Discrete Abstractions for Robot Motion Planning and Control in Polygonal Environments

Calin Belta, Volkan Isler, and George J. Pappas

***Abstract*— In this paper, we present a computational framework for automatic generation of provably correct control laws for planar robots in polygonal environments. Using polygon triangulation and discrete abstractions, we map continuous motion planning and control problems specified in terms of triangles to computationally inexpensive finite state transition systems. In this framework, powerful discrete planning algorithms in complex environments can be seamlessly linked to automatic generation of feedback control laws for robots with under-actuation constraints and control bounds. We focus on fully-actuated kinematic robots with velocity bounds and (under-actuated) unicycles with forward and turning speed bounds.**

***Index Terms*— Motion planning, control, triangulation, discrete abstraction, hybrid system, bisimulation.**

I. Introduction

MOTION planning for robots in geometrically complex environments is a fundamental problem that received a lot of attention lately [24], [25], [6]. The vast literature on this topic can be divided in two schools of thought. The first focuses on the complexity of the environment, while assuming that the robot is fully actuated with no control bounds, or ”free flying” [24]. This is the main simplifying assumption in most of the path planning methods based on navigation techniques. Continuous paths from initial to final configurations in the robot task space can be found using roadmap methods such as Voronoi diagrams, visibility graphs, and freeway methods [24], potential fields [19], [21], [24], [37], or navigation functions [20], [36]. Discrete paths can also be built using cellular decompositions of the configuration space [24] or probabilistic roadmaps [18], [26]. Even though these methods produce paths that are perfectly valid from a planning perspective, the robot might fail to accomplish the task because of under-actuation and control constraints.

The other school of thought focuses on the detailed dynamics or kinematics of the robot, while assuming trivial environments. Most of these methods are continuous and based on nonlinear control theory. To properly deal with nonholonomic systems, some of these approaches are differential geometric [23], [43] or exploit concepts such as flatness [39]. Other approaches use different types of input parametrization leading to multi-rate [42] or time varying [29], [9], [34], [28] control laws. Finally, discontinuous control laws obtained by combining different controllers [22] or by applying nonsmooth transformations of the state space [10], [3] have been proposed. While properly dealing with issues such as under-actuation and nonholonomy, such approaches face serious algorithmic challenges in complex environments [41].

In real world applications, the robots have control and under-actuation constraints and the environments can be very complex. It seems very difficult, if at all possible, to mathematically formulate and solve (analytically or computationally) such a motion planning problem using either of the approaches presented above. *Integrating* the methods of the two schools of thought is a very promising and challenging research avenue. In this paper, we advocate a *hierarchical* or *compositional* approach for robot motion planning which integrates the strengths of algorithmic motion planning in complex environments with continuous motion generation for robots with control constraints. Our integration of discrete and continuous approaches necessarily results in a *hybrid systems* framework [1].

We focus on polygonal (planar) environments and start by constructing a triangulation of the environment. The triangulation provides a partition of the environment in a manner that complex environments can be thought of as compositions of simple triangles. This geometric decomposition reduces the complexity of motion generation as it allows focusing on the complexity of the robot dynamics defined in triangles. A novel technical challenge then arises, as we need to generate motion (or design controllers) for robots with actuation and/or control constraints that are able to steer the robot from one triangle to an adjacent triangle, or keep the robot in a given triangle. If the robot controllers (one per triangle) can achieve this *independently* of the initial condition inside the triangle, then the partition due to triangulation satisfies the so-called *bisimulation* property [2]. This special reachability-preserving partition of the state space allows us to have a formal notion of system equivalence, namely bisimulation, between the discrete abstraction of the robot used for algorithmic motion planning, and the continuous robot dynamics that is operating under the influence of a hybrid controller which is switching on the boundaries of the triangulation. Being equivalent allows the high-level, discrete model to focus on the complexity of the environment and ignore the low-level dynamics of the robot, while having a formal guarantee that the sequence of triangles generated by (any) discrete algorithm are dynamically feasible by the robot dynamics. *The novelty of our hierarchical approach is on providing mathematically precise relationships*

This work was supported by ARO MURI and NSF ITR grants.

C. Belta is with the Department of Mechanical Engineering and Mechanics, Drexel University, Philadelphia, PA 19104. E-mail : calin@drexel.edu

V. Isler is with the Department of Computer and Information Sciences, University of Pennsylvania, Philadelphia, PA 19104. E-mail : isleri@cis.upenn.edu

G. J. Pappas is with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104. E-mail : pappasg@ee.upenn.edu

C. Belta is the corresponding author

between the high-level discrete model and the low-level continuous model using modern approaches from hybrid control theory.

Among all literature on robot planning and control, this work is closest related to [7], [8]. In these papers, the authors consider a polygonal partition of a planar configuration space and assign vector fields in each polygon 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 (temperature-like) scalar functions determined as solutions of Laplace's equation with boundary conditions imposed so that the integral curves can only leave through a desired facet. The resulting vector field in a polygon has fixed direction and is determined up to a multiplying scalar, which can then be used to accommodate speed constraints for fully actuated kinematic robots. For dynamic robots modelled as double integrators with speed and acceleration bounds, the authors use a composition of three hybrid controllers based on previous results published in [38], [35].

Even though the motivating ideas are the same, in this paper we use fundamentally different tools, which are much more suited for computation and composition than the one presented in [7], [8]. By exploiting some interesting properties of affine functions in simplexes, we can characterize all affine vector fields whose integral curves leave the simplex through a desired facet in finite time. They are parameterized by polyhedral sets capturing the allowed velocities at the vertices. We use these degrees of freedom to accommodate general polyhedral velocity bounds for kinematic robots and to "stich" the vector fields in adjacent triangles to produce smooth trajectories. Moreover, we don't have to construct any diffeomorphism, solve any boundary value problem, and smooth out any boundary conditions. *Given the vertices of a polygon, the triangulation and generation of provably correct feedback controllers implementing a high level discrete strategy is fully automated.*

The paper is structured as follows. In Section II, we formulate the problem, give the necessary definitions, and present our approach. The main results and the algorithms for automatic generation of vector fields mapping to discrete specifications are given in Sections III and IV. These results are then used in Section V to generate provably correct feedback control laws for fully actuated kinematic robots and unicycles. Simulation results are shown in Section VI. The paper ends with concluding remarks and brief exposition of future research directions in Section VII.

II. Problem formulation and approach

We consider planar robots described in coordinates by control systems of the form:

$$\dot{q} = F(q,u),\ q \in Q,\ u \in U \tag{1}$$

where q is the state of the robot and u is its control input. Q and U are subsets of Euclidean spaces of appropriate dimensions. For example, for a fully actuated kinematic point-like robot with position vector x in some world frame, $q = x \in \mathbb{R}^2$ and $F(q,u) = u$. For a planar unicycle described by centroid coordinates position vector x and orientation θ in a world frame, and controlled by driving and steering velocities v and ω, we have $q = \{[x^T, \theta]^T \,|\, x \in \mathbb{R}^2,\ \theta \in [0, 2\pi)\}$, $u = (v, \omega) \in \mathbb{R}^2$, $F(q,u) = [\cos\theta v\ \sin\theta v\ \omega]^T$.

As is usually the case in practice when dealing with complex environments, we assume that the motion planning task is "qualitatively" specified. This notion has a dual meaning. First, the task is specified in terms of a robot "observable", while the entire internal state q of the robot is not of interest. This observable can be the centroid of the robot, an interesting point on the robot where a sensor such as a camera is attached, the center of a disk capturing the size of the robot, etc. Formally, this can be modelled by defining a map

$$x = h(q),\ x \in \mathcal{P} \tag{2}$$

where we assumed that the observable x takes values in a polygon $\mathcal{P}$, which does not change in time, *i.e.*, the environment is static. This polygon can be complex, with a large number of vertices and it can contain polygonal holes modelling obstacles or undesired regions in the environment. $\mathcal{P}$ can be the original planar environment if the size of the robot is negligible or its image through some map which accounts for the size and shape of the robot [41].

Second, it is not necessary to have information on the exact value of observable x, but rather to be able to decide its inclusion in certain regions of interests. For example, we need to make sure that the robot does not collide with an obstacle of given geometry. Or, to win the visibility-based game as the one formulated in [16], [15], the pursuer only needs to make sure that it is in the same triangle as the evader. Throughout this paper, we assume that these regions are triangles or unions of adjacent triangles. There are several supporting arguments for our choice. First, the problem of triangulating a polygon is well studied and computationally efficient algorithms are available [31]. Second, as we will see later in the paper, triangles have special properties that can be exploited to map such qualitatively described tasks to discrete transition systems over a finite set of symbols, with automatic generation of provably correct robot control laws.

We label each triangle using a finite set of symbols $L = \{l_1, l_2, \ldots, l_M\}$ and use the notation $I(l_i) \subset \mathcal{P}$ to denote the region of $\mathcal{P}$ contained by triangle l_i, including its boundary. Clearly

$$\mathcal{P} = \bigcup_{l_i \in L} I(l_i) \tag{3}$$

This idea is illustrated in Figure 1, where the shaded polygons are forbidden regions in a task specification (*e.g.,* obstacles), and the triangulation is achieved by a maximal set of non-intersecting diagonals [31].

Definition 1 (Dual graph): The *dual graph* of a triangulation is a simple graph

$$DG = (L, t) \tag{4}$$

whose nodes $L = \{l_1, l_2, \ldots, l_M\}$ correspond to the symbols used for labelling the triangles, and the edge set $t \subset L \times L$ denotes an adjacency relation between the corresponding triangles.

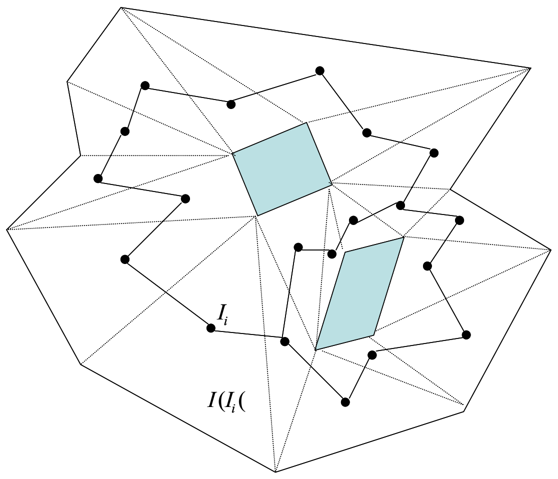

Fig. 1. Triangulation of a planar polygon and the dual graph.

