AIAA Guidance, Navigation and Control Conference and Exhibit
18 - 21 August 2008, Honolulu, Hawaii

AIAA 2008-7135

On-line Path Generation for Small Unmanned Aerial Vehicles Using B-Spline Path Templates

Dongwon Jung* and Panagiotis Tsiotras†
Georgia Institute of Technology, Atlanta, GA, 30332-0150

In this study we investigate the problem of generating a smooth, planar reference path, given a family of discrete optimal paths. In conjunction with a path representation by a finite sequence of square cells, the generated path is supposed to stay inside a feasible channel, while minimizing certain performance criteria. Constrained optimization problems are formulated subject to geometric (linear) constraints, as well as boundary conditions in order to generate a library of B-spline path templates. As an application to the vehicle motion planning, the path templates are incorporated to represent local segments of the entire path as geometrically smooth curves, which are then joined with one another to generate a reference path to be followed by a closed-loop tracking controller. The on-line path generation algorithm incorporates the path templates such that continuity and smoothness are preserved when switching from one template to another along the path. Combined with the $\mathcal{D}^*$-lite path planning algorithm, the proposed algorithm provides a complete solution to the obstacle-free path generation problem in a computationally efficient manner, suitable for real-time implementation.

I. Introduction

Guidance and navigation control of mobile agents has been an important research topic for several decades. For unmanned aerial vehicles (UAVs), in particular, the ability of fully automated guidance and navigation allows UAVs to accomplish missions under various situations with minimal human intervention. Because of the stringent operational requirements and the restrictions imposed on UAVs for autonomy, safety, efficiency, etc., a complete solution to fully automated guidance and navigation (including path planning, path generation, path following, etc.) for *low-cost* UAVs is a real challenge.

Guidance and navigation or aerial vehicles is being traditionally accomplished by simple tracking controllers of a sequence of way-points, which are computed by a mission planner. A better solution to the guidance and navigation control problem takes into account the dynamic constraints of the vehicle, resulting in a smooth, flyable path that passes through the way-points. The way-points must be connected such that the generated smooth paths preserve the continuity of the curvature between line and arc segments, while minimizing the maximum curvature.[1,2] In Ref. [3], the authors proposed a dynamic trajectory smoothing algorithm, by which the path segments are smoothed to yield an extremal trajectory with explicit consideration of the kinematic constraints of a fixed-wing UAV.

Splines have been widely adopted when computing smooth, dynamically feasible trajectories for UAVs. A series of cubic splines was employed in Ref. [4] to connect straight line segments in a near-optimal manner. The authors of Ref. [5] presented an implicit time-parametrization of the trajectory using a B-spline representation. The design of an obstacle-avoiding B-spline path has been dealt with by Berglund et al.,[6] whereas real-time modification of a spline path has been proposed in Ref. [7]. The advantage of employing (B-)splines in generating a smooth path stems from the fact that the path can be represented using a smaller number of parameters than using a complete geometric description of the path. Accordingly, it is straightforward to deal with a relatively small number of parameters both for path optimization and for on-line implementation. This results in a small computational cost; thus splines are suitable for on-board implementation.

*Post-doctoral Fellow, School of Aerospace Engineering, dongwon.jung@gatech.edu, AIAA Member.
†Professor, tsiotras@gatech.edu, AIAA Associate Fellow.

The path planning problem using B-splines involves finding the solution of a constrained optimization problem not only to avoid forbidden regions, but also to generate flyable trajectories. In Ref. [8], a polygonal channel comprised of piecewise polylines is assumed to be known. A B-spline curve that remains within the channel is found by quadratic programming. This was made possible by adopting tight linear envelopes for the splines,[9] by which a B-spline is represented as an approximate bounding polygon. In the approach of Ref. [8], one dimensional B-spline functions are utilized to describe a smooth path subject to the channel constraints.

The complete solution to fully automated guidance and navigation is simplified by the use of a hierarchical control structure[10–12] that may include path planning, path generation, and path following control layers. In this paper, we investigate the path generation problem for obtaining a reference path to be used for the path following controller layer. Incorporating a high-level path planning algorithm such as $\mathcal{D}^*$-lite, we obtain an on-line path smoothing algorithm using a set of B-spline path templates. We extend the results of Ref. [9] in two dimensions, by incorporating two dimensional B-spline curves in the associated constrained optimization problem. Instead of smoothing the entire path from the initial position to the final goal, we smooth the path segments over a finite planning horizon with respect to the current position of the UAV. This approach is somewhat similar to implementing a receding horizon control in the path generation stage. To this end, we first generate a set of all possible combinations of discrete path cell sequences within a finite planning horizon. We thus obtain the corresponding channel constraints; the path templates are computed by (off-line) optimization. These path templates are stored in a library for on-line implementation. Each B-spline segment from these path templates is stitched with one another to come up with a reference path *on-line*, while preserving second order continuity along the entire path. The proposed approach has the benefit of generating a reference path that is appropriate for a complex environment with obstacles. The reader may want to compare this approach with the extremal trajectories between way-points only (and without obstacles), computed in Ref. [3]. In addition, the algorithm is fast, and generates a feasible path using off-line path templates, while the smoothing algorithm proposed in Ref. [4] requires on-line optimization, thus it may increase the overall computational burden. Finally, it should be noted that similarly to the algorithm proposed in Ref. [7], the real-time modification of the reference path for a dynamically changing environment is implicitly considered in the proposed algorithm by incorporating the high-level path planner.

The paper is organized as follows. In Section II, we discuss tight envelopes for two dimensional B-spline curves. The details about the two dimensional channel optimization problem, which is an extension of the results in Ref. [8], are discussed in Section III. The algorithm for path templates generation and the on-line smoothing by stitching path segments are presented in Section IV and Section V, respectively.

II. Tight Envelope for B-Spline Curves

II.A. Tight envelopes for a B-spline function[9]

An one-dimensional spline function $b(u)$ is expressed by[13]

$$b(u) = \sum_{j=0}^{m} b_j N_j^d(u), \tag{1}$$

where b_j are the control points and $N_j^d(u)$ are the B-spline basis functions of degree d, which are defined over a non-decreasing knot sequence $\{u_k\}_{k=0}^{m+d+1}$, such that $u_0 \le u_1 \le \cdots \le u_{m+d+1}$. The number of knots is determined by the sum of the number of control points $(m+1)$ and the B-spline order $(d+1)$. The first and the last knots of the sequence should have multiplicity $(d+1)$ for a B-spline to pass through the first and the last control points, such that $u_0 = u_1 = \cdots = u_d$ and $u_{m+1} = u_{m+2} = \cdots = u_{m+d+1}$, respectively. The B-spline basis functions are computed by the well-known *Cox-de Boor* recursion formulas[13] as follows,

$$N_j^0(u) = \begin{cases} 1 & \text{if } u_j \le u < u_{j+1}, \\ 0 & \text{otherwise,} \end{cases} \tag{2a}$$

$$N_j^d(u) = \frac{u - u_j}{u_{j+d} - u_j} N_j^{d-1}(u) + \frac{u_{u+d+1} - u}{u_{j+d+1} - u_{j+1}} N_{j+1}^{d-1}(u). \tag{2b}$$

The B-spline basis functions have several useful properties. Among them, it is well known that a B-spline basis function has local support,[14] that is, $N_j^d(u)$ is a non-zero polynomial over the knot span $[u_j, u_{j+d+1})$.

Alternatively, given any knot span $[u_j, u_{j+1})$, at most $(d+1)$ B-spline basis functions of degree d are non-zero. This local support property is important for curve design, since it allows one to modify the B-spline curve locally, without changing the entire shape.

Let the control polygon of the B-spline $\ell(u)$ be defined by piecewise line segments connecting the control points b_j at the Greville abscissae,[15]

$$u_j^* = \sum_{i=j+1}^{j+d} u_i/d, \tag{3}$$

such that at each Greville abscissae the equation $\ell(u_j^*) = b_j$ is satisfied. Accordingly, the envelope of the B-spline specifies a bound on the distance between $b(u)$ and its control polygon $\ell(u)$. This envelope should provide a good estimate of the shape of the B-spline by approximating the spline using limited information. Although a number of bounds for splines are proposed in the literature, the bound derived in Refs. [9,16] is known to be a tight, quantitative bound. It is therefore possible to approximate B-splines by piecewise linear envelopes, as the envelopes carry most of salient information about the curve itself. In light of this fact, the envelopes have the advantage of being incorporated in a B-spline optimization problem, thus simplifying the analysis.

