce the problem to the standard linearly constrained minimization problem.

For *functions* in one variable, a channel problem was formulated in [11]. However, for functions, as opposed to free-form curves, the problem is much simpler since the parametrization is fixed and 'above' and 'below' make sense. Another closely related class of problems appears in graph drawing [3,16]. There, however, the emphasis is on a large number of piecewise linear curves with few pieces.

3 Modeling the problem

We treat both the spline and the channel as a sequence of N *pieces* (see Figure 5), one to one matching each channel piece with one spline piece. A spline piece is one polynomial piece, but a channel piece can consist of n_c segments. That is, one spline piece can be threaded through a longer stretch of the channel. Each spline

piece is enclosed by the n_e segments of its enclosing sleeve.

Since we are dealing with parametrized space curves, we need to create a correspondence of spline pieces with the channel. We could do this by explicitly matching spline parameters to channel parameters, i.e. traversing in lockstep. The other extreme is to allow any piece of the spline to be associated with any piece of the channel. The first option is convenient for implementation since it simplifies the constraints. However, we found that it is too restrictive: good solutions were not found. The drawback of the second solution is not only that we need to check every breakpoint against every channel segment, but also that the curve can turn back onto itself. We avoid both extremes by matching pieces, specifically, associating the break points of the sleeve with specific channel segments. This allows working locally and prevent multiple traversal of segments.

In the next two subsections, we define the channel and the sleeve in detail so that we can formalize the correspondence of in Section 4.

3.1 Channel definition

Each channel piece, or c-piece, is a sequence of *cross-sections*. A cross-section has σ edges. For example, in Figure 5, each cross-section is quadrilateral, i.e. $\sigma = 4$. The vertices need not be co-planar. Adjacent cross-sections span a channel segment, short *c-segment*. The σ faces of the c-segment are split into triangles with normals consistently oriented outwards. In Figure 5, *left*, each c-piece consists of two adjacent c-segments.

The cross-sections (and hence the channel segments) do not have to be convex. However, the formulated constraints will force the curve through the (convex) intersection of all the halfspaces defined by the facets of the channel segment.

In our implementation, the faces are triangulated using the same pattern to simplify the channel representation and σ is constant; but the channel can be designed with different values of σ at every cross-section and the triangulation can be arbitrary with consistent normals.

3.2 Spline definition

For simplicity, we represent the space curve in Bézier form; it is not difficult to use a b-spline representation instead. A polynomial b of degree d with respect to the

interval $[0..1]$ can be written in terms of the Bézier basis functions $B_\ell^d(t)$ as:

$$b(t) := \sum_{\ell=0}^{d} b_\ell B_\ell^d(t), \quad B_\ell^d(t) := \binom{d}{\ell}(1-t)^{d-\ell}t^\ell,$$

where the b_ℓ are the *control points*. If $b_\ell = (b_{\ell,x}, b_{\ell,y}, b_{\ell,z})$ is a three-dimensional control point then $b(t) = (b_x(t), b_y(t), b_z(t))$.

3.3 slefe *definition*

Our method is based on slefes (slefe is short for: subdividable linear efficient function enclosure, and is pronounced like sleeve) developed in [12], [13], and [**?**]. For polynomials of degree d in Bézier form, the upper and the lower boundary of the slefe are a linear combination of piecewise linear functions $\overline{a}_i^d$ and $\underline{a}_i^d$, $i = 1, \ldots, d-1$ (see Equation 3 or 1 below). Specific $\overline{a}_i^d$ and $\underline{a}_i^d$ have been tabulated for various combinations of d and n_e and are available for download from [18].

Since slefes are a relatively new construct, we give some background in the next paragraph. This information is not necessary to use slefes in the context of this paper, but may be of interest to some readers.

The functions $\overline{a}_i^d$ and $\underline{a}_i^d$ bound a specific polynomial a_i^d of degree d from above and below for $t \in [0..1]$:

$$\underline{a}_i^d(t) \le a_i^d(t) \le \overline{a}_i^d(t).$$

The Bézier coefficients of a_i^d, after scaling by the negative number $\frac{i-1}{i} + \frac{d-i-1}{d-i} - 2$ are (see Figure 6 for $d = 5$ and $i = 2$)

$$\frac{0}{i}, \frac{1}{i}, \ldots \frac{i-1}{i}, \left(\frac{i}{i} = \frac{d-i}{d-i}\right), \frac{d-i-1}{d-i}, \ldots \frac{1}{d-i}, \frac{0}{d-i}$$

It is easy to check, that a_i^d is uniquely defined by the constraints

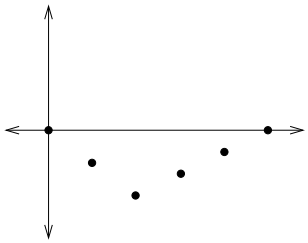

Fig. 6. Bézier coefficients of a_2^5.

$$a_i^d(0) = a_i^d(1) = 0, \quad D_j a_i^d = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{else} \end{cases} \text{ where } D_j b := b_{j-1} - 2b_j + b_{j+1}.$$

For degree $d = 3$, a_1^d has the coefficients $0, -2/3, -1/3, 0$ and $a_2^d(t) = a_1^d(1-t)$ so that $\begin{bmatrix} D_1 a_1^d & D_2 a_1^d \\ D_1 a_2^d & D_2 a_2^d \end{bmatrix} = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$. Since there are only few a_i^d and these polynomials are independent