Therefore, $(l_i, l_j) \in t$ for $i \neq j$, if the triangles $I(l_i)$ and $I(l_j)$ are adjacent, *i.e.*, if they share a line segment. Naturally, the edge set is symmetric, that is if $(l_i, l_j) \in t$ then $(l_j, l_i) \in t$.

The dual graph DG defined by (4) serves as our discrete modelling abstraction for algorithmic motion planning and provably correct control of robots with specifications given in terms of sets. Its nodes can be seen as "qualitative" robot states, while its edges model state transitions. More formally, the task specifications are given in the language of the dual graph:

Definition 2 (Language of dual graph): The language $\mathcal{L}(DG)$ of the dual graph DG is the set of all strings $(l_{i_1}, l_{i_2}, \ldots, l_{i_m})$, $l_{i_j} \in L$, $i_j \in \{1, \ldots, M\}$, $j = 1, \ldots, m$, with $(l_{i_j}, l_{i_{j+1}}) \in t$, $j = 1, \ldots, m-1$.

The high level specifications given in terms of strings in the language of the dual graph are determined at a higher hierarchical level, which is beyond the scope of this paper. For example, such strings can be determined as solutions of path searching problems on graphs, for which there exist many powerful algorithms, such as depth-first search, breadth-first search, etc. Or, these strings can be solutions to coverage or motion generation with respect to temporal logic specifications [40]. Other examples include solutions to discrete games. For example, in the visibility based game presented in [16], [15], which is the main motivation for the framework proposed in this paper, the wining strategy of the pursuer is to randomly generate strings in the language $\mathcal{L}(DG)$. The focus of this paper is not on determining such strings in the language $\mathcal{L}(DG)$, but rather on creating a computationally efficient and provably correct framework in which a given string is automatically translated to robot control laws. More formally, we provide a solution to the following problem:

Problem 1: Construct a set $\mathcal{U}$ of state feedback controllers so that, for any string $(l_{i_1}, l_{i_2}, \ldots, l_{i_m}) \in \mathcal{L}(DG)$, there exists $u \in \mathcal{U}$ driving the robot (1), (2) from any initial state $q_0 \in Q$ with $h(q_0) \in I(l_{i_1})$ so that its observable x moves through the regions $I(l_{i_1})$, $I(l_{i_2}), \ldots, I(l_{i_m})$ in finite time, and stays in $I(l_{i_m})$ for all future times.

In other words, if a solution to Problem 1 exists, then the robot can automatically achieve any discrete specification in the language $\mathcal{L}(DG)$. The set $\mathcal{U}$ will contain two types of controllers: (I) feedback controllers driving the robot from any initial state $q_0 \in Q$ with $h(q_0) \in I(l)$ so that its observable x moves in finite time to $I(l')$, for any $l, l' \in L$ with $(l, l') \in t$, and (II) feedback controllers driving the robot so that the observable x stays in $I(l)$ for all times, for all initial states $q_0 \in I(l)$, and for all $l \in L$. Indeed, it is easy to see any string $(l_{i_1}, l_{i_2}, \ldots, l_{i_m})$ can be implemented by using controllers of type (I) for (l_{i_1}, l_{i_2}), (l_{i_2}, l_{i_3}), $\ldots$, $(l_{i_{m-1}}, l_{i_m})$ and a controller of type (II) for l_{i_m}. On the other hand, we need controllers of type (I) and (II) to implement all strings of length two and one, respectively.

We provide a solution to Problem 1 by first constructing vector fields in the observable polygonal space and then by generating corresponding robot control laws. We construct a set of (maximum) four vector fields for each triangle: one that makes the triangle an invariant for the observable, which will lead to a controller of type (II), and (maximum) three that drive all initial values of the observable in the triangle to each of its neighbors, which will lead to controllers of type (I). The natural framework for representing such a construction is that of *hybrid systems*, and is presented below. A more general definition on a hybrid system can be found in [1].

Definition 3 (Hybrid system): A hybrid system storing vector fields implementing the language $\mathcal{L}(DG)$ is a tuple

$$HS = (\mathcal{P}, \mathcal{Q}, Inv, f, T, O), \tag{5}$$

where

- $\mathcal{P}$ is its (polygonal) continuous state space (3). $x \in \mathcal{P}$ is called continuous state.

- $\mathcal{Q}$ is its finite set of locations defined by

$$\mathcal{Q} = \{q_{ij} \,|\, i, j = 1, \ldots, M, \text{ and } i = j \text{ or } (l_i, l_j) \in t\}. \tag{6}$$

$q_{ij} \in \mathcal{Q}$ are called discrete states, or locations. The overall state of the system is therefore $(q_{ij}, x) \in \mathcal{Q} \times \mathcal{P}$.

- $Inv : \mathcal{Q} \to 2^{\mathcal{P}}$ is a map which assigns to each discrete state $q_{ij} \in \mathcal{Q}$ an invariant set defined by

$$Inv(q_{ij}) = I(l_i). \tag{7}$$

- $f : \mathcal{Q} \to (\mathcal{P} \to T\mathcal{P})$ is a mapping that specifies the continuous flow (vector field) in each location q_{ij}. $f_{q_{ii}}$ keeps the system in the triangle $I(l_i)$ for all times. $f_{q_{ij}}$, with i, j so that $(l_i, l_j) \in t$, drives all initial continuous states $x \in I(l_i)$ to $I(l_j)$ in finite time through the common boundary $I(l_i) \bigcap I(l_j)$.

- $O : \mathcal{Q} \times \mathcal{P} \to L$ is an output map defined as

$$O(q_{ij}, x) = l_i, \; q_{ij} \in \mathcal{Q}, \; x \in \mathcal{P} \tag{8}$$

Note that the number of discrete states (locations) $|\mathcal{Q}|$ of the hybrid system defined above is at most $4 \times |L|$, since every vertex of DG has at most three transitions.

According to the above definition, while in location $q_{ij} \in \mathcal{Q}$, the system evolves according to

$$\dot{x} = f_{q_{ij}}(x), \; x \in Inv(q_{ij}), \tag{9}$$

and outputs l_i. Similarly to the dual graph, the language of HS is defined as the set of discrete states reached by the system:

Definition 4 (Language of hybrid system): The language $\mathcal{L}(HS)$ of the hybrid system HS is the set of all strings produced by the output map O as HS evolves in time.

If a hybrid system HS can be constructed according to Definition 3, then DG and HS produce the same language, *i.e.,* they are *language equivalent*.

Remark 1: The strings in the language of the dual graph DG defined by (4) can be seen as transition systems. The hybrid system HS defined by (5) and constructed as shown above is *bisimilar* with all such transition systems. The bisimilarity relation, introduced in [33], [27], formally defined for linear control systems in [32], and for nonlinear systems in an abstract categorical context in [13], is the main tool in providing a framework in which infinite dimensional continuous and hybrid systems can be collapsed to finite state automata. In these works, a continuous or hybrid system is iteratively partitioned until it becomes equivalent with its discrete quotient induced by the partition with respect to reachability properties. In this paper, motivated by robotic motion planning, we consider the inverse problem: given a set of discrete states and allowed transitions in the form of a dual graph, we construct a hybrid system bisimilar with all possible transition systems. However, in the future, we will consider refined partitioning as in the bisimulation algorithm presented in [2] to accommodate different robot dynamics and control constraints.

In this paper, we restrict our attention to affine vector fields with polyhedral bounds:

$$f_{q_{ij}}(x) = A_{q_{ij}}x + b_{q_{ij}} \in V,\ x \in Inv(q_{ij}),\ q_{ij} \in \mathcal{Q} \quad (10)$$

where $A_{q_{ij}} \in \mathbb{R}^{2\times 2}$, $b_{q_{ij}} \in \mathbb{R}^2$, and $V \subseteq \mathbb{R}^2$ is a polyhedral set. For this class of systems, which we call *triangular affine hybrid systems*, we show in Section III that there is a simple and computationally efficient method for characterization of existence and explicit construction of HS. If requirements such as smoothness of the produced control laws over several triangles or minimization of time spent traversing a set of triangles are required, then the algorithm is refined to produce a corresponding solution satisfying the additional requirements in Section IV. Finally, depending on the robot kinematics and control constraints, feedback control laws mapping to these vector fields are determined depending on the robot kinematics. These results are shown in Section V.

III. Triangular affine hybrid systems

In this Section, we characterize all affine vector fields driving all initial states in a triangle through a facet in finite time or keeping all initial states in a triangle forever. We also provide formulas for the construction of such vector fields which leads to the construction of the triangular affine hybrid system (5). Even though in this paper we are only concerned with triangles, the results are presented for the case of an arbitrary dimensional Euclidean space, where the generalization of a triangle is a simplex. A related exposition of some of the results in this section can be found in [4], [12].

A. Affine functions in simplexes

This section presents an interesting property of an affine function defined in a simplex: it is uniquely determined by its values at the vertices of the simplex and its restriction to the simplex is a convex combination of these values.

Let $N \in \mathbb{N}$ and consider $N+1$ affinely independent points $v_1, \dots, v_{N+1}$ in the Euclidean space $\mathbb{R}^N$, *i.e.,* there exists no hyperplane of $\mathbb{R}^N$ containing $v_1, \dots, v_{N+1}$. Then the simplex S_N with vertices $v_1, \dots, v_{N+1}$ is defined as the convex hull of $v_1, \dots, v_{N+1}$:

$$S_N = \{x \in \mathbb{R}^N \mid x = \sum_{i=1}^{N+1} \lambda_i v_i,\ \sum_{i=1}^{N+1} \lambda_i = 1,\ \lambda_i \geq 0\} \quad (11)$$

For $i \in \{1, \dots, N+1\}$, the convex hull of $\{v_1, \dots, v_{N+1}\} \setminus \{v_i\}$ is a facet of S_N and is denoted by F_i. Let n_i denote the corresponding unit outer normal vector. The following Lemma states a well known result:

Lemma 1: In any simplex S_N, for an arbitrary $i = 1, \dots, N+1$, the vectors n_j, $j = 1, \dots, N+1$, $j \neq i$ are linearly independent. Moreover, n_i is a strictly negative linear combination of n_j, $j = 1, \dots, N+1$, $j \neq i$.