The envelopes in Ref. [9] are expressed in terms of the weighted second difference of the control points,

$$\Delta_2 b_j \triangleq b'_{j+1} - b'_j, \qquad b'_j = \frac{b_j - b_{j-1}}{u_j^* - u_{j-1}^*}, \tag{4}$$

and by the non-negative, convex functions over the interval $[u_k^*, u_{k+1}^*]$ $(k = 0, 1, \cdots, m)$ as follows,

$$\beta_{ki}(u) = \begin{cases} \sum_{j=i}^{\overline{k}} (u_j^* - u_i^*) N_j^d(u) & i > k, \\ \sum_{j=\underline{k}}^{i} (u_i^* - u_j^*) N_j^d(u) & j \le k, \end{cases} \tag{5}$$

where, $\underline{k}$ and $\overline{k}$ denote respectively the index of the first and the last B-spline basis functions that are nonzero over the corresponding interval. Subsequently, the distance between the spline function $b(u)$ and its control polygon $\ell(u)$ is calculated as follows,[17]

$$b(u) - \ell(u) = \sum_{i=\underline{k}}^{\overline{k}} \Delta_2 b_i \beta_{ki}(u). \tag{6}$$

Furthermore, by choosing the minimum and maximum variation of the weighted second difference as $\Delta_i^- = \min\{0, \Delta_2 b_i\}$ and $\Delta_i^+ = \max\{0, \Delta_2 b_i\}$, we obtain the maximum offsets in both positive and negative directions with respect to the control polygon, which, in turn, become the upper and lower bounds of the spline function with respect to the control polygon, as follows

$$\ell(u) + \sum_{i=\underline{k}}^{\overline{k}} \Delta_i^- \beta_{ki}(u) \le b(u) \le \ell(u) + \sum_{i=\underline{k}}^{\overline{k}} \Delta_i^+ \beta_{ki}(u). \tag{7}$$

Since the $\beta_{ki}(u)$'s are non-negative and convex functions over the corresponding interval $[u_k^*, u_{k+1}^*]$, the maximum function values occur at each end of the interval, i.e., at each Greville abscissae. Then the piecewise linear functions $\underline{e}(u)$ and $\overline{e}(u)$ that interpolate their values at each Greville abscissa can be employed to represent tight envelopes of the spline function, as follows

$$\begin{aligned} \underline{e}(u) &= \ell(u) + \mathcal{L}\left(\sum \Delta_i^- \beta_{ki}(u_k^*), \sum \Delta_i^- \beta_{k+1,i}(u_{k+1}^*) \right), \\ \overline{e}(u) &= \ell(u) + \mathcal{L}\left(\sum \Delta_i^+ \beta_{ki}(u_k^*), \sum \Delta_i^+ \beta_{k+1,i}(u_{k+1}^*) \right), \end{aligned} \tag{8}$$

where $\mathcal{L}(\cdot, \cdot)$ denotes a linear interpolation between two values. Therefore, the maximal bounds from the B-spline function to its control polygon are obtained by the simple form given as follows

$$\underline{e}(u) \le b(u) \le \overline{e}(u). \tag{9}$$

Figure 1 shows a cubic B-spline function $b(u)$ over the knot sequence $u \in [0,\ 1]$. The bounding envelopes $\underline{e}(u)$ and $\overline{e}(u)$ are drawn by dotted and dashed-dot lines, respectively.

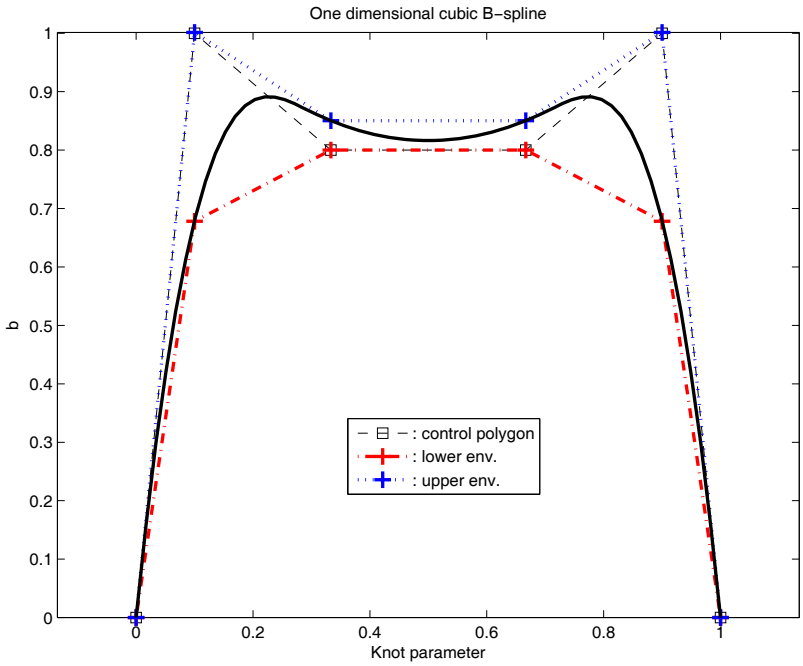


Figure 1. One-dimensional cubic B-spline bounding envelopes.

II.B. Tight envelope for planar B-spline curves

A two dimensional planar B-spline curve $\boldsymbol{b}(u) = [b^1(u)\ b^2(u)]^\mathsf{T}$ is expressed in terms of the B-spline basis functions, as follows

$$\boldsymbol{b}(u) = \sum_{j=0}^{m} \boldsymbol{b}_j N_j^d(u), \tag{10}$$

where, $\boldsymbol{b}_j = [b_j^1\ b_j^2]^\mathsf{T}$ are the corresponding control points. It is also assumed that the B-spline curve is clamped at the first and last control points by assigning the $(d+1)$ multiple knots at the first and last knots.

At each Greville abscissa u_k^*, the one-dimensional bound derived by Eq. (9) constitutes a two-dimensional bounding box, whose ith axis is determined by the one-dimensional envelope as follows

$$\underline{e}^i(u_k^*) \le b^i(u_k^*) \le \overline{e}^i(u_k^*), \qquad i = 1, 2. \tag{11}$$

Let the axis-aligned bounding box be denoted by S^k. Then the curve segment $\boldsymbol{b}(u)$, for $u \in [u_k^*, u_{k+1}^*]$, lies in the convex combination of S^k and the consecutive box S^{k+1} owing to the linearity of $\underline{e}(u)$ and $\overline{e}(u)$. This is denoted by

$$H^k = \mathcal{L}(S^k, S^{k+1}). \tag{12}$$

Note that H_k is the convex hull of S^k and S^{k+1}, which is circumscribed by the edges of S^k and S^{k+1}, and the lines that connect the corners of S^k and S^{k+1}. Let $\boldsymbol{v}_i^k$ $(i = 1, \cdots, 4)$ be the line segments connecting the corresponding corners of S^k and S^{k+1}. In other words, $\boldsymbol{v}_1^k$ connects the lower left corner of S^k to the lower left corner of S^{k+1}, $\boldsymbol{v}_2^k$ connects the lower right corner of S^k to the lower right corner of S^{k+1}, and so on. Figure 2 shows these line segments. As mentioned above, the convex hull H^k over the knot $u \in [u_k^*, u_{k+1}^*]$ consists of parts of the edges of S^k and S^{k+1} and exactly two extra line segments $\boldsymbol{\ell}_L^k$ and $\boldsymbol{\ell}_R^k$ chosen among $\boldsymbol{v}_i^k$. The line segments $\boldsymbol{\ell}_L^k$ and $\boldsymbol{\ell}_R^k$ are separated by the line that connects the control points $\boldsymbol{b}_k$ and $\boldsymbol{b}_{k+1}$. Thus, $\boldsymbol{\ell}_L^k$ denotes the left envelope line segment and $\boldsymbol{\ell}_R^k$ denotes the right envelope line segment with respect to the control polygon. These line segments $\boldsymbol{\ell}_L^k$ and $\boldsymbol{\ell}_R^k$, where $k = 0, \cdots, m$ are joined together to form piecewise linear envelopes of the B-spline curve $\boldsymbol{e}_L(u)$ and $\boldsymbol{e}_R(u)$, respectively.

It might be the case, however, that two line segments do not intersect. Consider, for instance, the line segments $\boldsymbol{\ell}_R^{k-1}$ and $\boldsymbol{\ell}_R^k$ shown in Fig. 2. In order to form piecewise linear envelopes by a set of line segments, those line segments are extended to get the intersection point $\boldsymbol{u}_R^k$. Consequently, the envelopes $\boldsymbol{e}_L(u)$ and

$\boldsymbol{e}_R(u)$ are determined by a set of line segments between the intersection points $\boldsymbol{u}_L^k$ and $\boldsymbol{u}_R^k$, respectively. Figure 3 shows an example of two-dimensional bounding envelopes of a given B-spline curve, which reveals

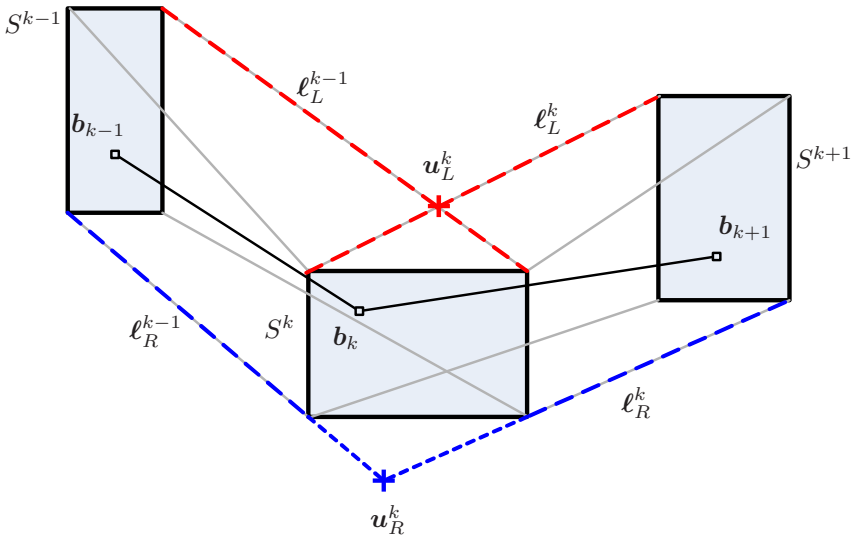


Figure 2. Constructing the bounding envelopes of a planar curve from neighboring bounding boxes.

that the entire B-spline curve stays inside the envelope generated by $\boldsymbol{e}_L(u)$ and $\boldsymbol{e}_R(u)$.