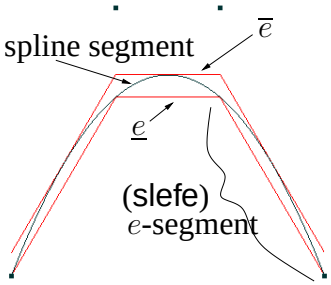

Fig. 7. An $n_e = 3$-piece slefe with upper boundary $\overline{e}$ and lower boundary $\underline{e}$. The four Bézier control points of the cubic piece are shown as squares.

of the spline pieces b, the *narrowest* enclosures have been pre-computed and tabulated once and for all for various n_e: the tables simply contain the values of $\underline{a}_i^d$ and $\overline{a}_i^d$ at n_e+1 (equally spaced) breakpoints.

The functions $\underline{a}_i^d$, $\overline{a}_i^d$ are weighted by the second differences $D_i b$ and the result is added to the linear interpolant $l(b) := (1-t)b_0 + tb_d$ of the end points. (In general, say for a spline in B-spline form, $l(b)$ can be chosen as any piecewise linear approximation, for example the control polygon.) That is, on the interval $[0..1]$,

$$\overline{e} \geq b(t) \geq \underline{e}, \tag{1}$$

where

$$\overline{e} := l(b) + \sum_{i=1}^{d-1} \underline{a}_i^d \min\{0, D_i b\} + \overline{a}_i^d \max\{0, D_i b\}$$
$$\underline{e} := l(b) + \sum_{i=1}^{d-1} \underline{a}_i^d \max\{0, D_i b\} + \overline{a}_i^d \min\{0, D_i b\}.$$

Note that $\overline{e}$ and $\underline{e}$ are piecewise linear since $l(b)$, $\underline{a}_i^d$ and $\overline{a}_i^d$ are, but that $\overline{e}$ and $\underline{e}$ do not linearly depend on the coefficient b_i due to the $\min$ and $\max$ selections. However, with care, we will still be able to formulate the expressions as linear *inequality* constraints in Section 5. Also note that either $\min\{0, D_i b\} = 0$ or $\max\{0, D_i b\} = 0$ so that $\overline{e}$ has only d summands.

We observe in practice that only little is lost by fitting a slefe inside the channel rather than a spline inside the channel since the tables recording the breakpoint values of $\underline{a}_i^d$ and $\overline{a}_i^d$ have been pre-optimized. [14] shows that for any convex cubic piece b, the ratio between the tightest possible sleeve for b and the slefe $\overline{e}$ and $\underline{e}$ of b is always less than 1.07.

3.4 Sleeve definition

Conforming to the segmentation of the channel, the slefe of each polynomial piece is called an enclosure or *e-piece* (Figure 7). Each linear segment in the e-piece is an *e-segment* and there are n_e e-segments per e-piece. With $\nu \in \{x, y\}$ for a planar curve or $\nu \in \{x, y, z\}$ for a spatial curve, e-pieces are defined in the following terms.

$b^p_{\ell,\nu}$ is the ν-component of the ℓ^{th} Bézier control point of the p^{th} polynomial piece of the solution spline curve. ${b^p}_\nu$ (note the gap in the subscript) is the vector whose ℓth entry is $b^p_{\ell,\nu}$.

$l(b^p)$ is the linear interpolant of the first and last control points of the p^{th} polynomial piece in the solution.

$\overline{e}^p_\nu$ and $\underline{e}^p_\nu$ are the the upper and lower e-pieces, respectively, of the p^{th} polynomial piece for the ν^{th} coordinate.

$D_i b^p$ is the ith second difference of the Bézier coefficients of $b^p(t)$:
$D_i b^p := b^p_{i-1} - 2b^p_i + b^p_{i+1}$.
The ν^{th} coordinate of $D_i b^p$ is denoted by ${D_i b^p}_\nu$ and
${D b^p}_\nu$ is the vector of the $d-1$ second differences of ${b^p}_\nu$.

$\underline{a}^d$ and $\overline{a}^d$ are matrices of size $n_e + 1 \times d - 1$ whose i^{th} column lists the values of the lower, respectively upper slefe of a^d_i at $(s/n_e)_{s=0,\ldots,n_e}$. These values have been tabulated [18]. For $d = 3$ and $n_e = 3$, the table for $i = 1$ is

$t =$	0	1/3	2/3	1
$\underline{a}^d_1$	-.0695214343	-.4398918047	-.3153515940	-.0087327217
$\overline{a}^d_1$	0	-.3703703704	-.2962962963	0

and the table for $i = 2$ is mirror symmetric since $a^d_2(1-t) = a^d_1(t)$.