For $r \in \mathbb{N}$, let $f : \mathbb{R}^N \to \mathbb{R}^r$ be an arbitrary affine function

$$f(x) = Ax + b, \quad (12)$$

with $A \in \mathbb{R}^{r\times N}$ and $b \in \mathbb{R}^r$. Then we have:

Lemma 2: The affine function (12) is uniquely determined by its values $f(v_i) = g_i$, $i = 1, \dots, N+1$ at the vertices of S_N. Moreover, the restriction of f to S_N is a convex combination of its values at the vertices and is given by:

$$f(x) = GW^{-1}\begin{bmatrix} x \\ 1 \end{bmatrix},\ x \in S_N \quad (13)$$

where

$$G = [\ g_1 \dots g_{N+1}\ ] \quad (14)$$

and

$$W = \begin{bmatrix} v_1 & \dots & v_{N+1} \\ 1 & \dots & 1 \end{bmatrix} \quad (15)$$

are $r \times (N+1)$ and $(N+1) \times (N+1)$ real matrices.

Proof: Since $v_1, \dots, v_{N+1}$ are affinely independent, $v_2 - v_1, v_3 - v_1, \dots, v_{N+1} - v_1$ are linearly independent, and therefore, constitute a basis of $\mathbb{R}^N$. An immediate consequence is that, for a given $x \in S_N$, the λ_i's from (11) are uniquely defined and given by:

$$\begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_{N+1} \end{bmatrix} = W^{-1}\begin{bmatrix} x \\ 1 \end{bmatrix},$$

where W is defined by (15) and is easily seen to be non-singular since $v_2 - v_1, v_3 - v_1, \dots, v_{N+1} - v_1$ are linearly independent. Indeed,

$$\begin{aligned} \det W &= \det\begin{bmatrix} v_1 & v_2 - v_1 & \dots & v_{N+1} - v_1 \\ 1 & 0 & \dots & 0 \end{bmatrix} \\ &= (-1)^{N+2}\det\begin{bmatrix} v_2 - v_1 & \dots & v_{N+1} - v_1 \end{bmatrix} \end{aligned}$$

Let $f(v_i) = g_i$, $i = 1, \dots, N+1$. For any $x \in S_N$, there exist unique $\lambda_i \geq 0$, $\sum_{i=1}^{N+1} \lambda_i = 1$ so that $x = \sum_{i=1}^{N+1} \lambda_i v_i$ and we have

$$\begin{aligned} f(x) &= f(\sum_{i=1}^{N+1} \lambda_i v_i) = A \sum_{i=1}^{N+1} \lambda_i v_i + b \\ &= A \sum_{i=1}^{N+1} \lambda_i v_i + b \sum_{i=1}^{N+1} \lambda_i \\ &= \sum_{i=1}^{N+1} \lambda_i (A v_i + b) = \sum_{i=1}^{N+1} \lambda_i g_i \\ &= [\; g_1 \dots g_{N+1} \;] \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_{N+1} \end{bmatrix} \\ &= [\; g_1 \dots g_{N+1} \;] V^{-1} \begin{bmatrix} x \\ 1 \end{bmatrix} \end{aligned} \quad (16)$$

and the Lemma is proved. ■

Remark 2: Note that the restriction of an affine function f to a facet F_i of S_N (i.e. F_i itself is a simplex in $\mathbb{R}^{N-1}$) is affine and for any $x \in F_i$, $f(x)$ is a convex combination of the values of f at the vertices of F_i.

Proposition 1: Let $w \in \mathbb{R}^r$ and $d \in \mathbb{R}$. Then $w^T f(x) > d$ everywhere in S_N if and only if $w^T f(v_i) > d$, $i = 1, \dots, N+1$.

Proof: The necessity follows immediately from the fact that the vertices $v_1, \dots, v_{N+1}$ belong to S_N. For sufficiency, for any $x \in S_N$ we have:

$$w^T f(x) = w^T f(\sum_{i=1}^{N+1} \lambda_i v_i) = w^T \sum_{i=1}^{N+1} \lambda_i f(v_i)$$

$$\sum_{i=1}^{N+1} \lambda_i w^T f(v_i) > d \sum_{i=1}^{N+1} \lambda_i = d$$

■

It is easy to see that the result of Proposition 1 remains valid if $>$ is replaced by $\geq$, $=$, $<$, $\leq$. Also, it is obvious that Proposition 1 remains valid if f is restricted to a facet F_i.

B. Affine feedback control laws in simplexes

In this section we use the properties of affine functions presented above to completely describe the set of all affine vector fields with polyhedral bounds driving all initial states in a simplex through a desired facet in finite time or making the simplex an invariant. We restrict our attention to affine functions (12) with $r = N$ defined on a simplex S_N and with values in a polyhedral subset V of $\mathbb{R}^N$, *i.e.*, to affine vector fields with polyhedral bounds:

$$\dot{x} = f(x),\ f: S_N \to V \subseteq \mathbb{R}^N,\ f(x) = Ax + b \quad (17)$$

where $A \in \mathbb{R}^{N \times N}$ and $b \in \mathbb{R}^N$. As stated before, if the vector field f is known at the vertices ($f(v_i) = g_i$, $i = 1, \dots, N+1$), then in equation (17), A is the matrix obtained by selecting the first N columns of GW^{-1}, while b is the last column of GW^{-1}, *i.e.*,

$$GW^{-1} = [A \,|\, b], \quad (18)$$

where G and W are given by (14) and (15), respectively.

Proposition 2 below gives a characterization of all affine vector fields driving all initial states in a simplex through a facet in finite time. Without restricting the generality of the problem, we assume that the exit facet is F_1.

Proposition 2 (Exit through a facet): There exists an affine vector field (17) driving all initial states in the simplex S_N through the facet F_1 in finite time if and only if the polyhedral sets V_j^e, $j = 1, \dots, N+1$ are nonempty, where

$$V_1^e = V \bigcap \bar{V}_1^e \quad (19)$$

$$V_j^e = V \bigcap \bar{V}_j^e,\ j = 2, \dots, N+1, \quad (20)$$

with

$$\bar{V}_1^e = \{g \in \mathbb{R}^N | n_j^T g \leq 0, j = 2, \dots, N+1, \text{ and } n_1^T g > 0\}, \quad (21)$$

$$\bar{V}_j^e = \{g \in \mathbb{R}^N | n_k^T g \leq 0, k = 2, \dots, N+1,\ k \neq j, \text{ and } n_1^T g > 0\} \quad (22)$$

Proof: For sufficiency, if the sets V_i^e are all nonempty, then choose arbitrary $g_i \in V_i^e$, $i = 1, \dots, N+1$ and construct the unique affine function (13) in S_N satisfying $f(v_i) = g_i$, $i = 1, \dots, N+1$. Since for every $x \in S_N$ $f(x)$ is a convex combination of $g_1, \dots, g_{N+1} \in V$, $f(x)$ is contained in the convex hull of $g_1, \dots, g_{N+1}$. This is the smallest convex set containing $g_1, \dots, g_{N+1}$, and therefore included in V. So, $f(x) \in V$, $\forall x \in S_N$, as required. The restriction of $f(x)$ to an arbitrary facet F_k, $k = 2, \dots, N+1$ is of course an affine function, therefore a convex combination of its values g_j at the corresponding vertices v_j, $j = 1, \dots, N+1$, $j \neq k$. Since $n_k^T g_j \leq 0$, $k = 2, \dots, N+1$, $j \neq k$, using Proposition 1, we conclude that $n_k^T f(x) \leq 0$ everywhere on F_k, so they cannot leave through the facet F_k, $k = 2, \dots, N+1$. On the other hand, since $n_1^T g_j > 0$, $j = 1, \dots, N+1$, we conclude that $n_1^T f(x) > 0$, $\forall x \in S_N$. Therefore, all trajectories of (17) will have a positive speed of motion towards F_1 everywhere in S_N which implies that the simplex will eventually be left.

For necessity, assume there is an affine vector field (17) driving all states in S_N through F_1 in finite time. Let $f(v_i) = g_i$, $i = 1, \dots, N+1$. We will show that g_i satisfies the inequalities of V_i, $i = 1, \dots, N+1$, so all these sets are nonempty. If we assume that there exists $j = 2, \dots, N+1$ so that $n_j^T g_1 > 0$, then system (17) initialized at v_1 (or very close to v_1 on F_j) will leave the simplex without hitting F_1 (by continuity). Therefore, $n_j^T g_1 \leq 0$, $\forall j = 2, \dots, N+1$. Similarly, for an arbitrary $j = 2, \dots, N+1$, $n_k^T g_j \leq 0$, $\forall k = 2, \dots, N+1$, $k \neq j$ because otherwise there will exist points close to v_j on F_k leaving the simplex. It is obvious that we need to have $n_1^T f(x) > 0$ everywhere on the exit facet F_1, which implies $n_1^T g_j > 0$, $\forall j = 2, \dots, N+1$. The only thing that remains to be proved is $n_1^T g_1 > 0$. Assume by contradiction that $n_1^T g_1 \leq 0$. According to Lemma 1, n_1 is a negative linear combination of $n_2, \dots, n_{n+1}$ and we can write $n_1 = \sum_{i=2}^{N+1} \mu_i n_i$, where $\mu_i < 0$, $i = 2, \dots, N+1$. This leads to $\sum_{i=1}^{N+1} \mu_i n_i^T g_1 \leq 0$. However, we have already proved that $n_i^T g_1 \leq 0$, for all $i = 2, \dots, N+1$, from which we conclude that $\mu_i n_i^T g_1 = 0$, for all $i = 2, \dots, N+1$. Since $n_2, \dots, n_{n+1}$ are linearly independent, it follows that $g_1 = 0$, *i.e.,* the vector field at the vertex v_1 is zero. This means that the system initialized at v_1 will stay there forever, and, therefore will not leave the simplex in finite time, which contradicts the hypothesis, and the Proposition is proved. A related proof of this result can be found in [12], [4] ■

Remark 3: The conditions of Proposition 2 guarantee that the trajectories of (17) leave the simplex S_N through F_1 first time they hit F_1.

