Proceedings of the 2006 IEEE/RSJ
International Conference on Intelligent Robots and Systems
October 9 - 15, 2006, Beijing, China

A Framework for Automatic Deployment of Robots in 2D and 3D Environments

Marius Kloetzer and Calin Belta
Center for Information and Systems Engineering
Boston University
Boston, MA 02446, USA
Email: {kmarius,cbelta}@bu.edu

Abstract— **We present a computational framework for automatic deployment of robots in 2D and 3D rectangular environments with polytopal obstacles. The results are derived for polytopal robots that can only translate with velocities restricted to polyhedral sets. Our approach consists of three steps: (1) constructing a discrete representation of the problem by using hierarchical partitions in the form of quad-trees and oct-trees, (2) planning the motion in the finite dimensional quotient produced by the partition, and (3) generating provably correct robot feedback control laws by constructing a hybrid system. Given the environment and robot geometry and constraints, generation of control laws is completely automated. The computation consists of polyhedral operations and searches on graphs.**

I. Introduction

Robot motion planning and control are fundamental problems that received a lot of attention. Some works on this topic focus on environment complexity, while assuming that the robot is fully actuated with no control bounds, and includes approaches based on Voronoi diagrams, visibility graphs, freeway methods, potential fields [1], navigation functions [2], cellular decompositions, and probabilistic roadmaps [3]. Others focus on the detailed dynamics of the robot, assuming simple environments. Some of these approaches are differential geometric [4], some exploit concepts such as flatness [5], while others use input parameterizations [6], or discontinuous control laws [7]. Among works combining discrete algorithms with continuous methods, generation of smooth curves with curvature guarantees is linked to a probabilistic roadmap planner in [8], while a method dealing with non-holonomic constraints was developed in [9]. Polygonal partitions and triangulations followed by assignment of vector fields in the regions produced by the partition were considered in [10], [11], [12].

In this paper, we present a method for planning and control of a fully actuated polytopal robot that can translate with speeds restricted to a polyhedral set in a 2D or 3D rectangular environment with polytopal obstacles. With a bit of conservatism, the method can be extended to rotating and translating under-actuated robots (such as unicycles) [13]. The main idea behind our method is to use *partitions to capture the complexity of the environment* and *discrete abstractions of hybrid systems to generate provably correct robot control laws*. The computational framework consists of three steps. (1) A hierarchical finite dimensional discrete representation of the obstacle - free space is constructed by using rectangular partitions in the form of quad-trees (for 2D case) and oct-trees (for 3D case) in a modified space where the robot is reduced to a point, the environment boundaries are shrunk, and the obstacles are enlarged. (2) A string is generated in the language of the discrete system, as solution of a path finding problem on a graph. (3) Provably correct robot control laws are designed by constructing a hybrid system whose discrete states (locations) together with their allowed transitions will correspond to the generated string. We implemented this method as a user-friendly software package [14].

This work is closely related to [10], [11], [12]. In [10], the authors consider a polygonal partition of a planar configuration space and assign vector fields so that initial states in each polygon can only flow to a neighbor through the corresponding common facet. The vector fields are defined as gradients of scalar functions determined as solutions of Laplace's equation. A computationally more attractive approach is proposed in [11], where the assignment of vector fields in triangulated polygons requires operations on polyhedral sets only, and arbitrary polyhedral control bounds can be satisfied. The computational framework of [11] is also used in [12], where paths for the discretized motion planning problem are found by temporal logic model checking.

As in [11], [12], our framework is fully algorithmic and the generation of vector fields is based on operations on polyhedral sets. However, we use rectangulations as opposed to triangulations, and take advantage of efficient hierarchical representations of partitions in the form of quad-trees and oct-trees [15]. The construction of the vector fields in this paper is inspired from [16], where sufficient conditions for the existence of a multi-affine vector field driving all states in a rectangle through an exit facet and making a rectangle an invariant were derived for Euclidean spaces of arbitrary dimension. Here, we focus on two and three dimensional spaces and are able to derive *necessary and sufficient conditions*. Moreover, we derive a controller driving all initial states in a rectangle to an arbitrary state inside it.