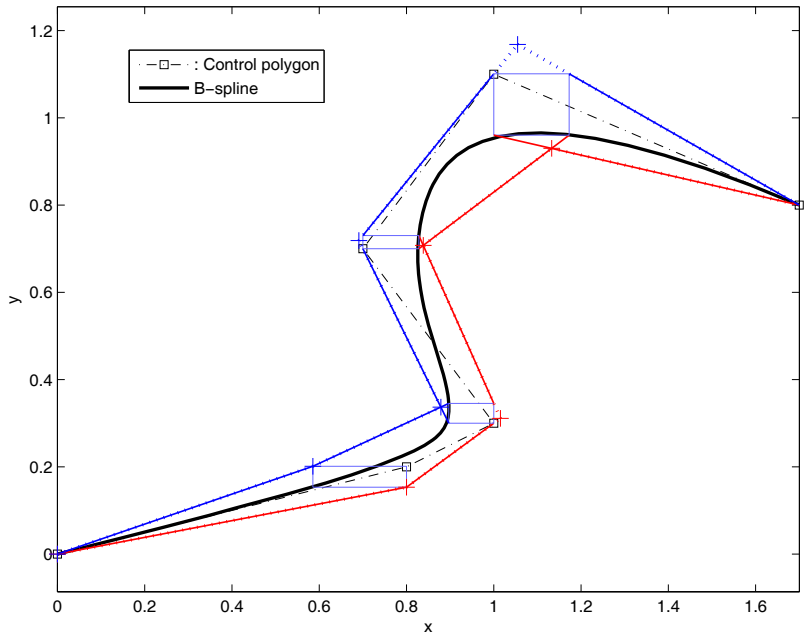


Figure 3. Bounding envelopes e_L (blue) and e_R (red) of two-dimensional cubic B-spline .

III. Obstacle Avoidance Path Optimization

In this section we employ the B-spline curve as a primitive for path design. Hence the resulting path not only is a smooth flyable path due to its inherent smoothness property, but is also represented by a few parameters in conjunction with the quantitative bounds of the curve itself. The constrained optimization problem is then formulated by constructing channel constraints, ensuring that the designed path stays on the known channel region that is assumed to be obstacle-free.

III.A. Channel constraints for obstacle avoidance

In the following optimization problem, a channel is referred to a feasible region on the plane over which a geometric path will be optimized. We assume that two non-intersecting polylines constitute a channel, separating the feasible region from the obstacle region. Hence, we can compute a path that avoids any obstacles, while satisfying the given performance criteria.

In Refs. [8, 18], linear inequality constraints with respect to the knot parameters are incorporated in the B-spline function optimization. Given non-intersecting channel polygons, the inequality expressions are formulated in conjunction with the lower and upper envelopes $\underline{e}(u)$ and $\overline{e}(u)$ with respect to the chosen parameters of the B-spline function, the control points, such that the B-spline function stays between the polygons. On the other hand, because we deal with a planar B-spline curve that cannot be represented by a B-spline function, the channel constraints should be formulated in terms of geometric constraints, as opposed to the linear inequalities with respect to the control point parameters. Nonetheless, it is crucial that the constraint equations capture the condition that the given geometric channel polygon contains the envelopes of the B-spline curve.

To this end, we first let $\mathcal{O} \subset \mathbb{R}^2$ be the obstacle region. Then we introduce a signed distance-map function $f(\boldsymbol{x}; \boldsymbol{\ell})$, $\boldsymbol{x} \in \mathbb{R}^2$ with respect to a polygonal line $\boldsymbol{\ell}$ in order to provide the relative distance metric for formulating geometric constraints, as follows

$$f(\boldsymbol{x}; \boldsymbol{\ell}) \triangleq s \, \min\{d_1, \; d_2\}, \tag{13}$$

where, $d_1 \in \mathcal{D}_1 = \{\|\boldsymbol{x} - \boldsymbol{c}_i\|, \; i = 0, \cdots, q\}$ is the distance from a point $\boldsymbol{x}$ to a corner point $\boldsymbol{c}_i$ of the polygonal line $\boldsymbol{\ell}_i$, $d_2 \in \mathcal{D}_2 = \{d(\boldsymbol{x}, \boldsymbol{\ell}_i), \; i = 0, \cdots, q-1\}$ is the perpendicular distance from $\boldsymbol{x}$ to the line segment $\boldsymbol{\ell}_i$ that connects two consecutive corner points $\boldsymbol{c}_i$ and $\boldsymbol{c}_{i+1}$, and s is a sign value which dictates the location of the point $\boldsymbol{x}$ with respect to $\boldsymbol{\ell}$ as follows,

$$s = \begin{cases} +1, & \boldsymbol{x} \in \mathcal{O}, \\ \; 0, & \boldsymbol{x} \in \boldsymbol{\ell}, \\ -1, & \boldsymbol{x} \notin \mathcal{O}. \end{cases} \tag{14}$$

Figure 4 shows the distance-map function with respect to an arbitrary zig-zag shape polygonal line. Far-away points from the line have bigger values, whereas points close to the line yield smaller values. The information about the relative location of the points with respect to the polygonal lines is determined by its sign.

In order to formulate inequality constraints similar to those in Ref. [8], we make use of the fact that the envelopes of a planar B-spline are characterized by the feature points $\boldsymbol{u}_L^k$ and $\boldsymbol{u}_R^k$ of the envelopes $\boldsymbol{e}_L(u)$ and $\boldsymbol{e}_R(u)$, shown in Fig. 2. It then follows that if all the feature points are placed inside the feasible region, then each bounding box at $u = u_k^*$ of the B-spline curve will be completely contained in the feasible region as well. To this end, let $\boldsymbol{\ell}_L$ and $\boldsymbol{\ell}_R$ be the polygonal lines representing the obstacle boundaries. Using Eq. (13) the feature points $\boldsymbol{u}_L^k$ and $\boldsymbol{u}_R^k$ at each Greville abscissa $u = u_k^*$ should satisfy the following inequality expressions,

$$f(\boldsymbol{u}_L^k; \boldsymbol{\ell}_L) \leq 0, \qquad f(\boldsymbol{u}_R^k; \boldsymbol{\ell}_R) \leq 0, \tag{15}$$

where $k = 0, \cdots, m$.

In addition to Eqs. (15), extra constraints are required to impose the conditions of complete inclusion of the envelopes inside the feasible channel: Recall that the envelope of the B-spline curve is derived by the convex hull of each bounding box, which results in the envelope being represented by piecewise polygonal lines. As a result, not only each bounding box at $u = u_k^*$ of the B-spline curve should be contained in the feasible region as discussed above, but also the convex hull is required to be contained in the feasible region. Since the obstacle boundaries are represented by polylines, in order to ensure that the convex hull of the B-spline envelope lies inside the channel, we have to make sure that corner points of the obstacle boundaries are placed outside the envelope. Because both the envelope and the obstacle boundaries are polylines, this condition along with Eq. (15) suffices to impose the channel constraints. Specifically, we apply this condition to the concave corners with respect to the obstacle region as marked by the triangles in Fig. 5. Accordingly, we formulate the inequality constraints using Eq. (13) at each concave corner points, in conjunction with the piecewise linear envelopes as follows,

$$f(\boldsymbol{c}_l^{\boldsymbol{\ell}_L}; \boldsymbol{e}_L) \geq 0, \qquad f(\boldsymbol{c}_m^{\boldsymbol{\ell}_R}; \boldsymbol{e}_R) \geq 0, \tag{16}$$

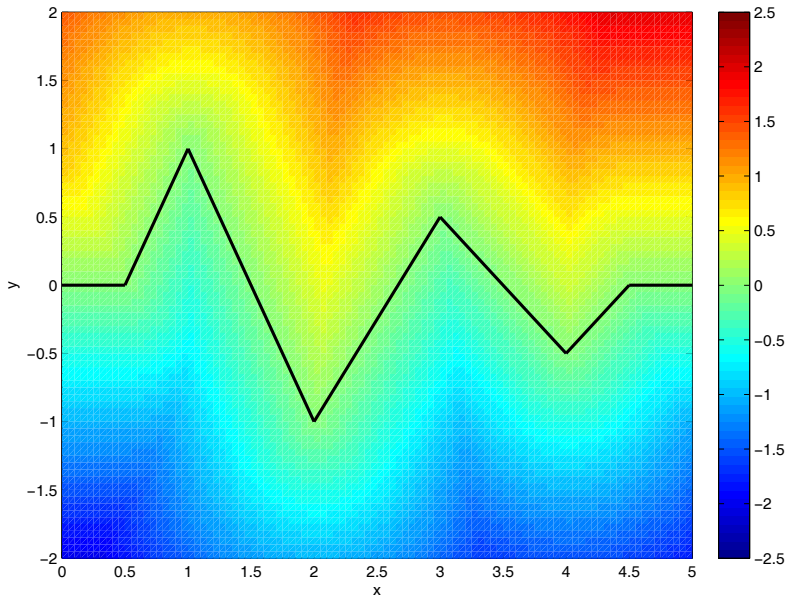


Figure 4. Signed distance map for an arbitrary polygonal line. The feasible region is characterized by the negative function values.

where $l = 1, \cdots, n_{\boldsymbol{\ell}_L}$, where $n_{\boldsymbol{\ell}_L}$ is the number of concave corner points of $\boldsymbol{\ell}_L$, and $m = 1, \cdots, n_{\boldsymbol{\ell}_R}$, where $n_{\boldsymbol{\ell}_R}$ is the number of concave corner points of $\boldsymbol{\ell}_R$. Note that the positive sign in Eqs. (16) implies that the corner points are excluded from the bounding envelopes of the B-spline curve. Consequently, the inequality constraints in Eqs. (15) and (16) ensure that the envelope of the B-spline stays inside the channel, as depicted in Fig. 5.