The following Proposition characterizes all affine vector fields for which the simplex is an invariant:

Proposition 3 (Stay inside a simplex): There exists an affine vector field (17) on S_N whose trajectories never leave S_N if and only if the polyhedral sets V_j^s, $j = 1, \dots, N+1$ are nonempty, where

$$V_j^s = V \bigcap \bar{V}_j^s,\ j = 1, \dots, N+1, \tag{23}$$

with

$$\bar{V}_j^s = \{g \in \mathbb{R}^N | n_i^T g \leq 0,\ i = 1, \dots, N+1,\ i \neq j\}. \tag{24}$$

Proof: The proof is a simpler version of that given for Proposition 2, and it is omitted. ■

Remark 4: Each polyhedral set $V_k^{e,s}$, $k = 1, \dots, N+1$ corresponds to a set of linear inequalities that has to be satisfied by the value g_k of the vector field f at vertex v_k. Moreover, these sets of linear inequalities are decoupled, *i.e.,* V_k^e and V_k^s depend only on g_k, $k = 1, \dots, N+1$. If one of the sets from Propositions 2 and 3 is empty, then there is no affine vector field in S_N satisfying the corresponding property. If they are all nonempty, then any choice of $g_i \in V_i^{e,s}$, $i = 1, \dots, N+1$ will give a valid (*i.e.,* bounded, as in (17)) affine vector field by formula (13).

Proposition 4: (i) The sets $\bar{V}_i^e$, $i = 1, \dots, N+1$ have a nonempty intersection with any open neighborhood of the origin in $\mathbb{R}^N$. (ii) The intersection of any two sets $\bar{V}_i^s$, $i = 1, \dots, N+1$ is the origin of $\mathbb{R}^N$.

Proof: It is easy to see that, in Proposition 2, $\bar{V}_1^e \subseteq \bar{V}_j^e$ (which also implies $V_1^e \subseteq V_j^e$), for all $j = 2, \dots, N+1$. Therefore, it is enough to prove (i) for $\bar{V}_1^e$. Let

$$C = \{g \in \mathbb{R}^N | n_j^T g \leq 0, j = 2, \dots, N+1\}.$$

It is easy to see that C is a cone with apex 0. Also,

$$\bar{V}_1^e = C \setminus \{0\}, \tag{25}$$

i.e., $\bar{V}_1^e$ is the cone C from which the apex has been removed. Indeed, any $g \in \bar{V}_1^e$ satisfies $g \in C \setminus \{0\}$ since $n_1^T g > 0$ guarantees $g \neq 0$. Therefore, $\bar{V}_1^e \subseteq C \setminus \{0\}$. For an arbitrary $g \in C \setminus \{0\}$, by Lemma 1, $n_1^T g = \sum_{i=2}^{N+1} \mu_i n_i^T g$, where $\mu_i < 0$, $i = 2, \dots, N+1$. Each term in this sum is larger or equal to zero. The sum can therefore be equal to zero if and only if each term is zero, which implies $n_i^T g = 0$, for all $i = 1, \dots, N+1$. This can only happen if $g = 0$ since n_i, $i = 2, \dots, N+1$ are linearly independent by Lemma 1. But $g \neq 0$, therefore $C \setminus \{0\} \subseteq \bar{V}_1^e$. (25) is proved which immediately implies (i).

For (ii), let $i, j \in \{1, \dots, N+1\}$, $i \neq j$. If $g \in \bar{V}_i^s \bigcap \bar{V}_j^s$, then $n_k^T g \leq 0$, for all $k = 1, \dots, N+1$. Since by Lemma 1 n_1 is a negative linear combination of $n_2, \dots, n_{N+1}$, it follows that $n_1^T g = \sum_{i=2}^{N+1} \mu_i n_i^T g$, with $\mu_i < 0$, $i = 2, \dots, N+1$. The left hand side of this equality is ≤ 0, while the right hand side is ≥ 0, and since $n_2, \dots, n_{N+1}$ are linearly independent, it follows that $g = 0$ and (ii) is proved. ■

Proposition 5 (Constant vector fields): (i) There exists a constant vector field (17) satisfying the requirements of Proposition 2 if and only if V_1^e is nonempty. (ii) There does not exist a nonzero constant vector field (17) satisfying the requirements of Proposition 3.

Proof: There exists a constant vector field satisfying the requirements of Propositions 2 or 3 if and only if $\bigcap_{i=1,\dots,N+1} V_i^e \neq \emptyset$ or $\bigcap_{i=1,\dots,N+1} V_i^s \neq \emptyset$, respectively. Indeed, $f(x) = g$, where g is an arbitrary element from the intersection, solves the Problems. This being said, (i) follows immediately from the observation that $V_1^e \subseteq V_j^e$, for all $j = 2, \dots, N+1$ and (ii) is an obvious consequence of Proposition 4 (ii). ■

Therefore, as expected, there will never exist a non-zero constant vector field keeping system (17) inside the simplex for all times. See Figure 2 for an illustration of these ideas for the particular case of $N = 2$, *i.e.,* the simplexes are triangles.

Proposition 6: (i) There exists a solution to Proposition 2 for an arbitrary simplex if and only if V contains an open neighborhood of the origin in $\mathbb{R}^N$. (ii) There exists a solution to Proposition 3 for an arbitrary simplex if and only if V contains the origin in $\mathbb{R}^N$.

Proof: The sufficiency for (i) is immediate from Proposition 4 (i). For the sufficiency of (ii), if V contains the origin, then all sets V_i^s contain it, so the zero vector field solves Proposition 3. For necessity, assume by contradiction that V does not contain the origin, not even on the boundaries. Since V is convex, there exists a hyperplane, say H, passing through the origin which leaves V on one side. Consider a simplex with facet F_1 contained in H and outer normal n_1 oriented on the opposite side of V. For such a simplex, all sets V_k^e, $k = 1, \dots, N+1$ are empty, because they are all contained in $\{g \in \mathbb{R}^N | n_1^T g > 0\}$, which has an empty intersection with V. This contradicts that there is a solution to Proposition 2 and (i) is proved. If we now consider a simplex whose facet F_1 is contained in H with outer normal n_1 oriented towards the hyperspace containing V, then all the sets V_k^s, $k = 2, \dots, N+1$ are empty because they are all contained in $\{g \in \mathbb{R}^N | n_1^T g \leq 0\}$, which has an empty intersection with V. This contradicts that there is a solution to Proposition 3 and (ii) is proved. ■

C. Construction of triangular affine hybrid systems

For the particular case of $N = 2$, Proposition 6 leads to the following Corollary, which is the main result of this paper.

Corollary 1: For an arbitrary triangulation of a polygon $\mathcal{P}$ (3), there exist a hybrid system HS (5) with affine vector fields (10) producing the same language as the corresponding dual graph, *i.e.,* $\mathcal{L}(HS) = \mathcal{L}(DG)$, if and only if the set V giving the polyhedral bounds of the vector fields contains an open neighborhood of the origin in $\mathbb{R}^2$.

Note that, if the condition of Corollary 1 is satisfied, then for each location $q_{ij} \in \mathcal{Q}$, there exists a whole set of vector

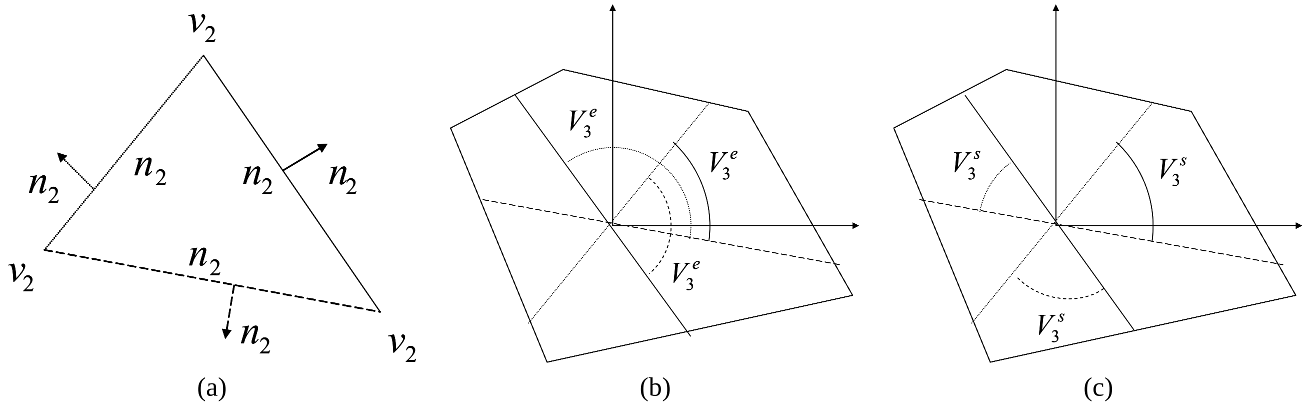

Fig. 2. When $N = 2$, the simplex defined by Equation (11) is a triangle shown in (a). For this example, the sets V_1, V_2, V_3 from Propositions 2 and 3 are the portions of cones shown in (b) and (c), respectively. The bounding polygon represents the polyhedral set V as in (17).

fields $f_{q_{ii}}$ keeping the system in the triangle $I(l_i)$. Each choice of $g_k \in V_k^s$ given in Proposition 3 will lead to a different vector field in $I(l_i)$ according to formula (17). Similarly, for each location q_{ij} there exists a whole set of vector fields $f_{q_{ij}}$ driving the system from triangle $I(l_i)$ to its neighbor $I(l_j)$, and each choice of $g_k \in V_k^e$ as in Proposition 2 will lead to a different vector field in $I(l_i)$ according to formula (17).

In the next Section, we present an algorithm for automatic generation of unique vector fields implementing an arbitrary string in the language $\mathcal{L}(DG)$.

IV. Algorithms for automatic generation of unique vector fields

In this Section, we will use the extra degrees of freedom present in the characterization of the vector fields in Corollary 1 to *guarantee smoothness of the produced trajectories*, if possible, and *minimize the time required for the accomplishment of a task* specified in terms of a string $(l_{i_1}, l_{i_2}, \ldots, l_{i_m}) \in \mathcal{L}(DG)$.