II. Problem formulation

Consider a rectangular environment $\mathcal{E} \subset \mathbb{R}^N$, $N \in \{2, 3\}$ and a finite set of $n \in \mathbb{N}$ possibly overlapping obstacles

1-4244-0259-X/06/$20.00 ©2006 IEEE

modelled as convex polytopes $O^i \subseteq \mathcal{E}$, $i = 1, \ldots, n$. Let $\{F\}$ denote a world frame fixed to the environment. Let $\mathcal{R}$ denote a polytopal robot free to translate in $\mathcal{E}$ and $x \in \mathcal{E}$ be the coordinates of a representative point (observable) of $\mathcal{R}$ in $\{F\}$. We assume that the velocity of the point x can be directly controlled, *i.e.,* we assume that the dynamics of the robot are described by

$$\dot{x} = u,\ u \in U \tag{1}$$

where u is the control input and U is a polyhedral subset of $\mathbb{R}^N$, $N = 2, 3$ capturing the robot control constraints. A trajectory $x(t)$ of the robot is called *feasible* if it is contained in the environment, it does not intersect the obstacles, and the corresponding velocity is allowed, *i.e.,* $\mathcal{R} \subset \mathcal{E}$, $\mathcal{R} \bigcap (\bigcup_{i=1}^{n} O^i) = \emptyset$, and $\dot{x} = u \in U$.

Problem 1: Given $\mathcal{E}$, O^i, $i = 1, \ldots, n$, $\mathcal{R}$, U, and two feasible positions $x_0, x_f \in \mathcal{E}$, determine whether there exists a feasible trajectory $x(t)$ of the robot from x_0 to x_f. If such trajectories exist, determine a continuous feedback control law $u = f(x) \in U$ producing a trajectory of robot from x_0 to x_f.

If there exist more trajectories satisfying Problem 1, we choose one for which the distance travelled by the robot is minimized and, during its motion, the robot does not approach the obstacles too much. A minimal trajectory should not be understood in the sense of an interpolating geodesic determined by a given metric capturing the geometry of the robot and the environment. Such a problem is computationally very expensive. In our representation, a minimal trajectory will correspond to a minimal path in a finite graph which is a discrete representation of our problem. The optimality criterion that we use is described in Section IV-B.

To illustrate our approach, throughout this paper we consider the example in Fig. 1, which shows the initial and final positions of a robot modelled as a square with side 1 in a planar rectangular environment with polyhedral obstacles. The robot observable is assumed to be its centroid and the control bounds are assumed to be $U = [-2, 2] \times [-2, 2]$.

III. Definitions and approach

In this paper, we provide a solution to Problem 1 that is algorithmic, fully automated, computationally efficient, and robust with respect to knowledge of environment, obstacles, and robot position. We use *iterative partitions* to capture the complexity of the environment and the quotients produced by such partitions to specify and solve motion tasks in a finite dimensional discrete (and, therefore, decidable) world. We then use *discrete abstractions of hybrid systems* to generate provably correct robot control laws implementing the specifications. In the rest of this section, we give the necessary definitions and outline these ideas.

a) Constructing a finite discrete representation: Using operations on polyhedral sets, we first construct a new problem, where the robot $\mathcal{R}$ is reduced to its reference point x, the boundaries of the environment $\mathcal{E}$ are shrunk, and the obstacles O^i are enlarged. We then construct a hierarchical representation of the free space of the robot by using rectangular partitions and

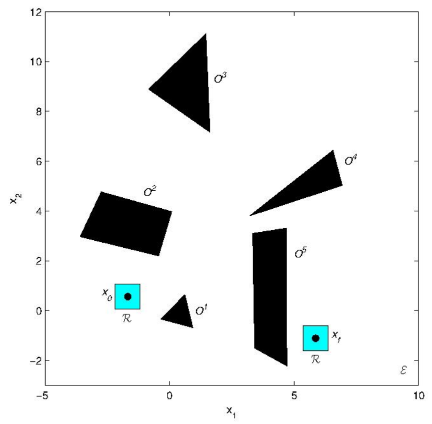


Fig. 1. Illustrative case study: initial (left) and final (right) positions of a square robot in a 2D rectangular environment with polyhedral obstacles. The observable is the centroid of the robot.