3.5 Correspondence: matching parametrizations of channel and spline enclosure

We parametrize the channel and the slefe so that we can locally associate c-segments with e-segments and vice versa. The start parameter of each (c- or e-) piece is $t = 0$ and the end parameter is $t = 1$. The internal parameters increase uniformly with the index as illustrated in Figure 8, *right*. Let $\hat{v}^p_c$ be a breakpoint (on the c^{th} cross section of the p^{th} piece) of the channel. Then $\frac{c}{n_c}$ is its per-piece, local parameter. Let $s := \lceil c\frac{n_e}{n_c} \rceil$ be the smallest integer larger than $c\frac{n_e}{n_c}$. Then e^p_s is the sleeve's breakpoint with the smallest parameter such that $\frac{s}{n_e} \geq \frac{c}{n_c}$, i.e. with a parameter just to

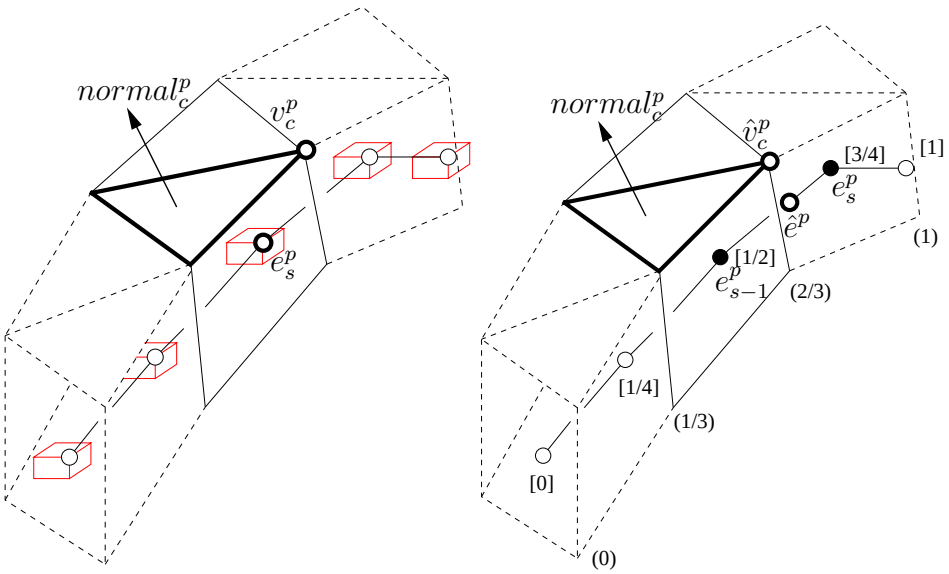

Fig. 8. **Parametrization and correspondence** for geometric constraints (3a) *left* and (3b) *right*. The boxes at each enclosure breakpoint represent the $2^{dim} = 8$ possible choices for $e_s^p \in \{\underline{e}_{x,s}^p, \overline{e}_{x,s}^p\} \times \{\underline{e}_{y,s}^p, \overline{e}_{y,s}^p\} \times \{\underline{e}_{z,s}^p, \overline{e}_{z,s}^p\}$.

the right of $\frac{c}{n_c}$. We define the point on the sleeve corresponding to $\hat{v}_c^p$ as

$$\hat{e}^p := ue_{s-1}^p + (1-u)e_s^p, \qquad \text{where } u := s - c\frac{n_e}{n_c}. \tag{2}$$

For example, Figure 8, *right* (see also Figure 9, *right*), shows the fractional parameters of the c-piece in parentheses and the parameters of the corresponding e-piece of the sleeve in brackets for $n_e = 4$, $n_c = 3$, $c = 2$, and $s = \lceil c(4/3) \rceil = \lceil 8/3 \rceil = 3$. Then $u = s - c(4/3) = 1/3$ yields $\hat{e}^p = (e_{s-1}^p + 2e_s^p)/3$.

4 The constraint system

We can now formulate the 3D channel problem as a linearly constrained feasibility problem for fitting a degree d spline with N polynomial e-pieces through a channel with N c-pieces. The following subsections make each of the constraints precise:

(1) Min-max selection of second differences.
(2) Channel correspondence and smoothness of the spline.
(3a) Sleeve corners are in the channel.
(3b) Channel corners are outside the sleeve.
(3c) slefe segment is inside the channel segment.

The constraints apply to pieces p, coefficients ℓ, differences i, and coordinates ν:

$$p = 1, \ldots, N, \quad i = 1 \ldots, d-1, \quad \ell = 0 \ldots, d, \quad \nu \in \{x, y, z\}.$$

4.1 Min-Max selection

Our first challenge is to formulate the $\min$ and $\max$ selection as linear equalities and inequalities. To this end, we define, in addition to the unknown coefficients

$$b^p_{\ell,\nu}, \qquad \text{the auxiliary variables} \quad D_i b^p, \quad D_i b^{p-}_{\ \nu} \quad \text{and} \quad D_i b^{p+}_{\ \nu}.$$

That is, we have a total of $N \times (3(d-1)+d+1) \times dim$ scalar-valued unknowns, where $dim = 3$ for a space curve and $dim = 2$ for a planar curve. The name $D_i b^{p-}_{\ \nu}$ for a single, scalar unknown is somewhat lengthy, but it is precise and would appear in code as dbm[p,i,ν]. $D_i b^{p-}_{\ \nu}$ will play the role of $\min\{0, D_i b^p{}_\nu\}$ and $D_i b^{p+}_{\ \nu}$ will stand in for $\max\{0, D_i b^p{}_\nu\}$. That is, for $p = 1..N$, $i = 1..(d-1)$ and $\nu \in \{x, y, z\}$ we will compute a sleeve $\tilde{e}^p_\nu$ and $\underset{\sim}{e}{}^p_\nu$

$$\tilde{e}^p_\nu := l(b^p_\nu) + \sum_{i=1}^{d-1} \underline{a}^d_i D_i b^{p-}_{\ \nu} + \overline{a}^d_i D_i b^{p+}_{\ \nu}, \tag{3}$$

$$\underset{\sim}{e}{}^p_\nu := l(b^p_\nu) + \sum_{i=1}^{d-1} \underline{a}^d_i D_i b^{p+}_{\ \nu} + \overline{a}^d_i D_i b^{p-}_{\ \nu} \tag{4}$$

such that

$$\tilde{e}^p_\nu \geq \overline{e}^p_\nu \geq \underline{e}^p_\nu \geq \underset{\sim}{e}{}^p_\nu.$$

The constraints are as follows.