To simplify the notation and without restricting the generality, assume that an arbitrary string in $\mathcal{L}(DG)$ is denoted by $(l_1, l_2, \ldots, l_m)$. To execute it, from the hybrid system HS, we need to select the locations $q_{12}, q_{23}, \ldots, q_{(m-1)m}, q_{mm}$. Any of the corresponding vector fields $f_{q_{12}}, f_{q_{23}}, \ldots, f_{q_{(m-1)m}}, f_{q_{mm}}$ will definitely accomplish the task, as discussed in the previous Section. However, even though the produced trajectories will be smooth inside each triangle, this property will in general be lost when transiting between adjacent triangles. Smoothness of trajectories is guaranteed everywhere in $\bigcup_{i=1}^m I(l_i)$ if and only if the vector fields $f_{q_{(i-1)i}}$, $f_{q_{i(i+1)}}$ match on the separating facet $I(l_{i-1}) \bigcap I(l_i)$ for all $i = 2, \ldots, m-1$ and the vector fields $f_{q_{(m-1)m}}$, $f_{q_{mm}}$ match on $I(l_{m-1}) \bigcap I(l_m)$. This guarantees the continuity of the vector field everywhere in $\bigcup_{i=1}^m I(l_i)$ and therefore the produced trajectories are C^1 (differentiable with continuous derivatives), or smooth. Using Lemma 2 and noting that the separating facets are S_1 triangles (or line segments), the matching condition everywhere on a separating facet is satisfied if and only it is satisfied at the vertices. This implies that matching can be achieved for a whole sequence if and only if all the polyhedral sets obtained as solutions of Propositions 2 or 3 for a given point, which can be a vertex of several triangles, have nonempty intersection.

In what follows, we present an algorithm that takes as input a set of points and a relation assigning these points to a sequence of pairwise adjacent triangles and outputs a set of vector fields guaranteeing smoothness of the corresponding trajectories in as large as possible subsequences of triangles. Let $p_1, \ldots, p_{m+2} \in \mathbb{R}^2$ denote the coordinates of the vertices of triangles $I(l_1), \ldots, I(l_m)$ in a reference frame $\{F\}$. Let $\mathcal{A} \subset \{1, \ldots, m+2\} \times \{1, \ldots, m\} \times \{1, 2, 3\}$ be a relation describing the assignment of the points $p_1, \ldots, p_{m+2} \in \mathbb{R}^2$ as vertices of the triangles $I(l_1), \ldots, I(l_m)$ with the following significance: $(i, j, k) \in \mathcal{A}$ means that p_i is a vertex of triangle $I(l_j)$ with rank k, which we denote by v_k^j. The rank k, $k = 1, 2, 3$ of a vertex v_k^j of triangle $I(l_j)$ is defined as follows. The vertex of rank 1 (v_1^j) of triangle $I(l_j)$, $j = 1, \ldots, m-1$ is not a vertex of $I(l_{j+1})$. For $I(l_m)$, the vertex of rank 1 (v_1^m) does not belong to triangle $I(l_{m-1})$. Ranks 2 and 3 (v_2^j and v_3^j) are defined so that if $(i_1, j, 1), (i_2, j, 2), (i_3, j, 3) \in \mathcal{A}$, then p_{i_1}, p_{i_2}, and p_{i_3} are coordinates of vertices of triangle $I(l_j)$ in counterclockwise order. See Figure 3 (a) and (c) for two examples of point-vertex assignment using the notation described above. Corresponding to this assignment of vertices, for each triangle $I(l_j)$, $j = 1, \ldots, m$, we define three facets F_k^j with outer normals n_k^j, where F_k^j is the facet opposite to vertex v_k^j of triangle $I(l_j)$, $k = 1, 2, 3$.

Let P_i, $i = 1, \ldots, m+2$ denote the polyhedral set for point p_i and V_k^j the polyhedral set obtained by applying Propositions 2 or 3 to the vertex v_k^j of triangle $I(l_j)$. In Table I, we present an algorithm that takes as input the set of coordinates $\{p_1, \ldots, p_{m+2}\}$ and the triangle-vertex relation $\mathcal{A}$ and returns the maximal subsequence $\{1, \ldots, j_1\}$ of triangle indexes for which matching conditions can be satisfied, *i.e.*, smooth trajectories can be achieved. The main idea is the following: the triangles are visited in the given order starting from $j = 1$ and restrictions V_k^j are added to sets P_i corresponding to the points p_i which act as vertices v_k^j corresponding to Propositions 2 or 3. When in a given triangle j the set P_i corresponding to a point becomes empty, then we stop, set $j_1 = j$ and keep the nonempty sets P_i from the previous step, which guarantees that smooth trajectories

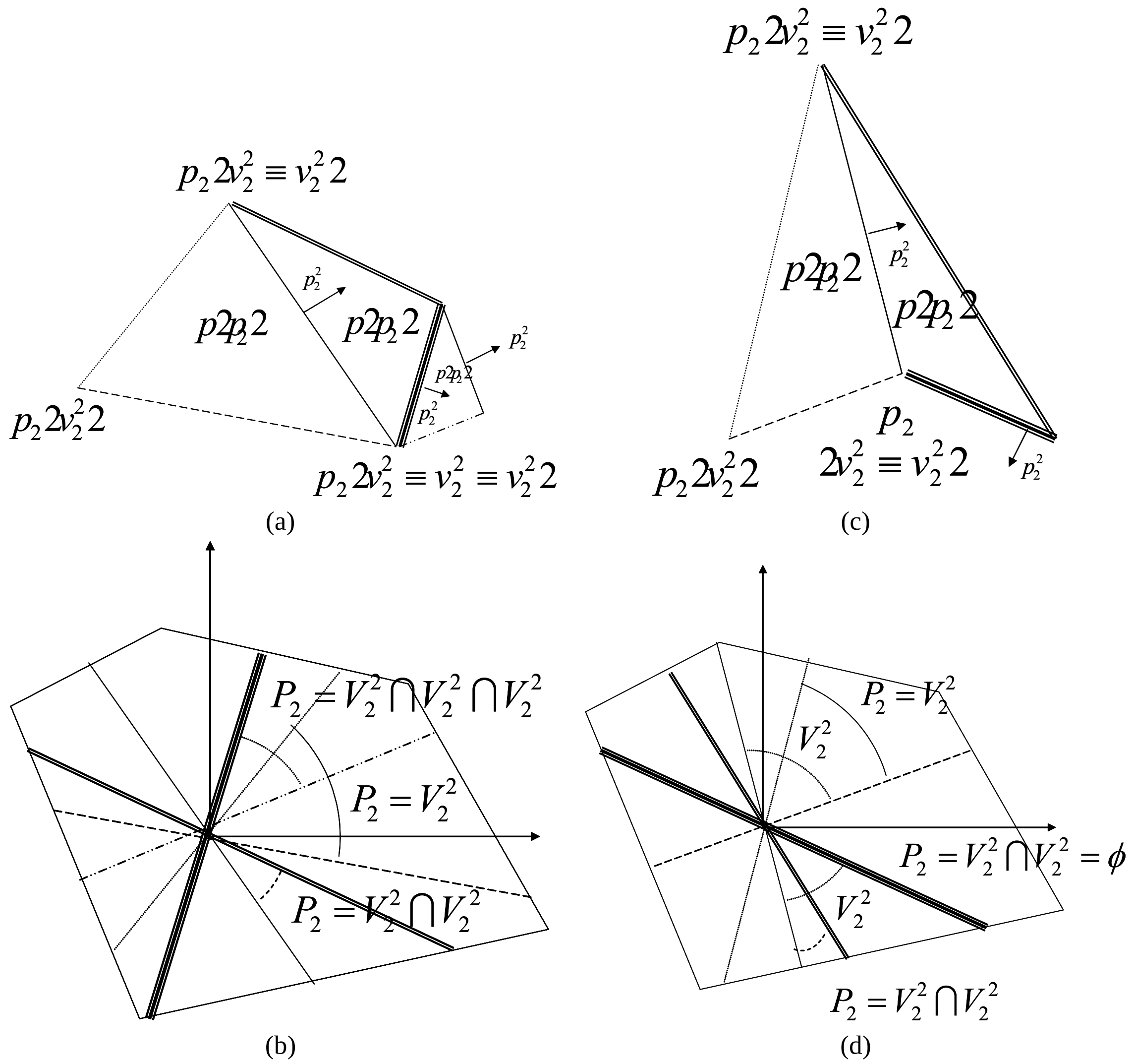
(a)
(c)
(b)
(d)
$P_2 = V_2^2 \bigcap V_2^2 \bigcap V_2^2$
$P_2 = V_2^2$
$P_2 = V_2^2 \bigcap V_2^2$
$P_2 = V_2^2 \bigcap V_2^2 = \phi$
V_2^2

Fig. 3. Examples of adjacent triangle sequences: (a) and (b) show an example where the matching condition can be satisfied, (c) and (d) illustrate a situation when matching is not possible.

bring all initial conditions from triangle $I(l_1)$ to $I(l_{j_1})$ in finite time. Then the algorithm can be reiterated starting from j_1 to produce another subsequence and finally provide a solution to Problem 1 with a minimum number of subsequences. Of course, at the facet separating $I(l_{j_1})$ and $I(l_{j_1+1})$, the vector field will be discontinuous.

There are two important points we need to make with regard to the matching condition. First, as stated in Proposition 5, if Proposition 2 is used in just one triangle and the constraint set V is such that the sets V_1^e, V_2^e, and V_3^e are nonempty, then it is always possible to construct a constant vector field solving the problem based on the fact that $V_1^e \subset V_2^e$ and $V_1 \subset V_3^e$ always. However, if matching is desired with subsequent triangles in a sequence, then the inclusion above might not be valid anymore and affine feedback controllers with explicit state dependence are necessary. A graphical illustration of this ideas is given in Figure 3 (a) and (b), where point p_3 is a vertex of rank 3 in $I(l_1)$ and of rank 1 in $I(l_2)$. If just the problem of reaching facet F_1^1 of $I(l_1)$ was considered, than $V_1^1 \subset V_3^1$. However, if matching is required for the sequence $I(l_1)$, $I(l_2)$, $I(l_3)$, then the allowed set of p_3 is $P_3 = V_3^1 \bigcap V_1^2$, which has an empty intersection with P_1. Therefore, the affine vector field $f_{q_{12}}$ in $I(l_1)$ cannot be chosen constant anymore.

Second, for the particular case of triangles in plane that we consider in this paper, there is a simple geometrical interpretation of the matching condition: it is violated if and only if there exists a sequence of adjacent triangles which ”rotates” around a common vertex with more than π. See Figure 3 (c),(d) for a graphical illustration of such a situation.