2^N - trees. Let $L = \{l_1, l_2, \ldots, l_{|L|}\}$ be a finite set of symbols labelling obstacle - free rectangles produced by such a partition ($|\cdot|$ denotes the cardinality of a set). To capture neighboring relations among obstacle - free rectangles, we construct a graph

$$G = (L, t) \tag{2}$$

where the edge set $t \subset L \times L$ denotes a symmetric adjacency relation between the corresponding rectangles. Details are included in Section IV-A and a graphical illustration of these ideas is given in Fig. 2.

b) Specifying tasks and generating solutions on the discrete representation: The graph in equation (2) is our framework for algorithmic planning and control. In other words, we will abstract a robot motion to a sequence of adjacent obstacle - free rectangular cells. Formally, we define the *language* $\mathcal{L}(G)$ of the graph G as the set of all its strings

$$\begin{aligned} \mathcal{L}(G) = \{s = (l_{i_1}, l_{i_2}, \ldots, l_{i_m}) \mid l_{i_j} \in L, \\ (l_{i_j}, l_{i_{j+1}}) \in t,\ j = 1, \ldots, m-1\} \end{aligned} \tag{3}$$

with $i_j \in \{1, \ldots, |L|\}$, $j = 1, \ldots, m$, $m \geq 1$. In this framework, a robot motion is a string in $\mathcal{L}(G)$. If we denote by $I(l)$ the rectangular region in the modified obstacle - free environment labelled by l, a feasible string solving Problem 1 satisfies $x_0 \in I(l_{i_1})$ and $x_f \in I(l_{i_m})$. Thus, feasible strings can be determined by using a standard algorithm for finding paths in a graph. This problem is treated in Section IV-B.

c) Generation of provably correct control laws: To generate continuous robot control laws implementing a discrete trajectory given in terms of a string $s = (l_{i_1}, l_{i_2}, \ldots, l_{i_m}) \in \mathcal{L}(G)$, we construct a hybrid system by assigning a vector field (feedback control law) to each of the regions labelled by l_{i_j}, $j = 1, \ldots, m$, so that the robot (1), reduced to its observable x, moves in finite time through the set of regions specified by the string and stays in the last region for all times, independent of the exact position of the robot in each of the regions. Such a hybrid system can be formally defined as

$$H = \{\mathcal{X}, \mathcal{Q}, I, f, T\} \tag{4}$$

where $\mathcal{X} = \bigcup_{j=1}^{m} I(l_{i_j})$ is its continuous state space, $\mathcal{Q} = \{l_{i_1}, l_{i_2}, \ldots, l_{i_m}\}$ is its finite set of locations (also called modes), $I : \mathcal{Q} \to 2^{\mathcal{X}}$ is as defined before, $f : \mathcal{Q} \to (\mathcal{X} \to T\mathcal{X})$ is a mapping that specifies the continuous flow (vector field) in each location l with the property that $f_l \in U$, for all $l \in \mathcal{Q}$ (U is the control constraint set as in (1)), and $T \subseteq t$ is its set of transitions given by $T = \{(l_{i_1}, l_{i_2}), (l_{i_2}, l_{i_3}), \ldots, (l_{i_{m-1}}, l_{i_m})\}$. According to this definition, while in location $l \in \mathcal{Q}$, the hybrid system evolves according to

$$\dot{x} = f_l(x),\ x \in I(l),\ l \in \mathcal{Q} \tag{5}$$

The vector fields will be designed so that all initial states in $I(l_{i_j})$ will flow in finite time to $I(l_{i_{j+1}})$, for $j = 1, \ldots, m-1$ and $I(l_{i_m})$ is an invariant for the system, *i.e.,* the system stays in $I(l_{i_m})$ for all times if it is initialized in $I(l_{i_m})$. We use some interesting properties of multi-affine vector fields on rectangles to construct bounded controllers that (i) drive all initial states in a rectangle through a desired facet in finite time, (ii) make a rectangle an invariant, and (iii) make all initial states in a rectangle converge to an arbitrary point inside the rectangle. Moreover, while constructing such controllers, we make sure that the produced vector field is continuous everywhere in $\mathcal{X}$, which ensures smoothness of the produced trajectories. These results are presented in Section V.

d) Conservativeness of the approach and extensions: At a first look, the problem and solution we propose in this paper seem very conservative. This conservativeness has three sources. First, in Problem 1, we assume that the polytopal robot $\mathcal{R}$ can only translate (no rotation is allowed) and it is fully actuated. However, with a bit of extra work, a rotating robot can be easily accommodated by constructing a large enough polytope $\mathcal{R}$ that includes all 2D (or 3D rotations) of the actual robot around the reference point x. Also, under-actuated robots such as unicycles can be treated under this framework by constructing a diffeomorphic map between its inputs and the velocity of the reference point x, and by properly mapping the corresponding constraint sets [13].

Second, in this paper, we restrict our attention to rectangular partitions, even though other types of partitions, such as triangulations, might capture the geometry of the environment and obstacles more efficiently. Our motivation for this choice is threefold. First, as opposed to triangulations, rectangulations are easy to define in 3D, or any other higher dimension. Second, rectangulations can be represented very efficiently using 2^N - trees (quad-trees, octrees, etc.). Third, the geometry of the rectangles is fundamental in developing a computationally efficient and fully automated procedure for the construction of the hybrid system (4).