(1) Min-max selection of second differences: ($dim \times N \times (d-1) \times 3$) constraints $D_i b^{p+}_{\ \nu} \geq D_i b^p{}_\nu$ and $D_i b^{p+}_{\ \nu} \geq 0$ and $D_i b^{p+}_{\ \nu} + D_i b^{p-}_{\ \nu} = D_i b^p{}_\nu$
where $D_i b^p{}_\nu := b^p_{i-1\,\nu} - 2b^p_{i\,\nu} + b^p_{i+1\,\nu}$.

This implies $D_i b^{p-}_{\ \nu} \leq D_i b^p{}_\nu$ and $D_i b^{p-}_{\ \nu} \leq 0$ and that, for some $\epsilon^p_{i,\nu} \geq 0$,

$$D_i b^{p-}_{\ \nu} := \min\{0, D_i b^p{}_\nu\} - \epsilon^p_{i,\nu} \text{ and } D_i b^{p+}_{\ \nu} := \max\{0, D_i b^p{}_\nu\} + \epsilon^p_{i,\nu}.$$

Therefore $\tilde{e}^p_\nu = \overline{e}^p_\nu + \sum_{i=1}^{d-1} \epsilon^p_{i,\nu}(\overline{a}^d_i - \underline{a}^d_i) \geq \overline{e}^p_\nu$ and $\underset{\sim}{e}{}^p_\nu \leq \underline{e}^p_\nu$ as claimed. That is, we are potentially fitting a wider sleeve than the slefe into the channel. This is the correct inclusion since fitting the wider sleeve guarantees that the narrower slefe of the solution curve fits; this, in turn, implies that the solution curve fits into the channel. In fact, the objective function discussed in Section 5 will drive the sum of the $\epsilon^p_{i,\nu}$ to zero.

4.2 Correspondence and Smoothness

Like all linear equality constraints, the next set of constraints could be eliminated by substitution. The first constraint prevents a curve piece from freely sliding to

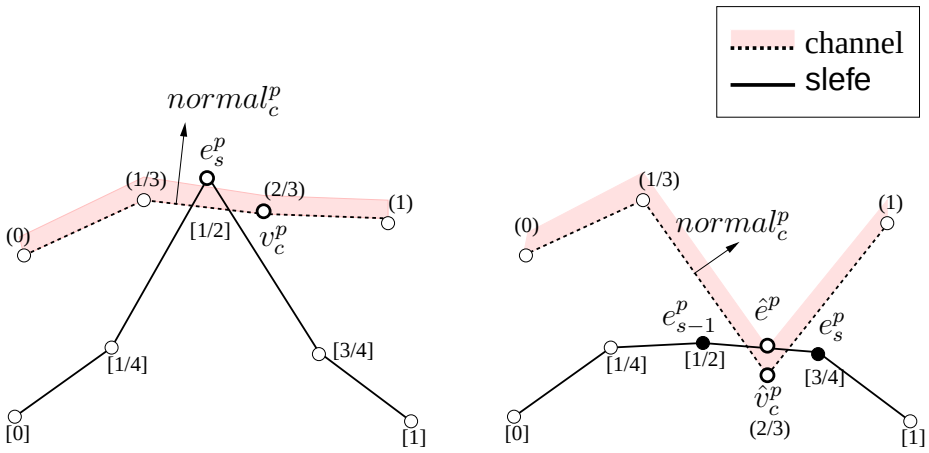

Fig. 9. **Geometric constraints** (3a) *left* and (3b) *right*. Lateral cross-section view of channel violations to be prevented: (*left*) a slefe breakpoint could pierce a channel segment, or (*right*) a channel breakpoint could pierce the slefe.

another c-segment (see also (3c) below).

(2) Correspondence and Smoothness: $((3N-2)\times dim)$ constraints
$b_{0,\nu}^p =$ center of the starting cross-section of the p^{th} piece.
C^0 continuity: $b_{d,\nu}^p = b_{0,\nu}^{p+1}$
C^1 continuity: $b_{d,\nu}^p = \frac{1}{2}\left(b_{d-1,\nu}^p + b_{1,\nu}^{p+1}\right)$.

Additional constraints that can be expressed as linear equalities or inequalities, such as C^2 continuity, can be just as easily added to the system.

4.3 Interference checks

The remaining constraints combine the x, y, and z components to ensure that the curve stays inside the channel. Here the piecewise linear nature of $\underline{e}_\nu$ and $\overline{e}_\nu$ pays off. For $e_{s,\nu}^p \in \{\underline{e}_{s,\nu}^p, \overline{e}_{s,\nu}^p\}$, let $e_i^p := (e_{s,x}^p, e_{s,y}^p, e_{s,z}^p)$ be one of 2^{dim} possible combinations of upper or lower slefe breakpoints for a given s (cf. Figure 8, *left*).