III.B. Smooth curve optimization

In this section we consider the problem of designing a smooth curve using a quartic B-spline, whose basis functions are computed from Eq. (2b), as degree four polynomials in terms of the knot parameter u. Hence, the quartic B-spline basis function preserves its continuity up to the third order derivative, thus resulting in the continuity of the derivative of the curvature. Without loss of generality, the knot parameter is selected as $u \in [0, 1]$, and the first and last knots have multiplicity 5, such that $u_0 = \cdots = u_4 = 0$ and $u_{m+1} = \cdots = u_{m+5} = 1$. Consequently, the B-spline curve will be clamped at, or pass through, the first and last control points.

We manipulate the $(m + 1)$ control points of a B-spline curve given by $\boldsymbol{b}_j = [b_j^1 \; b_j^2]^\mathsf{T}$ $(j = 0, \cdots, m)$, which have a direct influence on the shape of the curve. Because the shape of the curve is closely related to the given channel geometry that encloses the curve, the number of control points can initially be opted for the minimum number, by taking into account the complexity of the given channel geometry. The knot sequence is chosen arbitrary for non-decreasing numbers in the interval $(0, 1)$, so that the number of knots matches that of the control points. It should be noted that the number of knots and control points may be altered during the optimization process by inserting both knots and control points if the envelopes of the B-spline need to be refined.[8]

Two different performance indices are adopted to optimize curves in terms of attaining two distinct objectives: In order to keep the B-spline curve as close as possible to a straight line, for which $\Delta_2 \boldsymbol{b}_j = 0$, we employ the cost function

$$\mathcal{J}_1\Big(\{\boldsymbol{b}_j\}_{j=0}^{m}, \{u_k\}_{k=0}^{m+5}\Big) = \sum_{j=1}^{m-1} (\Delta_2 \boldsymbol{b}_j)^\mathsf{T} (\Delta_2 \boldsymbol{b}_j). \tag{17}$$

The performance index (17) implicitly minimizes the curvature variation of B-spline curve, thus resulting in a smooth B-spline curve. On the other hand, suppose that the arc length of the B-spline curve is approximately captured by the total length of the control polygon $\boldsymbol{\ell}(u)$. Hence, for the shortest path, we employ a cost

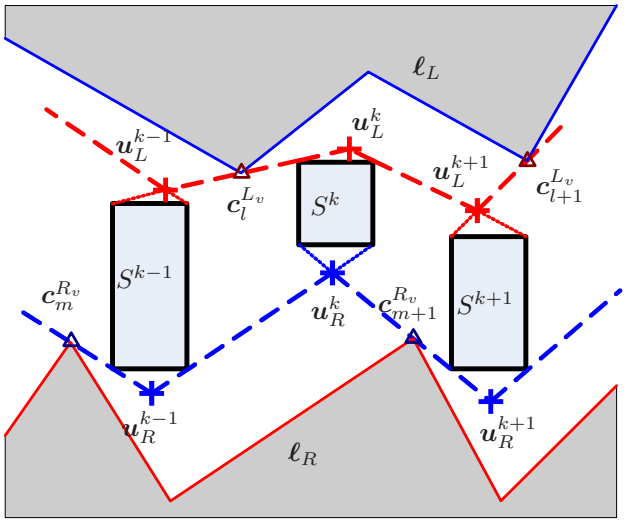


Figure 5. Geometric constraints formulation. The channel is given by two polylines $\boldsymbol{\ell}_L$ and $\boldsymbol{\ell}_R$, the envelope of the B-spline is drawn by the dashed lines, which is supposed to stay inside the channel.

function as the sum of the length of the piecewise control polygon, as follows

$$\mathcal{J}_2\Big(\{\boldsymbol{b}_j\}_{j=0}^{m}\Big) = \sum_{j=0}^{m-1} \|\boldsymbol{\ell}_j\|_2. \tag{18}$$

The constraints for the optimization problem are comprised of both equality and inequality constraints. The equality constraints stipulate boundary conditions for the position, tangent direction, and curvature at each end point at $u_0 = 0$ and $u_{m+5} = 1$ as follows,

$$\boldsymbol{b}(0) = \boldsymbol{p}_0 \ , \qquad \boldsymbol{b}(1) = \boldsymbol{p}_f, \tag{19a}$$
$$\psi(0) = \psi_0 \ , \qquad \psi(1) = \psi_f, \tag{19b}$$
$$\kappa(0) = \kappa_0 \ , \qquad \kappa(1) = \kappa_f, \tag{19c}$$

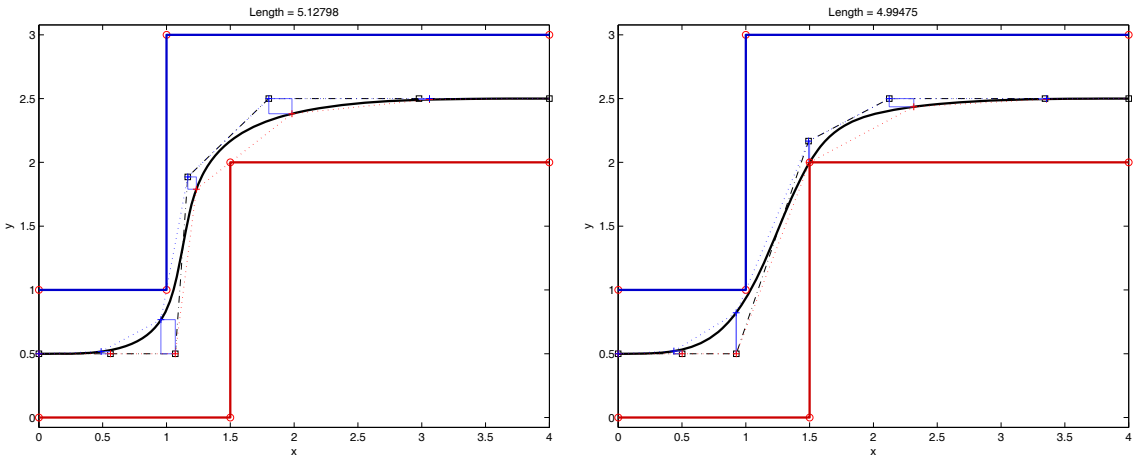
where $\psi(u)$ and $\kappa(u)$ are the tangent direction and the curvature of the B-spline curve at each knot u. The channel constraints in Eqs. (15) and (16) become the inequality constraints for the optimization problem.

Using the ingredients discussed so far, the path optimization problem can be stated as follows: Given a knot sequence $\{u_k\}_{k=0}^{m+5}$, two polygonal lines for the channel geometry, and boundary conditions for each end point, find a B-spline curve which minimizes the cost function in Eqs. (17) or (18), subject to the equality constraints in Eqs. (19) and the inequality constraints in Eqs. (15) and (16). We have used a standard SQP solver for this optimization problem.

Figure 6(a) shows the optimization result using the cost function in Eq. (17). The constructed quartic B-spline curve is drawn by a solid line, and the bounding envelopes are drawn by dashed and dashed-dot lines. The B-spline, as well as the envelopes, stay inside the specified channel polygon. For the case of the shortest path that involves the cost function in Eq. (18), Fig. 6(b) reveals that the computed B-spline curve is indeed shorter than the previous case.

IV. Path Templates for Different Channels

In this section we construct path templates for on-line path smoothing. The templates contain a set of planar B-spline curves, which will be regarded as local path segments to smooth a discrete path sequence. The obstacle-free discrete path sequence is provided by a high-level path planner, such as the multiresolution path planning algorithm proposed in Ref. [19]. The algorithm constructs an obstacle-free channel such that the discrete path sequence is represented by a series of square cells. For all probable channels that correspond to path sequences over a finite planning horizon, we solve for a set of B-spline curves via the path optimization discussed in the previous section. Different channel constraints and boundary conditions are imposed in each case, resulting in a set of smooth local path primitives that pass through the obstacle-free region.



(a) A smooth curve in the channel (b) A shortest curve in the channel

Figure 6. Two optimization results.

IV.A. Path rules within a finite horizon

Suppose the world environment $\mathcal{W} \subset \mathbb{R}^2$ is decomposed into a uniform cell representation which consists of square cells $c_{k,\ell}$ of dimension $1/2^J \times 1/2^J$. We adopt four-connectivity relationship between cells. A high-level path planner computes an optimal path as a sequence of cells from the current cell to the goal cell. The path sequence is written in a *path word*, by which transitions toward the North, South, East, and West directions between two cells is encoded by `N`, `S`, `E`, and `W`, respectively. We specify the range of interest within a four-cell horizon from the current cell, as shown in Fig. 7. If the goal cell is located outside the horizon,

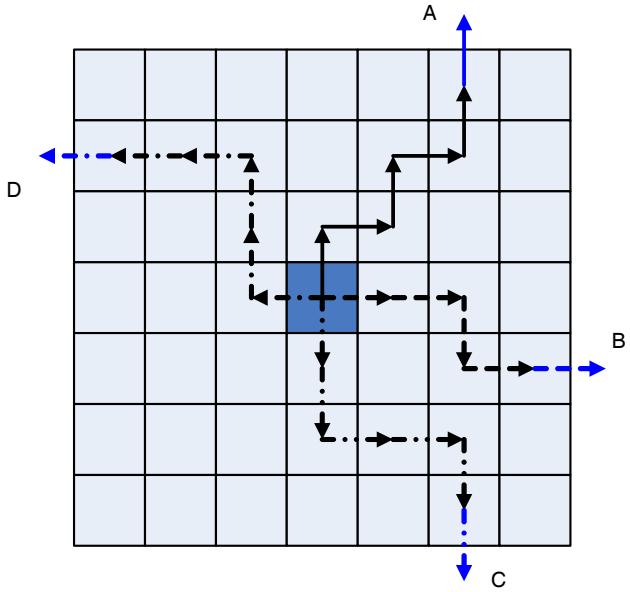


Figure 7. Examples of path sequences starting from the current cell at the center. We adopt four-connectivity between cells. The goal cell is located beyond the horizon. Possible path sequences are the path A, is written as `NENEN`$\cdots$, the path B, written as `EESE`$\cdots$, the path C, written as `SSEES`$\cdots$, and the path D, written as `WNNWW`$\cdots$.

then an optimal path sequence is computed in a such manner that the path eventually passes through one of the cells at the horizon boundary. Let a local path sequence from the current cell to reach the boundary cells be a *local path instance*. If the path is supposed to visit each cell only once, the number of all possible combinations of local path instances is finite. In addition, taking advantage of the symmetry about the x-axis (East direction) and y-axis (North direction), we investigate only local path instances restricted on one quadrant of the original 7×7 cell grid.