Third, our solution to Problem 1 might seem conservative in the sense that a whole set of robot initial states around x_0 are driven in finite time to a set around x_f, with eventual convergence to x_f. However, this approach provides robustness with respect to knowledge of initial and current state of the robot and exact geometry of environment.

IV. Discrete representation and solution

For clarity, in this section we restrict our attention to the 2D case ($N = 2$). The extension to the 3D case ($N = 3$) is straightforward, as stated at the end of this section.

A. Constructing a finite discrete representation

We start by formulating a new problem in which the non-rotating polytopal robot $\mathcal{R}$ is shrunk to its observable x by correspondingly shrinking the environment boundaries and enlarging the obstacles. If we assume that the obstacles O^i, $i = 1, \ldots, n$ and the robot $\mathcal{R}$ are all described as the convex hull of their vertices, there exists a simple algorithm to solve this problem, which consists of polyhedral operations only [1]. In Fig. 2 (a), we illustrate the application of the algorithm to the case study from Fig. 1. It is important to note that one can easily accommodate non-convex polygonal obstacles by modelling them as overlapping convex obstacles.

To construct a discrete representation of the obstacle - free space in the modified environment, we use quad-trees [15], [1], to which we add adjacency relations to obtain the graph G from equation (2). The edges of this graph correspond to possible transitions between neighbor rectangles of the partitioned free space. A quad-tree decomposition and the graph G for the modified environment of Fig. 2 (a) is given in Fig. 2 (b).

B. Generating a solution to the discrete problem

Problem 1 is feasible if there exists a path in the graph G from the start node containing x_0 to the end node containing x_f. Such a path does not exist if the start and the end points are separated by overlapping obstacles or by obstacles "unsafely" close to each other.

If the problem is feasible, we choose among possibly several feasible paths by using an optimality criterion, which is briefly described here. Our goal is to produce trajectories which are "safe" with respect to approaching obstacles and "minimal" with respect to the distance travelled by robot. If only safety was of interest, since in general small rectangles are a measure of "closeness" to obstacles, a good choice of cost for $(l_i, l_j) \in t$ would be the inverse of area of $I(l_j)$. In this case, at each step, a transition towards a larger neighbor would be cheaper. However, such costs would cause long trajectories through large rectangles. To deal with this, we attach to each transition $(l_i, l_j) \in t$ of G a cost equal to the ratio of perimeter over area of rectangle $I(l_j)$. Such a cost, while making transitions to larger rectangles cheaper, gives a trajectory through a rectangle the same cost as a trajectory through two larger rectangles produced at the previous quad-tree iteration. We use Dijkstra's algorithm for finding the path with minimum cost in the weighted graph. The obtained path for the graph in Fig. 2 (b) is given in Fig. 3 (a).

In order to implement such a path, we need to construct a hybrid system (4) whose location succession will follow exactly the transitions from the obtained path. To this goal we develop

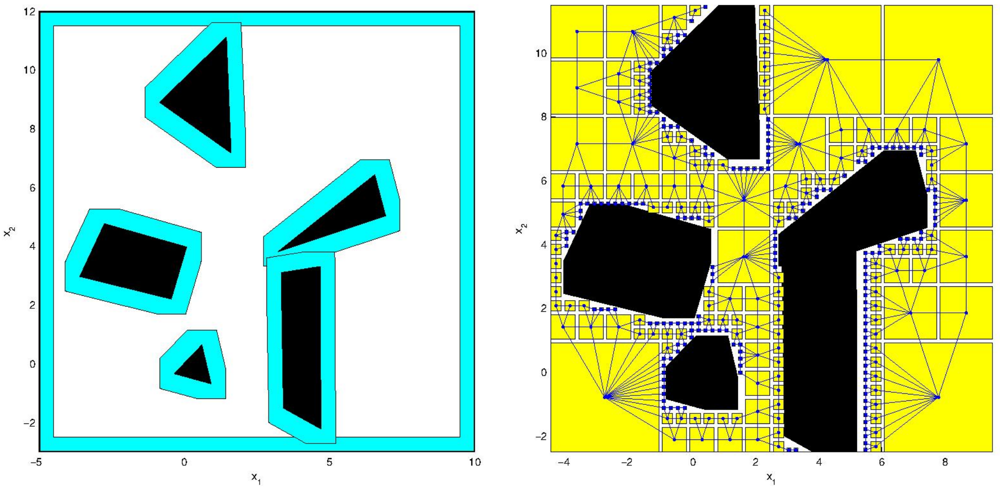


Fig. 2. Construction of the discrete representation of the obstacle - free environment for the case study in Fig. 1: (a) The obstacles are enlarged and the environment is shrunk (shaded portions will be considered occupied by the robot shrunk to its observable), (b) The graph G (equation (2)) is the quotient of the quad-tree partition of the free space augmented with adjacency relations.