To minimize the time spent on the produced trajectories, from the polyhedral sets P_i corresponding to each point p_i in a subsequence where the matching condition is satisfied, we select a velocity vector which has a maximum projection along a weighted sum of all outward normals of all exit facets of which the point is a vertex. This problem is a linear program and has a unique solution. A lower bound for the projection of velocity at vertices along a constant vector is a lower bound for the projection of the affine vector field everywhere in the triangle by the convexity property of Lemma 2. The algorithm shown in Table II also returns the corresponding vector fields which guarantee smoothness of trajectories in the subsequence and maximization of speed.

```
Determine maximal smooth subsequence ($p_1, p_2, \ldots, p_{m+2}, \mathcal{A}$)
$P_i \leftarrow \mathbb{R}^2$, for all $i = 1, \ldots, m+2$ (* initialize polyhedral sets for all points *)
$j \leftarrow 1$ (* start with first triangle *)
while $P_i$ is nonempty, for all $i = 1, \ldots, m+2$
  $v_k^j \leftarrow p_i$, for all $(i,j,k) \in \mathcal{A}$ (* identify the vertices of triangle $j$ *)
  if $j = m$ (* last triangle *)
    apply Proposition 3 in triangle $j$ with vertices $v_k^j$ to obtain $V_k^j$, $k = 1,2,3$
  else (* not last triangle *)
    apply Proposition 2 in triangle $j$ with vertices $v_k^j$ to obtain $V_k^j$, $k = 1,2,3$
  endif
  $P_i \leftarrow P_i \bigcap V_k^j$, for all $(i,j,k) \in \mathcal{A}$ (* update allowed polyhedral sets for $p_i$'s which are vertices of triangle $j$ *)
  $j \leftarrow j+1$ (* move to the next triangle *)
endwhile
$S \leftarrow \{1, 2, \ldots, j-1\}$ (* sequence of triangles for which smooth trajectories can be designed *)
(* $P_i$, $(i,j,k) \in \mathcal{A}$, $j \in S$ are nonempty polyhedral sets for the points which are vertices
of the triangles in sequence $S$ *)
```

TABLE I

ALGORITHM FOR DETERMINING A MAXIMAL SEQUENCE OF TRIANGLES FOR WHICH SMOOTH TRAJECTORIES CAN BE GENERATED.

```
Construction of continuous vector fields ($S$)
for all $i$ so that $(i,j,k) \in \mathcal{A}$ and $j \in S$ do
  $c_i \leftarrow \sum_{j \in S, (i,j,k) \in \mathcal{A}} n_1^j$ (* sum of all outer normals to exit facets to which $p_i$ is a vertex *)
  the velocity $g_i$ at $p_i$ is the solution to the following LP: $\max_{g_i} c_i^T g_i$, $g_i \in P_i$
endfor
for all $j \in S$ do
  using (13) construct $f_{q_{j(j+1)}}(x)$ so that $f_{q_{j(j+1)}}(v_k^j) = g_i$, $(i,j,k) \in \mathcal{A}$
endfor
```

TABLE II

ALGORITHM FOR CONSTRUCTION OF VECTOR FIELDS IN A SMOOTH SEQUENCE OF TRIANGLES.

V. ROBOT CONTROL

In this section, we show how the computational framework developed above can be used for automatic generation of provably correct robot control laws for motion plans specified in terms of strings in the language of a dual graph describing the triangulation of a polygon, as required in Problem 1. As already suggested in Section II, we consider two types of planar robots: fully actuated with control bounds and unicycles with bounded driving and steering controls.

A. Fully-actuated kinematic planar robot

Following the notation introduced in Section II, the state q of a fully actuated kinematic robot is its position vector in a world frame, which coincides with its observable x. For such a robot, its velocity is directly controllable, *i.e.*, the robot is described by:

$$\dot{q} = u, q \in \mathcal{P} \subset \mathbb{R}^2, \ u \in U \subseteq \mathbb{R}^2 \tag{26}$$

where $\mathcal{P}$ is a polygon and U is a polyhedral set capturing control (velocity) constraints. In this case, the feedback controllers solving Problem 1 are given by the vector fields of the hybrid system constructed as shown in the previous sections. From Corollary 1, we have the following:

Corollary 2: For a fully actuated robot (26) with polyhedral control bounds U, there exists a solution to Problem 1 for arbitrary polygons and triangulations if the polyhedral set U contains an open neighborhood of the origin in $\mathbb{R}^2$.

Note that the necessary and sufficient condition in Corollary 1 becomes sufficient in the above Corollary, since the results only hold for the class of affine feedback control systems. Also, the condition of Corollary 2 is in accordance with one's intuition: if the robot is able to move in all directions, then it can execute arbitrary strings. However, the robot can execute certain strings under affine feedback even if the above condition is not satisfied. The equivalent conditions and analytical formulas for automatic generation of feedback control laws are presented in the previous Sections.

An example showing the assignment of maximally smooth vector fields in a sequence of adjacent triangles and corresponding simulated trajectories is shown in Section VI.

B. Unicycle

Consider a differentially driven wheeled robot as the one shown in Figure 4. In the world frame $\{F\}$, the robot is described by $(R,d) \in SE(2)$, where $d \in \mathbb{R}^2$ gives the position vector of the robot center and $R \in SO(2)$ is the rotation of the robot frame $\{M\}$ in $\{F\}$.

The control $u = [u_1, u_2]^T \subset U \subseteq \mathbb{R}^2$ consists of driving (u_1) and steering (u_2) speeds, where U is a set capturing control bounds. The kinematics of the robot are described by the well known equations of the unicycle:

$$\dot{d} = R \begin{bmatrix} u_1 \\ 0 \end{bmatrix} \tag{27}$$

$$\dot{R} = RE_1 u_2 \tag{28}$$

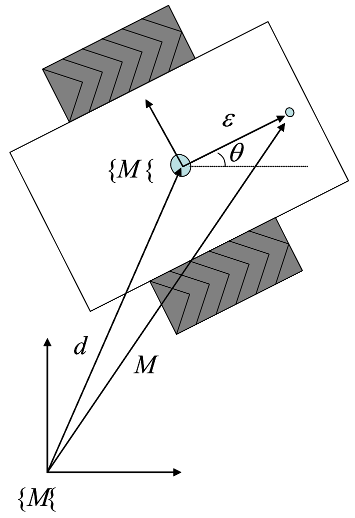

Fig. 4. Unicycle.

where E_1 is defined in equation (32).

If the 1-dimensional rotation R is parameterized by $\theta \in [0, 2\pi)$, *i.e.*,

$$R(\theta) = \left[\begin{array}{cc} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{array} \right], \tag{29}$$

then equation (28) is obviously equivalent to $\dot{\theta} = u_2$. Following the notation from Section II, the state of the robot is therefore $q = \{[\theta, d^T]^T\}$.

It is well known that the under-actuated system (27), (28) with state (θ, d) and control $u = [u_1, u_2]^T$ is uncontrollable [14]. For this reason, as in [11], we define a reference point different from the robot center and with coordinates $(\epsilon, 0)$ in the robot frame $\{M\}$ (see Figure 4). The coordinates $x = [x_1, x_2]^T$ of this reference point (or observable, as defined in equation (2)) in the world frame $\{F\}$ are used to formulate the motion planning tasks. Using frame transformation rules, we have:

$$x = R \left[\begin{array}{c} \epsilon \\ 0 \end{array} \right] + d, \tag{30}$$

which, by differentiation with respect to time, and using (27) and (28), becomes:

$$\dot{x} = RE_2 u \tag{31}$$

where E_2 is defined by:

$$E_1 = \left[\begin{array}{cc} 0 & -1 \\ 1 & 0 \end{array} \right], \; E_2 = \left[\begin{array}{cc} 1 & 0 \\ 0 & \epsilon \end{array} \right] \tag{32}$$

Note that equations (27), (28), (30), and (31) represent an input output feedback linearization problem [14] for the system with state (R, d), input u, and output x. The next step, as usually in such a problem, is to define a feedback control law $v(x)$ so that the system evolving corresponding to

$$\dot{x} = v(x) \tag{33}$$

satisfies given requirements specified in terms of the output x. Such requirements usually include stabilization to a point, when a proportional (P) controller is enough, and trajectory tracking, when a proportional derivative (PD) controller is necessary. The original controls u driving the robot so that the specifications in terms of x are met are eventually found using

$$u = E_2^{-1} R^T v(x) \tag{34}$$

It is easy to see that the map in (34) is well defined whenever $\epsilon \neq 0$, i.e., the reference point used to specify the task is different from the robot center.

As in the fully actuated case, $v(x)$ is determined by the construction of the hybrid system (5), and particular strings can be implemented as shown in Section IV. Using equation (34), it is easy to see that bounds V on the velocity of the reference point v easily translate to bounds U on the original control u, by noting that they are related by a rotation and a scaling factor dependent on ϵ. If V is polyhedral, then U is guaranteed to be contained in an ellipse obtained by rescaling the disc determined by applying all planar rotations to V. If $v(x)$ is constant, *i.e.*, V is a point, then U is an ellipse. However, in practice, the bounds U are usually imposed. The set V used in our algorithms will then be a polyhedron contained in the image of U through the map (34).

By applying Corollary 2 to the reference point x and using (34), we have the following:

Corollary 3: For a unicycle (27), (28) with control bounds U, there exists a solution to Problem 1 for arbitrary polygons and triangulations if the set U contains an open neighborhood of the origin in $\mathbb{R}^2$.

Indeed, for any set U containing the origin of $\mathbb{R}^2$, one can always find a polyhedral set V containing the origin whose image through given positive scaling and all planar rotations is included in U. Again, the intuition works here as well: a unicycle can execute arbitrary strings over the dual graph induced by a triangulation of its polygonal observable space if it can rotate both left and right and translate both forward and backward.

VI. SIMULATION RESULTS

Consider a unicycle with driving and steering speeds u_1 and u_2 limited to 1 and 2, respectively. In other words, $U = [-1, 1] \times [-2, 2]$. Assume that the displacement of the reference point in the unicycle frame is $\epsilon = 0.5$ (see Figure 4). Then, it is easy to see that, with a bit of conservatism, the rectangular bounds $V = [-\sqrt{2}/2, \sqrt{2}/2]^2$ for the reference point will guarantee the imposed control bounds U. Indeed, under all planar rotations R, V becomes a disk centered at 0 with radius 1, which is then scaled to an ellipse with semi-axes 1 and 2, according to (34). The actual controls of the robot are inside an ellipse centered at 0 with semi-axes 1 and 2, which is contained in the rectangle U. Therefore, the initial control bounds are guaranteed if V is chosen as above.

A. Simple environment

To illustrate the assignment of vector fields and the satisfaction of matching conditions and control bounds, we first consider a simple polygonal environment consisting of the sequence of adjacent triangles shown in Figure 5. This example can be also be interpreted as the execution of a string from the language of a dual graph of larger triangulated polygon. Δ_1 denotes the initial triangle and Δ_{14} is the final triangle.