Using the parametric correspondence of Section 3.5, we can test the sign of the signed volume of tetrahedra with one point on the e-piece and a *corresponding* channel triangle as base. Equivalently, we can compute the sign of the inner product of the triangle's normal with the edge pointing from any of the three vertices of the triangle to the point on the e-piece. Specifically, by keeping e_s^p inside the corresponding channel segment, Constraint (3a) and (3c) below makes sure that the sleeve does not pierce the channel. Conversely, Constraint (3b) makes sure that the channel does not pierce the sleeve. The number of constraints can be reduced by only testing concave, inward-pointing breakpoints of the channel. Both in (3a) and (3b), $normal_c^p$ is the outward-pointing channel normal. Note that the constraints depend on $b_{s,\nu}^p$ via Equation 3 on page 11.

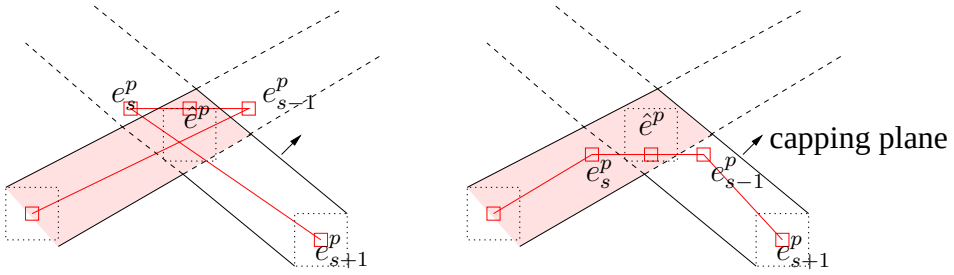

Fig. 10. **Overshooting** (prevented by capping half-spaces in Constraint (3c)).

(3a) Sleeve corners in the channel: ($2^{dim} \times 2\sigma \times N \times (n_e + 1)$) constraints
For each piece and segment, for each e^p_s where $e^p_{s,\nu} \in \{\underline{e}^p_{s,\nu}, \overline{e}^p_{s,\nu}\}$ and every triangle of the corresponding c-segment, choose any vertex v^p_c of the triangle to form the constraint (cf. Figures 8 and 9, *left*)

$$normal^p_c \cdot (e^p_s - v^p_c) \leq 0.$$

(3b) Channel corners outside the sleeve: ($2^{dim} \times 4\sigma \times N \times (n_c + 1)$) constraints
At each channel cross-section, for every triangle on the two adjacent c-segments connected at the cross-section, choose any vertex $\hat{v}^p_c$ on the triangle and the point $\hat{e}^p$ on the sleeve defined by Equation 2, p.10 and enforce (cf. Figures 8 and 9, *right*)

$$normal^p_c \cdot (\hat{e}^p - \hat{v}^p_c) \leq 0.$$

By constraining *one* interior point $\hat{e}^p := ue^p_{s-1} + (1-u)e^p_s$ of the line segment from e^p_{s-1} to e^p_s to lie in *both* (extended) channel segments, (3b) actually enforces that the whole segment lies inside the channel, provided e^p_{s-1} and e^p_s do. This is remarkable since testing to which side of a facet of the sleeve the point $\hat{v}^p_c$ lies (in order to rule out that it pierces the sleeve) is off-hand not linear in the unknowns: it is a constraint on the sign of the signed volume of the tetrahedra where three points lie on the unknown sleeve and the fourth point is $\hat{v}^p_c$. Effectively, the formulation (3b) reduces a nonlinear to a linear constraint!

There is one final consideration: Constraint (3a) only confines e^p_s to the tunnel-like intersection of the halfspaces forming the corresponding c-segment. We still need to cap off this tunnel to prevent overshooting, as illustrated in 2D in Figure 10, *left*. That is, a slefe breakpoint could satisfy (3a) but lie outside the channel. The planes of the next and previous c-segment that cap off the tunnel are those whose normal forms a sharp angle with the lateral, non-cross-section edges of the c-segment oriented from $s-1$ to s. At the s^{th} cross-section, the normals must be nearly aligned with the oriented lateral edges; at the $s-1^{st}$ cross-section they must be nearly opposite to the oriented lateral edges.

(3c) slefe inside channel segment: ($2^{dim} \times \{2+2\} \times N \times n_e$) constraints
For each e^p_s where $e^p_{s,\nu} \in \{\underline{e}^p_{s,\nu}, \overline{e}^p_{s,\nu}\}$ and every capping plane (with normal $normal^p_j$) of the c-segment, choose any vertex v^p_c in the capping plane to form

the constraint

$$normal_c^p \cdot (e_s^p - v_c^p) \leq 0.$$

We observe that for any curve violating (3c) there appears to be another feasible curve with lower sum of absolute second differences (Figure 10, *right* without the loop) that can be obtained by purely local operations. Given our choice of objective function in Section 5 below, this explains why we have never observed Constraint (3c) taken on.