in Section V a controller driving all initial states in a rectangle through an exit facet in finite time. Since we cannot determine exactly the point on the facet through which a rectangle is left, in order to avoid hitting a wrong rectangle, whenever we need to make a transition from a rectangle to a smaller rectangle, the larger rectangle is split (in up to three rectangles) according to the dimension of the smaller rectangle. Moreover, as detailed in Section V, in order to construct a hybrid system whose vector field is continuous everywhere (therefore producing smooth trajectories), adjacent rectangles are required to exactly share facets. To this goal, a path might need to be refined by further splitting the corresponding rectangles before constructing the hybrid system. Fig. 3 (b) illustrates this idea for the path in Fig. 3 (a).

The extension of these algorithms to the 3D case is straightforward. The only differences are that a quad-tree becomes an oct-tree and that the cost associated with a transition to a rectangle is facet area/volume of that rectangle.

V. Generation of provably correct robot control laws

A. Preliminaries

A full dimensional (closed) rectangle R_N in $\mathbb{R}^N$ is defined by two vectors $m = (m_1, m_2, \ldots, m_N) \in \mathbb{R}^N$ and $M = (M_1, M_2, \ldots, M_N) \in \mathbb{R}^N$, $m_j < M_j$, $j = 1, \ldots, N$ as $R_N = \prod_{i=1}^{N}[m_i, M_i]$. We denote the vertices of R_N by v_i, $i = 1, \ldots, 2^N$ and the set of all vertices by $V_N = \prod_{i=1}^{N}\{m_i, M_i\}$. Let F_i and n_i, $i = 1, \ldots, 2N$ denote the facets and the corresponding unit outer normals, respectively. For simplicity of exposition, we denote the opposite facet of F_j by $\bar{F}_j$, $j = 1, \ldots, N$. The facets are numbered so that $\bar{F}_j = F_{j+N}$, $j = 1, \ldots, N$.

A multi-affine vector field $f : \mathbb{R}^N \longrightarrow \mathbb{R}^N$ is a polynomial in the indeterminates $x_1, \ldots, x_N$ with the property that the degree of f in any of the indeterminates $x_1, \ldots, x_N$ is less than or equal to 1. Formally, f has the form

$$f(x) = \sum_{i_1,\ldots,i_N \in \{0,1\}} f_{i_1,\ldots,i_N} x_1^{i_1} \cdots x_N^{i_N}, \qquad (6)$$

with $f_{i_1,\ldots,i_N} \in \mathbb{R}^N$ for all $i_1, \ldots, i_N \in \{0,1\}$ and using the convention that if $i_k = 0$, then $x_k^{i_k} = 1$. For example, for $N = 2$, all multi-affine functions have the form $f(x_1, x_2) = f_{00} + f_{10}x_1 + f_{01}x_2 + f_{11}x_1x_2$, where $f_{ij} \in \mathbb{R}^2$, $i, j \in \{0,1\}$. In [16] it was proved that a multi-affine vector field is uniquely determined by its values at the vertices of a full dimensional rectangle and its restriction to the rectangle is a convex combination of these values. Specifically,

$$f(x) = \sum_{i=1}^{2^N} c_i g_i,\ g_i = f(v_i), i = 1, \ldots, 2^N \qquad (7)$$

with c_i defined as:

$$c_i = \prod_{j=1}^{N} \left(\frac{M_j - x_j}{M_j - m_j}\right)^{1-b_{i_j}} \cdot \left(\frac{x_j - m_j}{M_j - m_j}\right)^{b_{i_j}} \qquad (8)$$

In (8), $(b_{i_1}\, b_{i_2}, \ldots, b_{i_N})$ is the representation of $(i-1)$ in base 2, $i = 1, \ldots, 2^N$. It is easy to see that $0 \le c_i \le 1$, $i = 1, \ldots, 2^N$, and $\sum_{i=1}^{2^N} c_i = 1$.