By applying the algorithm given in Table I, we determined that the maximal smooth sequence starting at Δ_1 is $\Delta_1, \Delta_2, \ldots, \Delta_9$, with stop in Δ_9. Indeed, it is easy to see that, if exit through the common facet of Δ_9 and Δ_{10} was desired, then the rotation around the common vertex of Δ_7, Δ_8, Δ_9, and Δ_{10} would be larger than π. The produced vector fields guaranteeing smooth motion in the sequence $\Delta_1, \Delta_2, \ldots, \Delta_9$ are plotted in Figure 6 (a). Then the algorithm is reiterated, and the vector fields corresponding to the next smooth sequence $\Delta_9, \Delta_{10}, \ldots, \Delta_{14}$, with stop in Δ_{14}, are shown in 6 (b). Note that the vector fields on adjacent triangles match on the separating facet in each of the subsequences shown in Figure 6 (a) and (b).

The motion of a unicycle arbitrarily initialized in Δ_1 is shown in Figure 5. The corresponding velocity v of the reference point x and the controls u are shown in Figure 7 (a) and (b), respectively. It is easy to see that each component of v and u are continuous everywhere, except for a time close to 2500, when the vector field in Δ_9 is switched from a stopping one as in Figure 6 (a) to a driving one as in Figure 6 (b). Also, note that the polyhedral bounds for v and u are satisfied for all times during the produced motion.

B. Complex environment

To illustrate the computational efficiency of the developed algorithms and the utility of the created framework, we consider a more realistic example as the one shown in Figure 8. The outer polygonal line represents the boundaries of the environment, while the inner closed polygonal lines model obstacles. The obtained polygon, which has 44 vertices, was triangulated using the algorithm available at [30]. The resulting triangulation, which consists of 46 triangles, and the corresponding dual graph are shown in Figure 8. Sample trajectories of the unicycle described at the beginning of this Section implementing strings in the language of this dual graph are shown in Figure 9.

Note that the triangulation procedure is computationally inexpensive, since it scales linearly with the number of vertices [5]. The generation of vector fields is done according to the algorithms described in Tables I and II. For a sequence of adjacent triangles in which smooth trajectories can be generated, we solved a number of linear programs equal to the number of polygon vertices pertaining to the triangles. The number of linear constraints in each of these LPs varies, and depends on how many triangles in the sequence share the corresponding vertex.

VII. Conclusion

In this paper we proposed a method for algorithmically generating and verifiably composing affine feedback control laws that solve various robot motion planning problems. In addition to being computationally efficient, our solution formally relates the high level plans and low level motions using modern tools from hybrid systems theory. Future work includes extensions on the discrete side resulting in more complicated plans such as temporal logic planning [40] or games on graphs [17]. On the continuous side we will extend the framework towards more complicated dynamics. This may force us to reconsider the discrete abstraction used for planning, but even if it does, our goal is to provide formal relationships between the discrete abstraction and the continuous model, leading to planning verified by construction.

References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Oliviero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[2] Rajeev Alur, Tom Henzinger, Gerardo Lafferriere, and George J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(2):971–984, jul 2000.

[3] A. Astolfi. Discontinuous control of nonholonomic systems. *IEEE Transactions on Automatic Control*, 27:37–45, 1996.

[4] C. Belta and L.C.G.J.M. Habets. Constructing decidable hybrid systems with velocity bounds. In *43rd IEEE Conference on Decision and Control*, Bahamas, 2004.

[5] B. Chazelle. Triangulating a simple polygon in linear time. *Disc. Comput. Geom.*, 6:485–524, 1991.

[6] H. Choset, K. M. Lynch, L. Kavraki, W. Burgard, S. A. Hutchinson, G. Kantor, and S. Thrun. *Robotic Motion Planning: Foundations and Implementation.* 2004. In preparation.

[7] D. C. Conner, A. A. Rizzi, and H. Choset. Composition of local potential functions for global robot control and navigation. In *Proceedings of the IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, 2003.

[8] D. C. Conner, A. A. Rizzi, and Howie Choset. Construction and automated deployment of local potential functions for global robot control and navigation. Technical Report CMU-RI-TR-03-22, Carnegie Mellon University, Robotics Institute, Pittsburgh, Pennsylvania, 2003.

[9] J.-M. Coron. Global asymptotic stabilization for controllable systems without drift. *Mathematics of Control, Signals and Systems*, 5:295–312, 1991.

[10] C. Canudas de Wit and O.J. Sordalen. Exponential stabilization of mobile robots with nonholonomic constraints. *IEEE Transactions on Automatic Control*, 13(11):1791–1797, 1992.

[11] J. Desai, J.P. Ostrowski, and V. Kumar. Controlling formations of multiple mobile robots. In *Proc. IEEE Int. Conf. Robot. Automat.*, Leuven, Belgium, 1998.

[12] L.C.G.J.M. Habets and J.H. van Schuppen. A control problem for affine dynamical systems on a full-dimensional polytope. *Automatica*, 40:21–35, 2004.

[13] E. Haghverdi, P. Tabuada, and G. Pappas. *Bisimulation relations for dynamical and control systems*, volume 69 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2003.

[14] A. Isidori. *Nonlinear Control Systems, 3rd ed.* Springer-Verlag, London, 1995.

[15] V. Isler, C. Belta, K. Daniilidis, and G. J. Pappas. Stochastic hybrid control for visibility-based pursuit-evasion games. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sentai, Japan, 2004.

[16] V. Isler, S. Kannan, and S. Khanna. Locating and capturing an evader in a polygonal environment. In *Workshop on Algorithmic Foundations of Robotics (WAFR'04)*, 2004.

[17] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion with limited visibility. In *Proc. of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1053–1063, 2004.

[18] L.E. Kavraki, P. Svetska, J. C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[19] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5:90–98, 1986.

[20] D. E. Koditschek. The control of natural motion in mechanical systems. *ASME Journal of Dynamic Systems, Measurement, and Control*, 113(4):548–551, 1991.

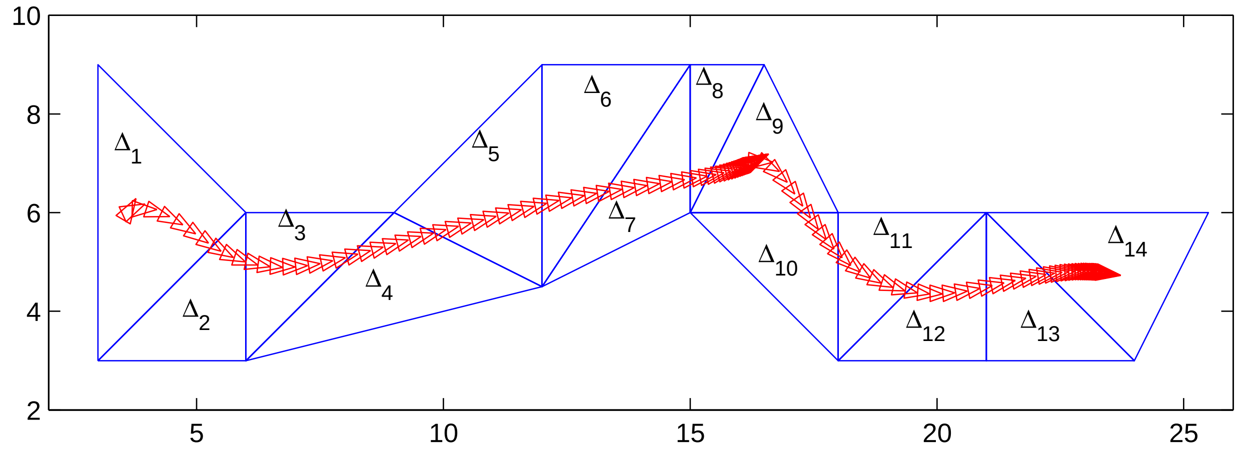

Fig. 5. A sequence of adjacent triangles and an example of unicycle motion.

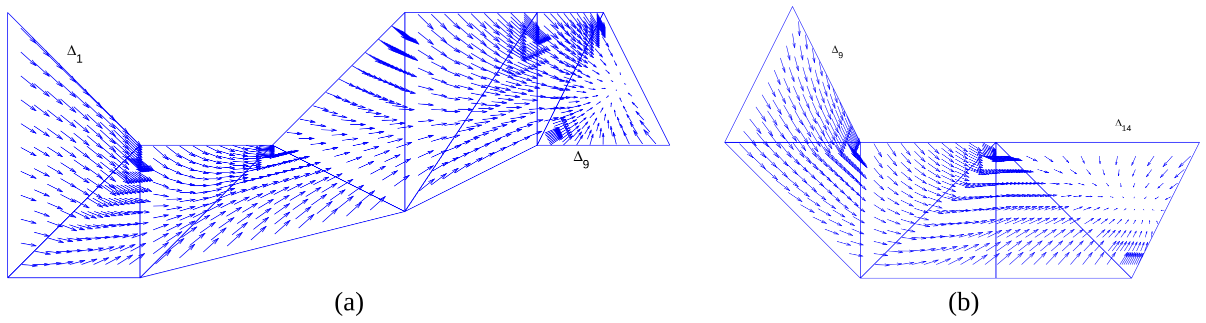

Fig. 6. The vector fields obtained by applying the algorithms in Tables I and II to the sequence of triangles given in Figure 5: (a) smooth sequence $\Delta_1, \Delta_2, \ldots, \Delta_9$, with stop in Δ_9; (b) smooth sequence $\Delta_9, \Delta_{10}, \ldots, \Delta_{14}$, with stop in Δ_{14}.