5 The objective function

The standard Linear Program has the form

$$\min f(b), \qquad \text{subject to linear equality and inequality constraints.}$$

The previous section listed the constraints (1,2,3a,3b,3c) and we may add an objective function f. (If there is no objective function, then the problem is called a linear feasibility problem.) A possible choice for f is to minimize the sum of the absolute second-differences, penalizing loops and oscillations. Since $D_i b^{p+}_\nu \geq 0$, $D_i b^{p-}_\nu \leq 0$, and $D_i b^{p+}_\nu + D_i b^{p-}_\nu = D_i b^p{}_\nu$, we have that $D_i b^{p+}_\nu - D_i b^{p-}_\nu$ is minimal if one of $D_i b^{p+}_\nu$ and $D_i b^{p-}_\nu$ is zero and the other equals $D_i b^p{}_\nu$, or, equivalently, $|D_i b^p{}_\nu| = D_i b^{p+}_\nu - D_i b^{p-}_\nu$. Hence, to minimize kinks, we choose the objective function

$$\sum_{p=1}^{N} \sum_{i=1}^{d-1} \sum_{\nu \in \{x,y,z\}} D_i b^{p+}_\nu - D_i b^{p-}_\nu .$$

This problem can now be handed to a standard linear constraint solver, say solvers that use the interior point method or the simplex method. If the problem has no solution, some solvers return an answer minimizing infeasibility and a list of violated constraints. This indicates the location where to increase the degree d or the number of slefe segments n_e, or to decrease the number of channel segments n_c per piece.

6 Specialization to planar channels and extension to parallel curves.

A 2D channel has an upper and a lower piecewise linear boundary (with normals oriented outwards). The slefe stays within such a planar channel if we enforce a simplified version of (3a) and (3b). We arbitrarily name one channel boundary 'upper' and the other 'lower', $normUpper_c^p$ and $normLower_c^p$ their normals and

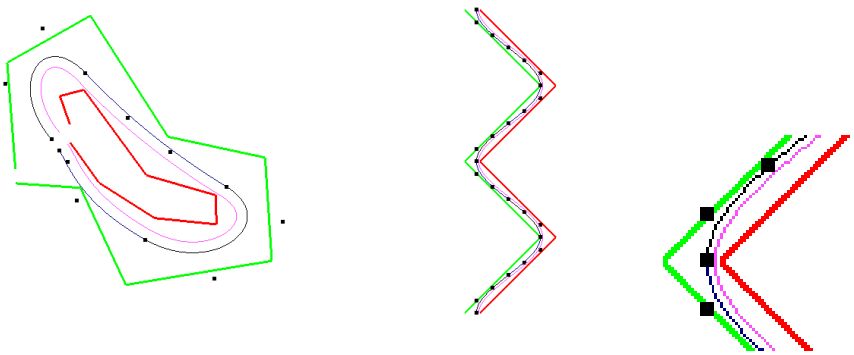

Fig. 11. Two **parallel planar curves** traversing channels. (*left*) a spline and its offset avoiding an inner and an outer channel boundary, (*right two*) a channel with long sections and sharp turns with detail enlarged.

$upper_c^p$ and $lower_c^p$ their respective breakpoints. Then (3a) and (3b) become

$$\begin{aligned} normUpper_c^p \cdot (e_s^p - upper_c^p) \le 0, \qquad & normLower_c^p \cdot (e_s^p - lower_c^p) \le 0, \\ normUpper_c^p \cdot (\hat{e}^p - upper_c^p) \le 0, \qquad & normLower_c^p \cdot (\hat{e}^p - lower_c^p) \le 0. \end{aligned}$$

The formulation of the 2D problem can be adapted to thread multiple parallel curves through a channel. For example, to thread two parallel curves, width w apart, the constraints for the upper channel change to:

$$normUpper_c^p \cdot (e_s^p - upper_c^p) \le w, \qquad normUpper_c^p \cdot (\hat{e}^p - upper_c^p) \le w.$$

This results in a spline solution $p_1(t)$ with a buffer distance of w between the curve and the upper channel. Since $(w\ normUpper_c^p) \cdot normUpper_c^p = w$, this is effectively equivalent to narrowing the channel by w. Then, a second curve $p_2(t) := p_1(t) + w\hat{n}(t)$ can fit through the channel, where $\hat{n}(t)$ is the normal at $p_1(t)$ oriented toward the upper channel. Since p_2 is an offset curve, it is in general not polynomial [4]. Moreover, offset curves are smooth only if w is smaller than the smallest radius of curvature of p_1; otherwise self-intersections have to be trimmed off and smoothed out. Of course w could also be chosen as $w(t)$ with a varying, but prescribed value.

7 Examples

The examples throughout the paper were run on a Pentium 4 2.4GHz with 512 MB RAM using the freely available open-source *PCx* [2] linear constraint solver. The constraints were automatically generated by a script on input of the channel vertices.

Figure 12 demonstrates that a fit based on bounding the control polygon would not

be able to generate our tight solutions to the channel problem even after subdivision.

The channel in Figure 13, *left* consists of twelve c-segments. To accommodate the sharp turns while maintaining C^1 continuity, we chose $n_c = 1$, i.e. one polynomial per c-segment. Formulated in 3D (the channel has 'depth' perpendicular to the viewing direction), the problem was solved in 0.24 seconds.

Both Figure 5, *right* and Figure 13, *right* consist of twenty segments. The solution spline is of degree $d = 4$, $n_c = 2$ and one polynomial piece fits each pair of c-segments. The solution time varies with the narrowness of the channel: 2.64 seconds for the left and 3.42 seconds for the right example in Figure 13.