Without loss of generality, we may consider local path instances on the first quadrant, as shown in Fig. 8. It follows from the square cell geometry that we can also take advantage of the symmetry about the diagonal

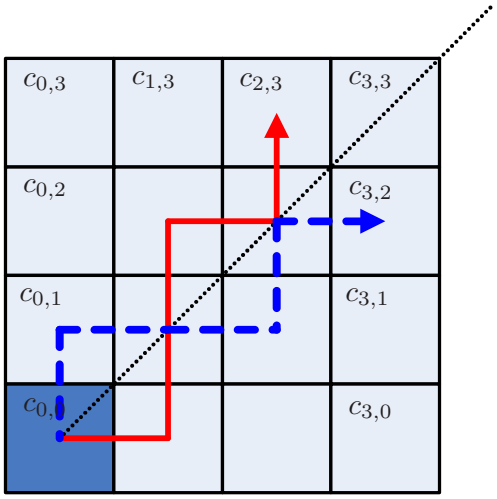


Figure 8. Local path instances in the first quadrant. From the additional symmetry about the diagonal axis, it is possible to transform the path instance drawn by a dashed line (`NEENE`) to the path instance drawn by a solid line (`ENNEN`). Path rules are given to determine several unique path instances to reach the cell at the top boundary.

axis. Hence, two path instances of `ENNEN` and `NEENE` are similar to each other. Thus, they are considered to be derived from the same local path instance. As a result, we only consider local path instances that start from the current cell and end at one of the top boundary cells $c_{0,3}$, $c_{1,3}$, $c_{2,3}$, and $c_{3,3}$. By applying the symmetric operations along the horizontal, vertical, and diagonal axes, any local path instances can be induced from these path instances. In order to find all possible combinations of path sequences to reach these cells, we describe the necessary path rules to determine a unique local path instance. These are listed as follows:

1. *(Terminal conditions)* Suppose a local path instance is restricted inside the first quadrant, that is, it never goes outside the horizon before it reaches a terminal cell on the top boundary. The terminal cell should be one of the top boundary cells, except the cell $c_{3,3}$. The reason for this is attributed to the four-connectivity between cells: The path instance that reaches $c_{3,3}$ necessarily passes through the adjacent boundary cell, either $c_{2,3}$ or $c_{3,2}$. Thus, we regard the path instances to the cell $c_{3,3}$ as a subset of the path instances to $c_{2,3}$, and exclude those from further consideration. In order to come up with a path sequence that reaches one of the boundary cells, a corresponding path word should have a certain number of occurrences of `N`, `S`, `E`, and `W`, satisfying the following conditions,

$$\sum \left(\#\mathtt{N} - \#\mathtt{S}\right) = 3,$$

$$\sum \left(\#\mathtt{E} - \#\mathtt{W}\right) = \begin{cases} 0 & \text{for } c_{0,3}, \\ 1 & \text{for } c_{1,3}, \\ 2 & \text{for } c_{2,3}. \end{cases}$$

2. *(Self-avoiding path)* The path must visit each cell exactly once, never intersecting itself. From this rule, we explicitly prevent pathological cases such as a cyclic loop in the path templates. This type of path can be described by a self-avoiding walk[20] on a 4×4 cell grid. The total number of self-avoiding walks on an $m \times n$ grid, which starts from a corner and ends at the opposite corner by only horizontal and vertical steps, is computed using a recurrence relation.[21] Table 1 gives the first few numbers of such walks for small m and n. Similarly, the number of candidates of self-avoiding paths from the current cell $c_{0,0}$ to the top boundary cells can be calculated from Table 1. Among these candidates, only a certain number of self-avoiding paths will be considered in the path templates.

3. *(Path optimality)* The path planning algorithm provides an optimal path sequence. The optimal path sequence is calculated so as to minimize the accumulated transition cost from the current node to the

Table 1. Number of self-avoiding walks on an $m \times n$ grid

m, n	2	3	4	5
2	2	4	8	16
3	4	12	38	125
4	8	38	184	976
5	16	125	976	8512

goal node. Typically, a directed edge cost is assigned to each transition to `N`, `S`, `E`, and `W` cells, taking into account the cost associated with the cells as follows,

$$\mathcal{J}(u, v) = f(v) + \alpha g(u, v), \tag{20}$$

where, $f(v)$ is a positive obstacle cost associated with the target cell v, $g(u, v)$ is (Euclidean) distance cost between u and v, and $\alpha \geq 0$ is a weight constant. Consider for example, the path corresponding to the path word `ENW`, which represents the transition among four cells u, v, w, and z in such an order. The accumulated cost for this transition is computed by

$$\mathcal{J}(u, v) + \mathcal{J}(v, w) + \mathcal{J}(w, z) > \mathcal{J}(u, z), \tag{21}$$

which turns out to be greater than the direct transition cost from u to z by a single path sequence `N`. It follows that the transition `ENW` is not an optimal, and neither are `ESW`, `NES`, `NWS`, etc. Consequently, we disregard any non-optimal path sequence when investigating the candidates of self-avoiding paths.

When establishing the previous path rules, we assumed that the local path instance necessarily ends up at one of the top boundary cells. In certain cases, however, the path sequence may be given in such a way that it crosses the quadrant boundary. In other words, the path sequence that starts from the current cell at the center of grid, is comprised of cells belonging to more than one quadrant. If this is the case, we can infer that the path sequence will finally exit the finite horizon after passing through at least two quadrants, which makes it difficult to take advantage of the symmetry of the templates. In particular, if one wants to consider all possibilities of inter-quadrant transitions, the number of templates will increase, thus losing the benefit of small-sized templates. In order to retain the symmetry of the templates, we consider additional cells between the quadrants as additional terminal cells. Applying the path rules to a 7×7 cell grid (thus 4×4 cell grid for the first quadrant), only the cell $c_{0,2}$ (see Fig. 8) can be considered as an additional terminal cell. The other cells on the $y-$axis cannot be terminal cells since the local path instances reaching them would conflict with the path rules discussed above. Any local path instance starting from the center cell to $c_{0,2}$, satisfying the path rules, is appended to the path templates.

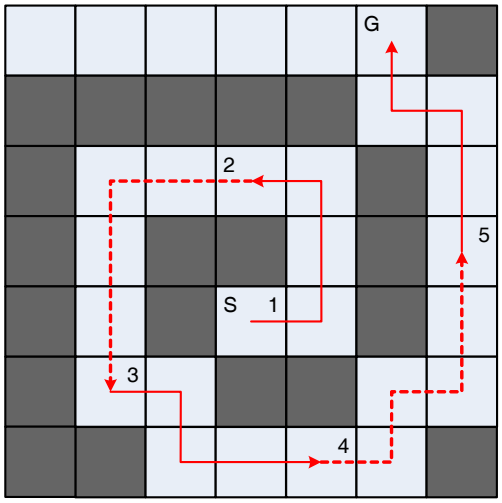
Following the previous discussion on the path rules, we finally come up with the path templates consisting of local path instances on the first quadrant, which are summarized in Table 2. Figure 9 shows an example of utilizing the path templates for a given path sequence. The starting cell is located at the center, while the path sequence was computed avoiding the shaded obstacle cells. For this example, in order to reach the goal cell, five local path instances are required to represent the overall path sequence. In order to search for the path templates, we first get the actual path words that correspond to each of the local path sequences. Then we associate the path words to the first quadrant by using the required symmetry operations, that is, horizontal (H), vertical (V), or diagonal (D) reflections. We can then identify the corresponding entry in the path templates. The table in Fig. 9 illustrates that five different path templates are utilized in order to construct the composite path from the starting cell to the goal cell.

IV.B. Construction of B-spline path templates

The B-spline path templates are composed of a set of B-spline curves, which are placed inside the channels. By the assumption of the high resolution representation of $\mathcal{W}$, the cells in the optimal path sequence are feasible for the agent to fly safely. Thus, a channel that corresponds to an optimal path sequence over a finite planning horizon is determined in a manner such that the outmost border lines of each cell yield a channel polygon. The channel polygon is then divided by two `Left`, `Right` polylines, which serve as channel

Table 2. Path templates for local path instances on the first quadrant.