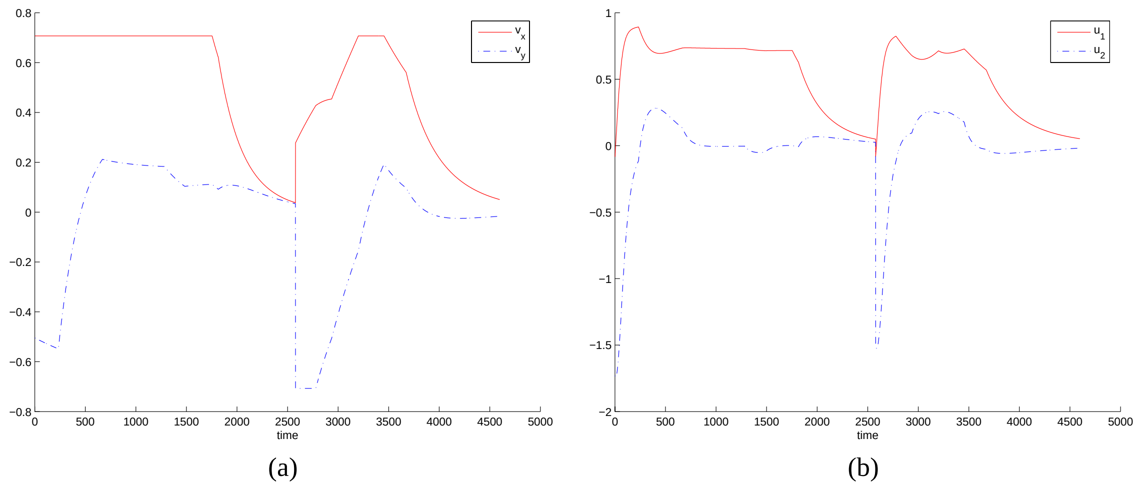

Fig. 7. (a) Time evolution of reference point velocity v; (b) Time evolution of driving and steering controls u_1 and u_2.

[21] B. H. Krogh. A generalized potential field approach to obstacle avoidance control. In *SME Conf. Proc. Robotics Research: The Next Five Years and Beyond*, Bethlehem, Pennsylvania, 1984.

[22] G.A. Lafferriere and E.D. Sontag. Remarks on control lyapunov functions for discontinuous stabilizing feedback. In *Proc. of the 32rd IEEE Conference on Decision and Control*, page 23982403, Austin, TX, December 1993.

[23] G.A. Lafferriere and H. Sussmann. A differential geometric approach to motion planning. In Z. Li and J.F. Canny, editors, *Nonholonomic Motion Planning*, page 235270. Kluwer Academic Publishers, 1993.

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

[25] S. M. LaValle. *Planning Algorithms*. [Online], 1999-2004. Available at http://msl.cs.uiuc.edu/planning/.

[26] S. M. LaValle and M. S. Branicky. On the relationship between classical grid search and probabilistic roadmaps. In *Workshop on the Algorithmic Foundations of Robotics*, Nice, France, 2002.

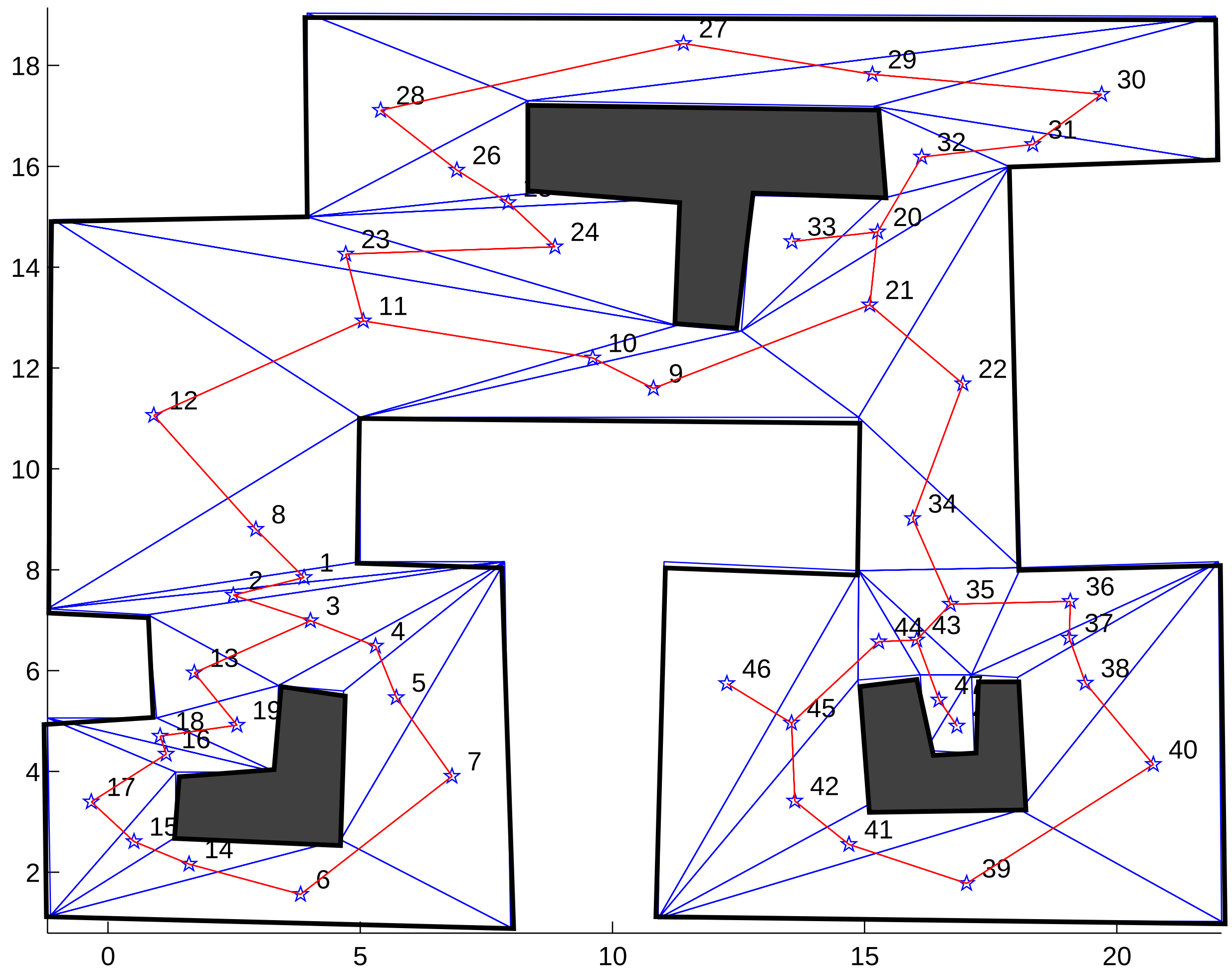

Fig. 8. A polygonal environment, its triangulation and the dual of the triangulation.

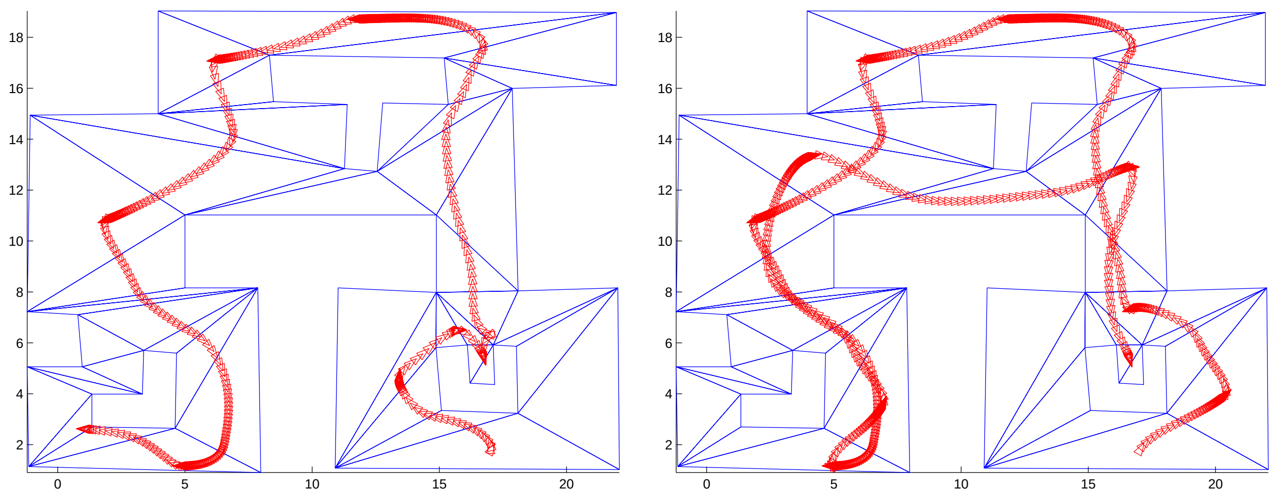

Fig. 9. Sample trajectories.

[27] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[28] P. Morin and C. Samson. Control of nonlinear chained systems: From the routh-hurwitz stability criterion to time-varying exponential stabilizers. *IEEE Transactions on Automatic Control*, 45(1):141–146, 2000.

[29] R.M. Murray and S.S. Sastry. Nonholonomic motion planning : Steering using sinusoids. *IEEE Transactions on Automatic Control*, pages 700–716, 1993.

[30] A. Narkhede and D. Manocha. Fast polygon triangulation based on seidel's algorithm.

[31] J. O'Rourke. *Computational Geometry in C.* Cambridge University Press, 1998.

[32] G. J. Pappas. Bisimilar linear systems. *Automatica*, 39(12):2035–2047, 2003.

[33] D. M. R. Park. *Concurrency and automata on infinite sequences*, volume

104 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[34] J.-P. Pomet. Explicit design of time-varying stabilizing control laws for a class of controllable systems without drift. *Systems and Control Letters*, 18:147–158, 1992.

[35] A. Quaid and A. A. Rizzi. Robust and efficient motion planning for a planar robot using hybrid control. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 4021–4026, 2000.

[36] E. Rimon and D. E. Kodischek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, 1992.

[37] E. Rimon and D. E. Koditschek. The construction of analytical diffeomorphisms for exact robot navigation on star worlds. *Transactions of the American Mathematical Society*, 327(1):71–115, 1991.

[38] A. A. Rizzi. Hybrid control as a method for robot motion programming. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 832–837, 1998.

[39] 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*, page 27002705, Austin, TX, December 1993.

[40] P. Tabuada and G. Pappas. Model checking ltl over controllable linear systems is decidable. volume 2623 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[41] H. G. Tanner, S. Loizou, and K. J. Kyriakopoulos. Nonholonomic navigation and control of cooperating mobile manipulators. *IEEE Transactions on Robotics and Automation*, 19(1):53–64, 2003.

[42] D. Tilbury and A. Chelouah. Steering a three input nonholonomic system using multirate controls. In *Proc. of the European Control Conference*, page 19931998, Groningen, Netherlands, 1992.

[43] D. Tilbury, R. Murray, and S. Sastry. Trajectory generation for the ntrailer problem using goursat normal forms. In *Proc. of the 32rd IEEE Conference on Decision and Control*, pages 971–977, Austin, TX, December 1993.