B. Design of controllers

All the results in this section are given for the 3D case $N = 3$, *i.e.*, for parallelepipeds R_3. They can be easily adjusted for the planar case $N = 2$ as stated at the end of this section. Due to space constraints, we omit all the proofs, and refer the interested reader to [13]. In what follows, we consider the following system

$$\dot{x} = f(x),\ f : R_3 \to U, \qquad (9)$$

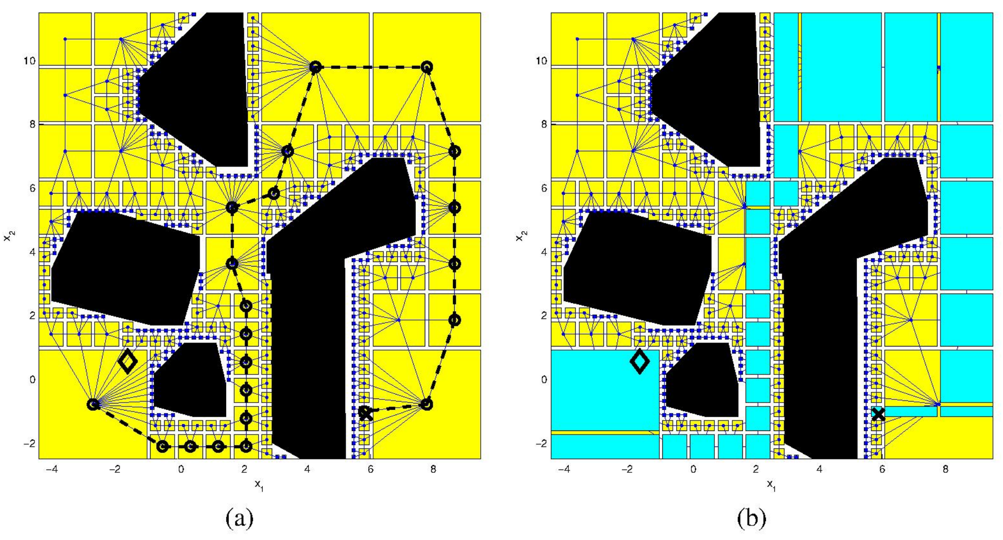


Fig. 3. Generating a solution to the discrete problem corresponding to the case study in Fig. 1: (a) optimal path from initial to final node (rectangle) in the graph G. $\diamond$ and $\times$ denote the initial and final position of robot observable, respectively. (b) new path after rectangle splitting.

where f is a multi-affine vector field as defined in equation (6) and $U \subseteq \mathbb{R}^3$ is a polyhedral set of constraints.

Theorem 1 gives a characterization of all multi-affine vector fields (9) with polyhedral bounds whose trajectories leave the rectangle R_3 through a desired facet. Without restricting the generality of the theorem, the exit facet is assumed to be F_1.

Theorem 1 (Exit through a facet): System (9) drives all initial states in R_3 through facet F_1 in finite time, with the additional property that the rectangle is left first time facet F_1 is hit, if and only if the values g_i of the vector field f at the vertices v_i, $i = 1, \ldots, 8$ are in $U \setminus \{0\}$ and satisfy the following conditions:

$$\begin{gathered} n_1^T g_i \geq 0,\ n_j^T g_i \leq 0, \\ i = 1, \ldots, 8; \text{ and } j \in \{2, \ldots, 6\} \text{ with } v_i \in F_j \end{gathered} \tag{10}$$

$$\textstyle\sum_{i,\, v_i \in F_1} n_1^T g_i > 0,\ \sum_{i,\, v_i \in \bar{F}_1} n_1^T g_i > 0 \tag{11}$$

$$\begin{gathered} \textstyle\sum_{i,\, v_i \in F_1 \cap F_j} (n_1^T - n_j^T) g_i > 0, \\ \textstyle\sum_{i,\, v_i \in \bar{F}_1 \cap F_j} (n_1^T - n_j^T) g_i > 0,\ j \in \{2, 3, 5, 6\} \end{gathered} \tag{12}$$

The satisfaction of inequalities (10) guarantees that R_3 cannot be left through any facet except F_1 and that the motion speed towards F_1 is positive in every point of R_3. Inequalities (11) mean that in at least one vertex of F_1 and at least one vertex of $\bar{F}_1$ the vector field projections on direction n_1 are strictly positive, while inequalities (12) guarantee that there does not exist any equilibrium point on edges of F_1 or $\bar{F}_1$.