Destination cell	Path words			
$c_{0,3}$	NNN	ENNWN	EENNWWN	
$c_{1,3}$	NNEN	NENN	ENNN	EENNWN
$c_{2,3}$	NNEEN	NENEN	ENNEN	
	NEENN	ENENN	EENNN	
$c_{0,2}$	ENNW	EENNWW		



Path #	Path words	Templates	Operations
1	ENNW	ENNW	-
2	WWSSS	EENNN	H, V
3	ESEE	NENN	H, D
4	ENENN	ENENN	-
5	NNWN	NNEN	V

Figure 9. Example incorporating the path templates on a complex path sequence. Five local path instances are connected to one another in order to reach the goal cell. The actual path words are recovered from the path templates with the corresponding symmetry operations of the horizontal (H), vertical (V), or diagonal (D) reflections.

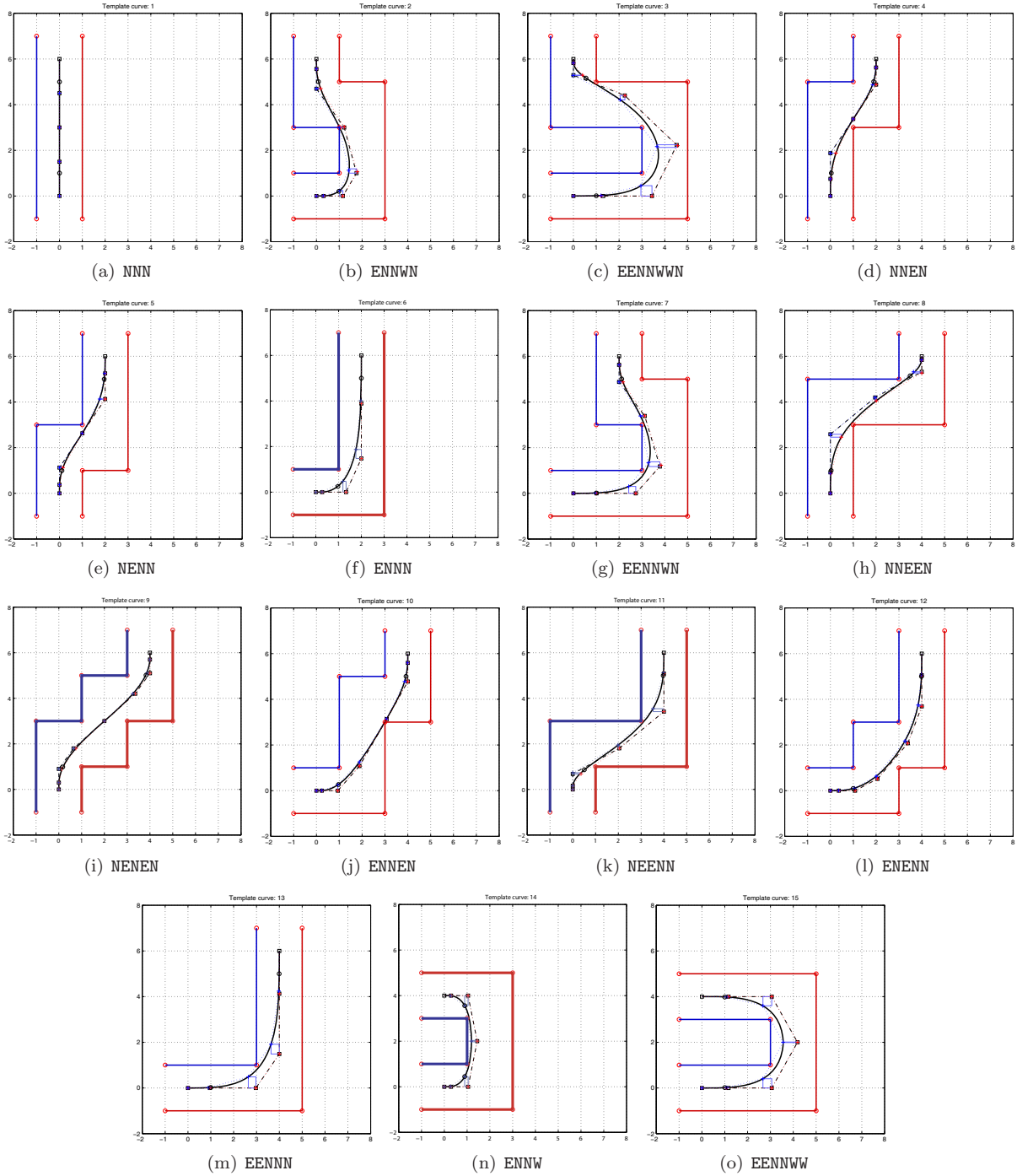
constrains in the following optimization. For the sake of convenience, the boundary conditions of each B-spline curve are imposed, such that the B-spline curve starts from the center of the first cell of the local path instance and ends at the center of the last cell of the local path instance. The tangent directions at each end of the curve are imposed such that they direct toward the center of the next adjacent cell, whereas the curvature values are set all to be zero. Subsequently, we solve the optimization problem discussed in Section III using the composite cost using Eqs. (18) and (17), subject to a set of channel constraints and the associated boundary conditions. Figure 10 shows the results of the optimization for the B-spline path templates corresponding to each local path instance shown in Table 2.

V. On-line Path Smoothing Algorithm

In this section, we present an on-line path smoothing algorithm incorporating the B-spline path templates. Given a discrete path sequence from a high-level path planner, each instance of the B-spline path templates becomes a part of the smooth path. Hence, the on-line path smoothing algorithm connects these path segments resulting in a smooth composite path.

V.A. Stitching the path segments

Two B-spline curves chosen from the library of path templates intersect each other at a single junction point. Due to the different boundary conditions at each end of these curves, it follows that a discontinuity of the tangent directions occurs at the junction points. Hence, in order to get a smooth path segment over two consecutive B-spline curves, we propose to stitch them with a *transient* B-spline curve, while preserving continuity over the distinct B-spline curves. To this end, let $\boldsymbol{p}_a$ and $\boldsymbol{p}_b$ be the points on the first and the second B-spline curves, respectively. These points are chosen nearly at the end points of corresponding curves, such that not only the transient B-spline curve passes through these points smoothly, but also the



(a) NNN (b) ENNWN (c) EENNWWN (d) NNEN

(e) NENN (f) ENNN (g) EENNWN (h) NNEEN

(i) NENEN (j) ENNEN (k) NEENN (l) ENENN

(m) EENNN (n) ENNW (o) EENNWW

Figure 10. B-spline path templates from the channel optimization results. Each plot corresponds to the local path instance in Table 2.

overall smoothed curve remains close to the B-spline curves in the templates. Suppose the transient curve intersects these points at its own end points. Then it follows that preserving smoothness over successive B-spline curves is ensured by imposing smoothness at these points. Hence, the transient B-spline curve, which stitches two distinct B-spline curves, should satisfy the continuity conditions with respect to the position, tangent angle, and curvature at each end in order to guarantee a smooth transition between the two path segments.

The minimum order of a B-spline curve that satisfies the continuity conditions described above is four. Subsequently, we employ a transient cubic B-spline curve, which is defined on a fixed clamped knot vector, $u = [0, 0, 0, 0, 1/3, 2/3, 1, 1, 1, 1]$. Hence, the unknown parameters for determining this transient B-spline curve are six control points. The first three control points are related to the boundary conditions associated with the first B-spline curve, and the rest are related to the boundary conditions associated with the second B-spline curve. Suppose the boundary conditions at the intersection point with the leading B-spline curve are given as follows. The intersection point is specified by $\boldsymbol{p}_a = [x_a, y_a]$, the tangent angle at $\boldsymbol{p}_a$ is given by ψ_a, and the curvature is given by κ_a. Let $\boldsymbol{p}_i = [x_i, y_i]$ $(i = 0, \cdots, 5)$ denote the control points. Then we impose the following boundary conditions,

$$x_a = x_0, \tag{22a}$$
$$y_a = y_0, \tag{22b}$$
$$\psi_a = \tan^{-1}\left(\frac{y_0'}{x_0'}\right). \tag{22c}$$

where, x_0' and y_0' are the derivatives with respect to the knot parameter u at the control point $\boldsymbol{p}_0$. Note that, with the chosen clamped knot vector above, the cubic B-spline basis functions evaluated at the knot $u = 0$ are computed as follows,

	b_0	b_1	b_2	b_3	$\cdots$
$N_i^3(0)$	1	0	0	0	
$N_i^{3\prime}(0)$	-9	9	0	0	$\cdots$
$N_i^{3\prime\prime}(0)$	54	-81	27	0	

(23)

Hence, the first derivative at $\boldsymbol{p}_0$ evaluated with the knot value $u = 0$ is computed by

$$x_0' = -9x_0 + 9x_1, \tag{24a}$$
$$y_0' = -9y_0 + 9y_1. \tag{24b}$$

From Eqs. (22c) and (24), it follows that

$$x_1 = x_a + R_a \cos\psi_a, \tag{25a}$$
$$y_1 = y_a + R_a \sin\psi_a, \tag{25b}$$

where R_a is the distance between $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$, which we choose as a design parameter. In addition, we impose the curvature boundary condition,

$$\kappa_a = \frac{x_0{}'y_0{}'' - y_0{}'x_0{}''}{\left(x_0{}'^2 + y_0{}'^2\right)^{3/2}}, \tag{26}$$

where, $x_0{}''$ and $y_0{}''$ are the second derivatives at $\boldsymbol{p}_0$, evaluated with the knot value $u = 0$. Utilizing Eqs. (23) and (24), and after some algebraic manipulation, results in the following expression with respect to x_2 and y_2,

$$-x_2 \sin\psi_a + y_2 \cos\psi_a = 3R_a^2\kappa_a + y_a \cos\psi_a - x_a \sin\psi_a. \tag{27}$$