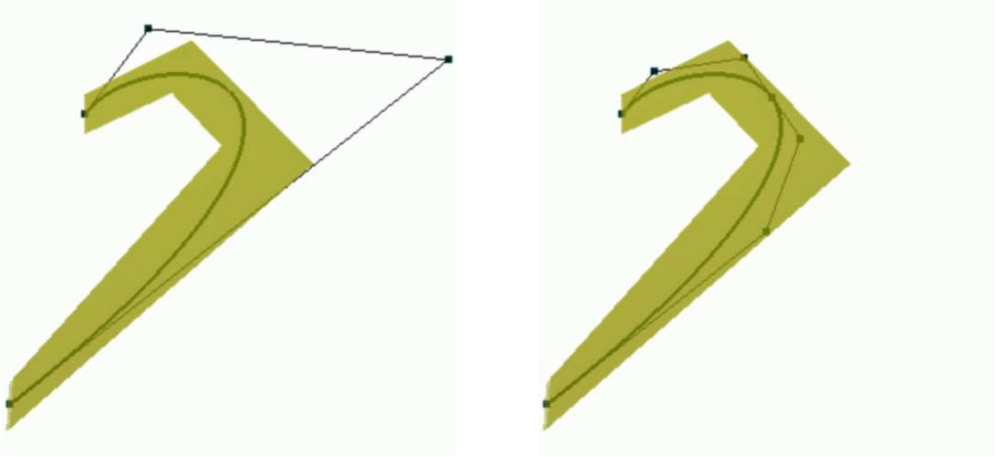

Fig. 12. Solution curve for a **2D channel**. The Bézier control points are indicated as black squares. (*left*) The control polygon of the solution lies well outside the channel although the slefe fits within. (*right*) Also after subdivision, the control polygon of the solution exceeds the channel boundaries.

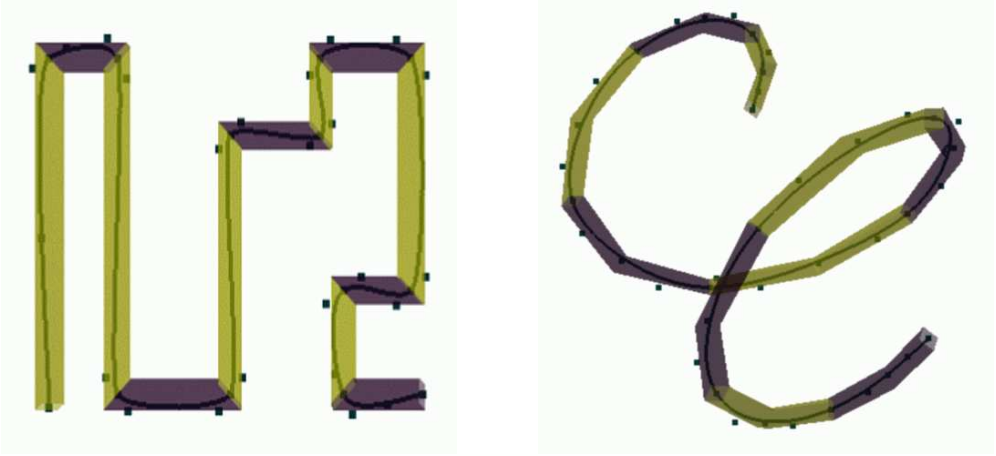

Fig. 13. (*left*) A 2D channel with 1 c-segment per c-piece. (c-pieces alternate in color). There is one piece of degree 3 per c-piece. (*right*) A 3D channel with two c-segments per c-piece. The spline curve is of degree 4.