Theorem 2 (Stay inside): A rectangle R_3 is an invariant for system (9) if and only if the values g_i of the vector field f at the vertices v_i, $i = 1, \ldots, 8$ are in U and satisfy the following conditions:

$$n_j^T g_i \leq 0,\ j \in \{1, \ldots, 6\} \text{ and } v_i \in F_j \tag{13}$$

Geometrically, inequalities (13) mean that on all facets, the vectors g_i point inside the rectangle R_3.

Theorem 3 (Asymptotic stabilization): If all inequalities (13) are strict, then system (9) has a unique equilibrium in R_3, which is asymptotically stable with region of attraction R_3. For arbitrary $x_e \in R_3$, a vector field (9) whose trajectories asymptotically converge to x_e from everywhere in R_3 can be chosen as:

$$g_i = (x_e - v_i) \cdot a,\ i = 1, \ldots, 8, \tag{14}$$

where $a > 0$ is any constant for which $g_i \in U$.

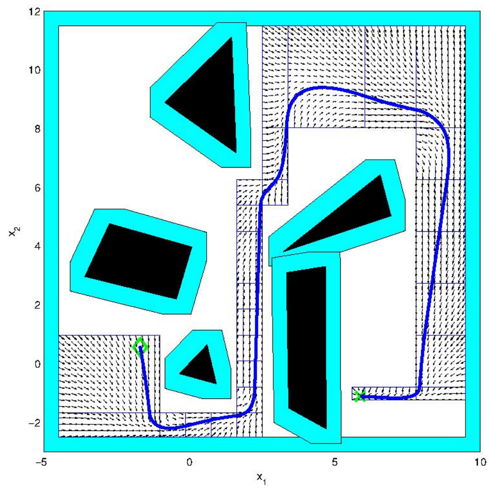
Remark 1: If the constraint set U contains an open neighborhood of the origin in $\mathbb{R}^3$, then there always exist values g_i, $i = 1, \ldots, 8$ satisfying the inequalities from Theorems 1 or 2. In this paper, we assume that this condition holds.

Remark 2: Since the set U was assumed polyhedral, checking for the existence of allowed g_i in the above theorems reduces to solving linear inequalities, for which there exists several software packages. For a choice of g_i satisfying such inequalities, the vector field everywhere in the rectangle is easily constructed using equations (7,8).

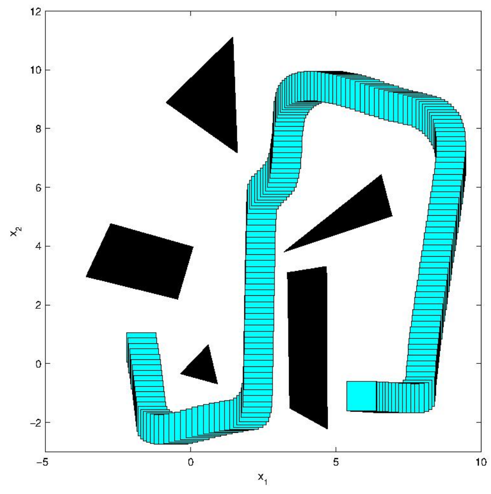
Remark 3 (2D case): All the above theorems hold also in the two-dimensional case $N = 2$. Since the facets of a rectangle R_2 are also its edges, in Theorem 1, inequalities (12) become redundant. A simple way to interpret the case $N = 2$ as a particular case of $N = 3$ is that a rectangle R_2 can be viewed as facet F_3 or $\bar{F}_3$ of R_3.

C. Construction of a hybrid system

For a given string $s = (l_{i_1}, l_{i_2}, \ldots, l_{i_m}) \in \mathcal{L}(G)$, with $x_0 \in I(l_{i_1})$, $x_f \in I(l_{i_m})$, and with the property that $I(l_{i_j})$ and $I(l_{i_{j+1}})$, $j = 1, \ldots, m-1$ share a facet, we use Theorem 1 in $I(l_{i_j})$, $j = 1, \ldots, m-1$ and Theorem 3 in $I(l_{i_m})$ to construct a hybrid system giving provably correct robot feedback control laws $f_{l_{i_j}}(x)$. As stated above, for each vertex, there will exist a whole polyhedral set of allowed values for the corresponding vector field. We use these degrees of freedom to achieve two desiderata: (i) smoothness of the corresponding trajectories, and (ii) maximize the speed of motion. For smoothness, we "stitch" together vector fields in adjacent rectangles so that



(a)



(b)