In order to solve for x_2 and y_2, we adopt an additional equation in terms of x_2 and y_2, which determines a unique solution of x_2 and y_2. Suppose $\boldsymbol{p}_2^{\perp}$ is the point at the distance $2R_a$ along the line segment $\overline{\boldsymbol{p}_0\boldsymbol{p}_1}$ from $\boldsymbol{p}_0$ (see Fig. 11). The control point $\boldsymbol{p}_2$ can then be chosen uniquely by imposing the additional condition that the projection of $\boldsymbol{p}_2$ onto the line segment $\overline{\boldsymbol{p}_0\boldsymbol{p}_1}$ ends up at the point $\boldsymbol{p}_2^{\perp}$. After some algebraic manipulations, we obtain the following equation,

$$x_2 \cos\psi_a + y_2 \sin\psi_a = 2R + x_a \cos\psi_a + y_a\psi_a. \tag{28}$$

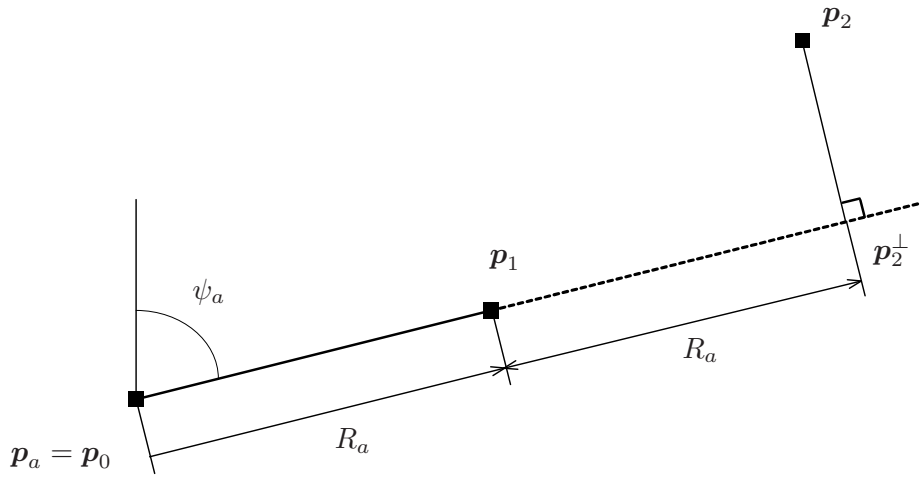


Figure 11. Determine the unique p_2 in terms of p_0 and p_1 in conjunction with the design parameter R_a.

Similarly, we impose the boundary conditions at the end of the transient B-spline curve. The cubic B-spline basis functions are evaluated at the knot $u = 1$ as follows,

$$\begin{array}{c|cccccc} & \cdots & x_2 & x_3 & x_4 & x_5 \\ \hline N_i^3(1) & & 0 & 0 & 0 & 1 \\ N_i^{3\prime}(1) & \cdots & 0 & 0 & -9 & 9 \\ N_i^{3\prime\prime}(1) & & 0 & 27 & -81 & 54 \end{array} \tag{29}$$

Imposing the given boundary conditions of the intersection point by $\boldsymbol{p}_b = [x_b, y_b]$, the tangent angle at $\boldsymbol{p}_b$ by ψ_b, and the curvature by κ_b, we obtain the following formulas that uniquely determine the rest of the three control points,

$$x_5 = x_b, \tag{30a}$$
$$y_5 = y_b, \tag{30b}$$
$$x_4 = x_b - R_b \cos\psi_b, \tag{30c}$$
$$y_1 = y_b - R_b \sin\psi_b, \tag{30d}$$
$$-x_3 \sin\psi_b + y_3 \cos\psi_b = 3R_b^2\kappa_b + y_b \cos\psi_b - x_b \sin\psi_b, \tag{30e}$$
$$x_3 \cos\psi_b + y_3 \sin\psi_b = -2R_b + x_b \cos\psi_b + y_b \sin\psi_b. \tag{30f}$$

Figure 12 shows an example of stitching the two B-spline curves derived from the path templates by a transient cubic B-spline.

We note in passing that one can avoid altogether the use of transient B-splines for connecting two regular B-spline curves by using path templates of one or more overlapping cells, and by interpolating the corresponding B-splines inside the overlapping cell(s). We will not elaborate on this implementation in this paper.

VI. Simulation Results of the Path Smoothing Algorithm

In this section we present simulation results of the on-line path smoothing in conjunction with the $\mathcal{D}^*$-lite path planning algorithm. The $\mathcal{D}^*$-lite (Dynamic $\mathcal{A}^*$-lite) algorithm was proposed by Koenig and Likhachev[22] for path planning in unknown or partially known environment. The algorithm reuses information from the previous search to find the solution at the next iteration much faster than solving each iteration from scratch.

It is assumed that the agent navigates over an unknown environment, while updating the map with the information gathered from a proximity sensor. The world data is given by a 256× 256 units (pixels) map. We adopt a uniform cell decomposition of cell size 8×8 pixels. The range of the proximity sensor is chosen to be $r = 28$, thus resulting in the finite horizon window by a 7×7 square cell grids.

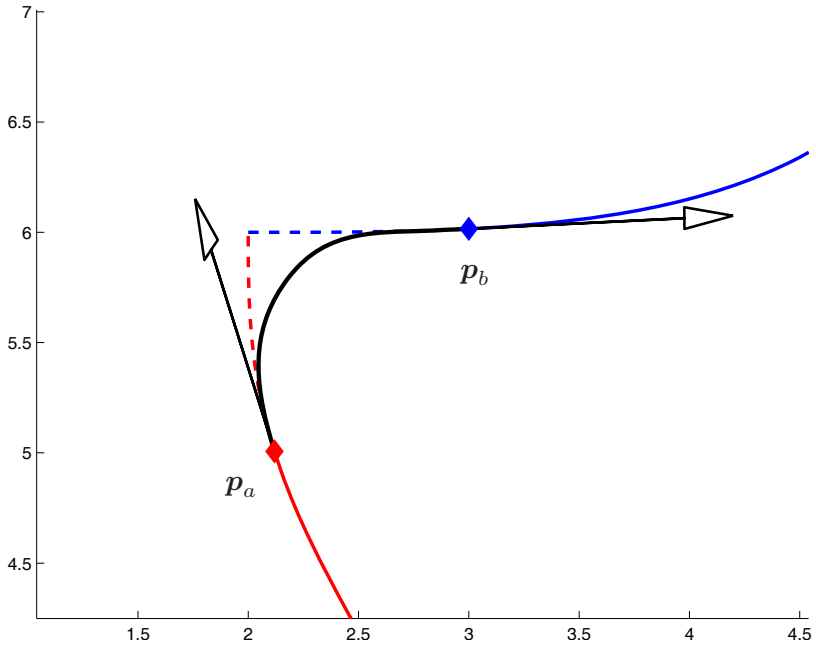


Figure 12. Example of stitching the two B-spline curves with a transient cubic B-spline curve.

The results from the on-line path smoothing algorithm combined with the $\mathcal{D}^*$-lite path planning are shown in Fig. 13. Specifically, Fig. 13 shows the evolution of the path at different time steps, as the agent moves to the final destination. At each step, the best proposed path is drawn by a dashed-dot line and the actual path generated by the algorithm is drawn by a solid line. A local channel is drawn by thin polylines, which corresponds to the discrete path sequence from the $\mathcal{D}^*$-lite algorithm. Accordingly, a smooth path segment is obtained from the B-spline path templates. Whenever the $\mathcal{D}^*$-lite algorithm updates with a (possibly new) path sequence, as the agent approaches close to the end of each path segment, the on-line path smoothing algorithm proceeds to stitch the previous path segment with the newly obtained path segment by a transient B-spline curve. This process repeats until the UAV reaches the final destination, as shown in Fig. 13(c). For on-board implementation using a micro-controller, the proposed algorithm utilizes small computer memory, since the B-spline path templates are stored using only a small number of control points and knot sequences. Furthermore, by incorporating the path templates over a finite horizon that have been computed off-line, the proposed algorithm generates a smooth path quickly, while ensuring the smoothness of the entire path. In particular, the algorithm is computationally efficient in the sense that it can yield close-to-optimal paths when the environment is changing, when used in conjunction with the high-level path planner that accommodates for the changes. The algorithm has been implemented on a small micro-controller for a hierarchical path planning and control algorithm of a small UAV using the hardware-in-the-loop (HIL) simulation. The results from these tests can be found in Ref. [23].

VII. Conclusions

We have presented an on-line path smoothing algorithm that incorporates path templates for generating a smooth path derived from a high-level path planner. The path templates are comprised of a set of B-spline curves, which have been obtained from an off-line optimization step, such that each path instance stays inside the prescribed channel, hence avoiding obstacles. In conjunction with the high-level path planner, the on-line implementation of the proposed algorithm finds the corresponding local path segments and stitches them together, while preserving the smoothness of the composite curve. Simulation results in conjunction with the discrete $\mathcal{D}^*$-lite path planning algorithm validate the effectiveness of the proposed algorithm. The algorithm has minimal on-line computational cost and provides a complete solution to the obstacle-free path

generation problem that is suitable for real-time implementation for small UAVs.

Acknowledgment: Partial support for this work has been provided by NSF award CMS-0510259.

References

[1]Kanayama, Y. and Hartman, B. I., "Smooth Local Path Planning for Autonomous Vehicles," *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 3, May 1989, pp. 1265–1270.