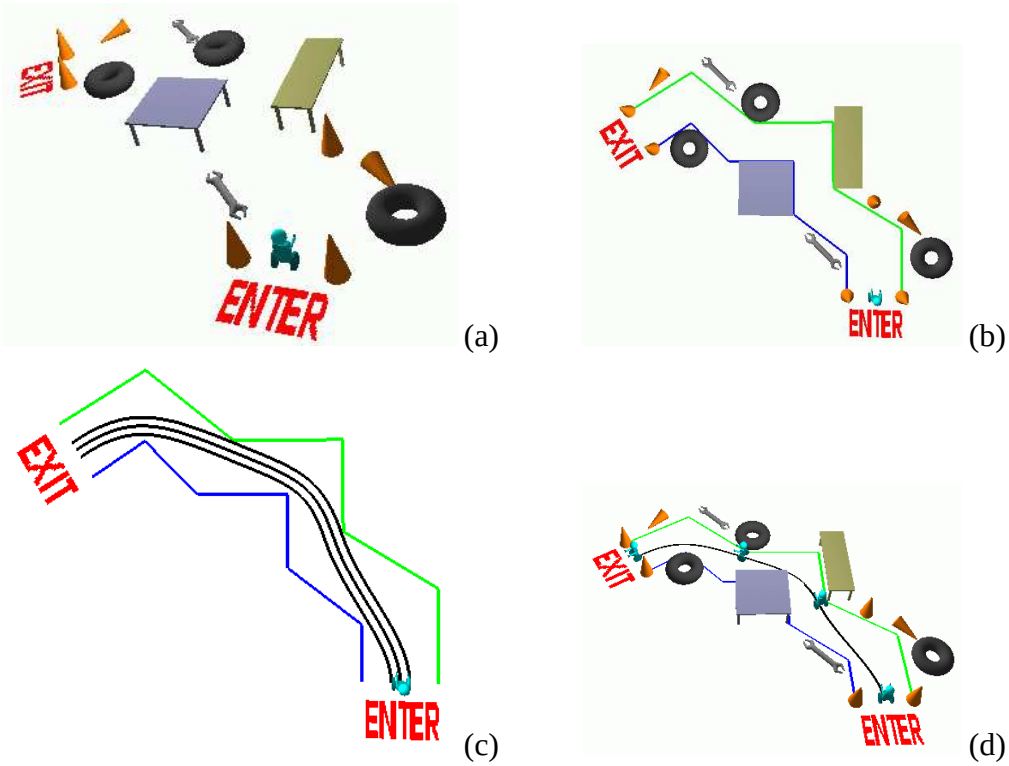

Fig. 14. (a) Robot faced with obstacles. (b) Abstraction of the problem by defining a channel. (c) Solution of the channel problem (with offsets). (d) **Robot path** (*black, smooth* curve).

The final example is Figure 14. Here a safe, smooth path for a (0.96-unit wide) robot is computed. The top-left image shows the obstacles. As a first step, an channel is traced through the obstacles so that our path-fitting method can be applied. The bottom-left shows the spline path fit through the channel maintaining a 0.48-unit buffer on either side. The bottom-right shows the robot at equal parameter increments traversing the environment.

8 Summary

The sleeve-based approach may be viewed as bridging a gap between techniques of computational geometry and geometric design since we tightly link linear discrete with non-linear continuous representations. While the constraints appear at first sight complex, they are all necessary to *guarantee* correctness while allowing for a wide range of solutions with flexible parametrization within each segment for parametric space curves. Although costlier than naive approaches, the overall optimization framework has the advantage of providing an objective function to pick solutions with preferred properties.

9 Acknowledgements

This research was supported by NSF Grant CCR-9901894 at the University of Florida. Xiaobin Wu contributed the slefe tabulations available in [18].

References

[1] A. Crosnier and J. R. Rossignac. Technical section — tribox bounds for three-dimensional objects. *Computers and Graphics*, 23(3):429–437, June 1999.

[2] Joe Czyzyk, Sanjay Mehrotra, Michael Wagner, and Stephen Wright. http://www-fp.mcs.anl.gov/otc/Tools/PCx.

[3] D. Dobkin, E.R. Gansner, E. Koutsofios, and S.C. North. A path router for graph drawing. In *Proceedings of the 14th annual symposium on Computational Geometry*, pages 415–416. ACM Press, New York, 1998. June 1-10 1998, Minneapolis, MN.

[4] R. T. Farouki and T. Sakkalis. Pythagorean hodographs. *IBM Journal of Research and Development*, 34(5):736–752, September 1990.

[5] D. Filip, R. Magedson, and R. Markot. Surface algorithms using bounds on derivatives. *Computer Aided Geometric Design*, 3(4):295–311, 1986.

[6] Stefan Gottschalk, Ming Lin, and Dinesh Manocha. OBBTree: A hierarchical structure for rapid interference detection. In *Proceedings of the ACM Conference on Computer Graphics*, pages 171–180, New York, August 4–9 1996. ACM.

[7] Jon Rokne H. Ratschek. *Geometric Computations with Interval and New Robust Methods: With Applications in Computer Graphics, GIS and Computational Geometry*. Horwood Publishing, Chichester, 2003.

[8] Timothy L. Kay and James T. Kajiya. Ray tracing complex scenes. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 269–278, August 1986.

[9] James T. Klosowski, Joseph S. B. Mitchell, Henry Sowizral, and Karel Zikan. Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, January 1998.

[10] Leif Kobbelt, Katja Daubert, and Hans-Peter Seidel. Ray tracing of subdivision surfaces. In *Rendering Techniques '98 (Proceedings of the Eurographics Workshop)*, pages 69–80, New York, June 1998. Springer-Verlag.

[11] D. Lutterkort and J. Peters. Smooth paths in a polygonal channel. In *Proceedings of the 15th annual symposium on Computational Geometry*, pages 316–321, 1999.

[12] David Lutterkort. *Envelopes for Nonlinear Geometry*. PhD thesis, Purdue University, May 2000.

[13] J. Peters. Efficient one-sided linearization of spline geometry. In R.R. Martin, editor, *Mathematics of Surfaces X*, page 297:319. IMA, 2003.

[14] J. Peters and X. Wu. On the optimality of piecewise linear max-norm enclosures based on slefes. In *Proceedings of the 2002 St Malo conference on Curves and Surfaces*, pages 335–344, 2003.

[15] Thomas W. Sederberg, Scott C. White, and Alan K. Zundel. Fat arcs: A bounding region with cubic convergence. *Comput. Aided Geom. Design*, 6:205–218, 1989.

[16] Roberto Tamassia. Graph drawing. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*. Elsevier, 2000.

[17] E. Welzl. Smallest enclosing disks (balls and ellipsoids). In Hermann Maurer, editor, *Proceedings of New Results and New Trends in Computer Science*, volume 555 of *LNCS*, pages 359–370, Berlin, Germany, June 1991. Springer.

[18] X. Wu and J. Peters. The SubLiME (subdividable linear maximum-norm enclosure) package. downloadable from http://www.cise.ufl.edu/research/SurfLab/download/SubLiME.tar.gz.