Fig. 4. Construction of a hybrid system for the sequence of rectangles from Fig. 3 (b): (a) vector field assignment and the trajectory of the observable in the modified environment; (b) robot motion in the initial environment

they completely agree on the common facet. Using the facts that the restriction of a multi-affine vector field to a facet remain multi-affine and a facet of a rectangle is a rectangle itself, then the agreement on a whole facet is equivalent to agreement at vertices, since the vector field is uniquely determined at vertices. For each vertex of each rectangle, we then determine a unique value of the field as the solution of a linear program, whose polyhedral constraint set is as defined above. The objective is the projection of the velocity along a weighted sum of outer normals of rectangles to be reached, where the weights are equal to the lengths of the rectangles in the direction of the outer normals of the exit facets. It can be shown that, if the set U contains an open neighborhood of the origin, there exist a smooth trajectory for any string. The details are omitted due to space constraints. An illustration of this idea is given in Fig. 4. The vector fields and the trajectory of the observable in the modified environment from x_0 to x_f are shown in Fig. 4 (a). Note that the vector field is everywhere continuous and the trajectory is smooth. The motion of the actual square robot is shown in Fig. 4 (b), which concludes the solution of the case study considered in Fig. 1. The time required to build the discrete representation was 1.3 seconds, the free path was found in 0.4 seconds, and the control laws were generated in 1.4 seconds, leading to a total conputation time of 3.1 seconds. A 3D case study is presented in [13].

VI. Concluding remarks

We developed a computational framework for automatic deployment of robots with control bounds in rectangular 2D and 3D environments with polytopal obstacles, available for download from [14]. Central to our approach is to use partitions in the form of quad-trees and oct-trees to capture the complexity of the environment and discrete abstractions of hybrid systems to construct provably correct robot control laws. Given an environment and robot geometry, and start and end positions of the robot, feedback control laws producing an interpolating motion are automatically generated.

References

[1] J. Latombe, *Robot Motion Planning*. Kluger Academic Pub., 1991.

[2] D. E. Koditschek, "The control of natural motion in mechanical systems," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 113, no. 4, pp. 548–551, 1991.

[3] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high dimensional configuration spaces," *IEEE Trans. on Robotics and Automation*, vol. 12, pp. 566–580, 1996.

[4] G. Lafferriere and H. Sussmann, "A differential geometric approach to motion planning," in *Nonholonomic Motion Planning*, Z. Li and J. Canny, Eds. Kluwer Academic Publishers, 1993, p. 235270.

[5] P. Rouchon, M. Fliess, J. Levine, and P. Martin, "Flatness, motion planning and trailer systems," in *Proc. of the 32rd IEEE Conference on Decision and Control*, Austin, TX, December 1993, p. 27002705.

[6] R. Murray and S. Sastry, "Nonholonomic motion planning : Steering using sinusoids," *IEEE Trans. on Automatic Control*, pp. 700–716, 1993.

[7] A. Astolfi, "Discontinuous control of nonholonomic systems," *IEEE Transactions on Automatic Control*, vol. 27, pp. 37–45, 1996.

[8] F. Lamiraux and J.P.Laumond, "Smooth motion planning for car-like vehicles," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 498–502, 2001.

[9] P. Cheng, Z. Shen, and S. M. LaValle, "Rrt-based trajectory design for autonomous automobiles and spacecraft," *Archives of Control Sciences*, vol. 11(XLVII), no. 3-4, pp. 167–194, 2001.

[10] D. C. Conner, A. A. Rizzi, and H. Choset, "Composition of local potential functions for global robot control and navigation," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Las Vegas, Nevada, 2003.

[11] C. Belta, V. Isler, and G. J. Pappas, "Discrete abstractions for robot planning and control in polygonal environments," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 864–874, 2005.

[12] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for mobile robots," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, April 2005.

[13] M. Kloetzer and C. Belta, "A framework for automatic deployment of robots in 2d and 3d environments," Boston University, Tech. Rep., 2006, http://iasi.bu.edu/ cbelta/tech-rep/KB-iros06-tr.pdf.

[14] ——, "Automatic deployment of robots in 2d and 3d environments," URL http://iasi.bu.edu/∼software/deploy-2D-3D.htm.

[15] R. A. Finkel and J. L. Bentley, "Quad-trees: a data structure for retrieval on composite keys," *ACTA Informatica*, vol. 4, pp. 1–9, 1974.

[16] C. Belta and L. Habets, "Control of a class of nonlinear systems on rectangles," Boston University, Tech. Rep. CISE 2005-IR-0060, 2005, http://www.bu.edu/systems/research/publications/2005/2005-IR-0060.pdf.