[2]Scheuer, A. and Laugier, C., "Planning Sub-Optimal and Continuous-Curvature Paths for Car-Like Robots," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Victoria, B. C., Canada, Oct. 1998, pp. 25–31.

[3]Anderson, E. P., Beard, R. W., and McLain, T. W., "Real-Time Dynamic Trajectory Smoothing for Unmanned Air Vehicles," *IEEE Transactions on Control Systems Technology*, Vol. 13, No. 3, May 2005, pp. 471–477.

[4]Judd, K. B. and McLain, T. W., "Spline Based Path Planning for Unmanned Air Vehicles," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Montreal, Canada, Aug. 2001, AIAA 2001-4238.

[5]Vázquez G., B., Sossa A., J. H., and Díaz-de-León S., J. L., "Auto Guided Vehicle Control Using Expanded Time B-splines," *IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 3, San Antonio, TX, Oct. 1994, pp. 2786–2791.

[6]Berglund, T., Jonsson, H., and Söderkvist, I., "An Obstacle-avoiding Minimum Variation B-Spline Problem," *Proceedings of International Conference on Geometric Modeling and Graphics*, July 2003, pp. 156–161.

[7]Dyllong, E. and Visioli, A., "Planning and Real-time Modifications of a Trajectory Using Spline Techniques," *Robotica*, Vol. 21, 2003, pp. 475–482.

[8]Lutterkort, D. and Peters, J., "Smooth Paths in a Polygonal Channel," *Proceedings of the fifteenth Annual Symposium on Computational Geometry*, Miami Beach, FL, 1999, pp. 316–321.

[9]Lutterkort, D. and Peters, J., "Tight Linear Envelopes for Splines ," *Numerische Mathematik*, Vol. 89, No. 4, Oct. 2001, pp. 735–748.

[10]McLain, T., Chandler, P., and Pachter, M., "A Decomposition Strategy for Optimal Coordination of Unmanned Air Vehicles," *Proceedings of the American Control Conference*, Chicago, IL, 2000, pp. 369–373.

[11]Beard, R. W., McLain, T. W., Goodrich, M., and Anderson, E. P., "Coordinated Target Assignment and Intercept for Unmanned Air Vehicles," *IEEE Transactions on Robotics and Automation*, Vol. 18, Dec. 2002, pp. 911–922.

[12]McLain, T. W. and Beard, R. W., "Coordination Variables, Coordination Functions, and Cooperative Timing Missions," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 1, 2005, pp. 150–161.

[13]de Boor, C., *A Practical Guide to Splines*, Vol. 27 of *Applied Mathematical Sciences*, Springer-Verlag, New York, NY, 1978.

[14]Piegl, L. and Tiller, W., *The NURBS Book*, Monographs in Visual Communication, Springer-Verlag, Berlin Heidelberg, 2nd ed., 1997.

[15]Greville, T. N. E., *Theory and Applications of Spline Functions*, Academinc press, New York, NY, 1968.

[16]Nairn, D., Peters, J., and Lutterkort, D., "Sharp, Quantitative Bounds on the Distance Between a Polynomial Piece and its Bézier Control Polygon," *Computer Aided Geometric Design*, Vol. 16, 1999, pp. 613–631.

[17]Lutterkort, D. and Peters, J., "The Distance Between a Uniform (B-)Spline and Its Control Polygon," Technical Report TR-98-013, The Department of Computer and Information Science and Engineering, University of Florida, Sept. 1998.

[18]Berglund, T., Jonsson, H., and Söderkvist, I., "An Obstacle-Avoiding Minimum Variation B-Spline Problem," *Proceedings of 2003 International Conference on Geometric Modeling and Graphics*, July 2003, pp. 156–161.

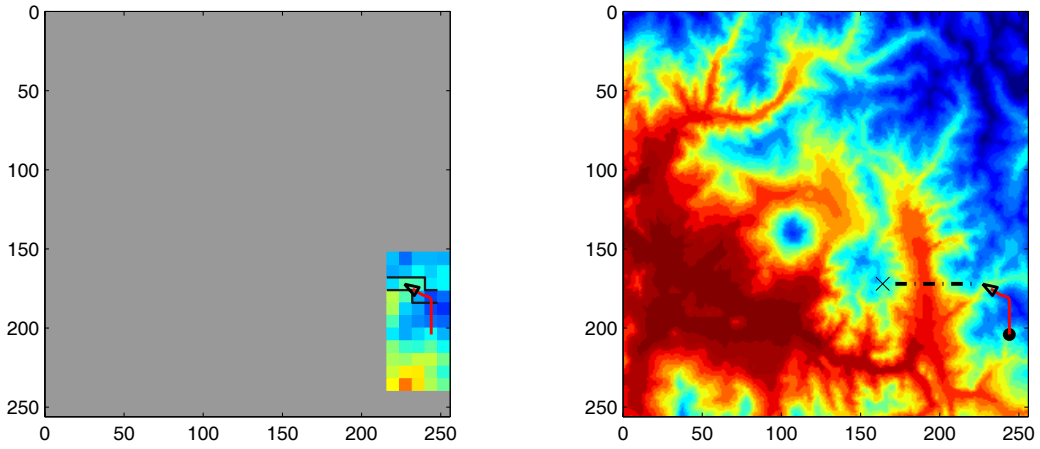
[19]Jung, D. and Tsiotras, P., "Multiresolution On-Line Path Planning for Small Unmanned Aerial Vehicles," *American Control Conference*, Seattle, WA, June 2008, pp. 2744–2749.

[20]Madras, N. and Slade, G., *The Self-Avoiding Walk*, Birkhäuser, Boston, MA, 1993.

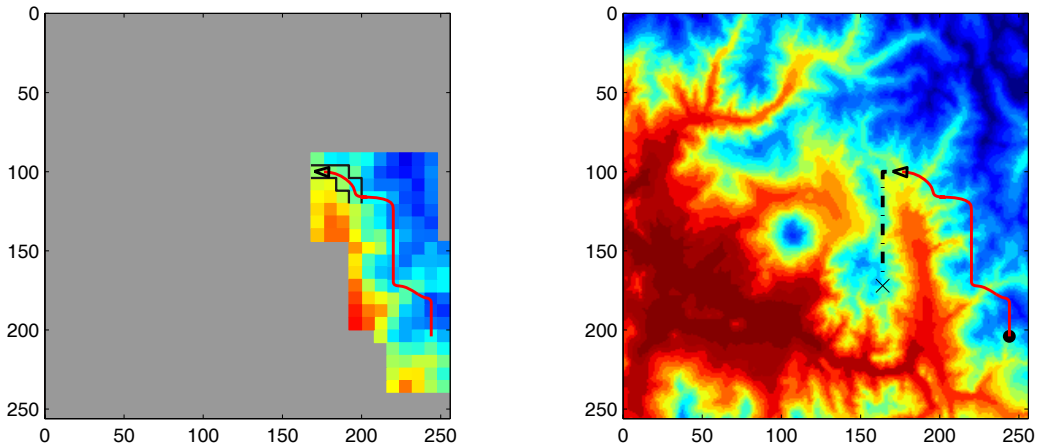
[21]Finch, S. R., *Mathematical Constants*, Cambridge University Press, Cambridge, England, 2003, pp. 331–339.

[22]Koenig, S. and Likhachev, M., "$\mathcal{D}^*$ Lite," *Proceedings of the National Conference of Artificial Intelligence*, 2002, pp. 476–483.

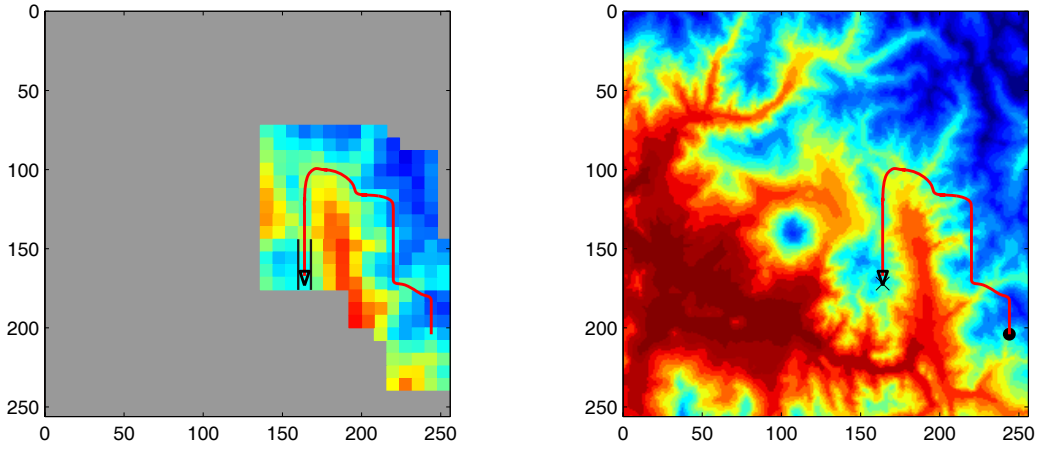
[23]Jung, D., *Hierarchical Path Planning and Control of a Small Fixed-Wing UAV: Theory and Experimental Validation*, Ph.D. thesis, Georgia Institute of Technology, Atlanta, GA, Dec. 2007.



(a) $t = t_{14}$



(b) $t = t_{30}$



(c) $t = t_f$

Figure 13. On-line path smoothing in conjunction with replanning using the $\mathcal{D}^*$-lite algorithm. Dashed-dot lines represent the currently tentative optimal path obtained from the $\mathcal{D}^*$-lite algorithm, based on the distance cost outside the finite horizon window. Actual reference path to be followed by the agent using the B-spline path templates is represented by a solid line.