Symbolic Robot Motion Planning Using Discrete Abstractions in Polygonal Environments

Behzad Sarfaridpour **Maziar Fooladi Mahani**
Email:bsadrfa@g.clemson.edu Email:mfoolad@g.clemson.edu
Department of Mechanical Engineering
Clemson University
Clemson, SC, 29634

I. Introduction

This project presents a ground mobile robot motion-planning problem. A computational framework for automatic generation of control laws for robot is used to map the continuous motion planning specified in terms of triangles to computationally inexpensive state transition systems. The state transition system is constructed by using Delaunay triangulation method for the environment where each triangle represents one state of the system and movement between each two adjacent triangles is denoted as a transition. After making the transition system from the triangulated environment, the path for the robot is found by using model checking techniques, for which the specifications are defined as to reach goals while avoiding obstacles. By using the property of affine functions in convex hulls, we are able to construct the vector fields that drive the robot through the triangles on the specified trajectory and stops the robot when all goals are reached. These results are then used to generate provably correct control laws for fully actuated kinematic robot and uncycles.

This report is organized as follows. Section II states the problem formulation and approach followed by symbolic discrete motion planning in section III that explains how the discrete paths are determined using model checking approach. Next, section IV presents the continuous motion planning using triangular affine functions for which the implementation is the most contribution of this work. At the end, the implementation and simulation results are provided and two major extention directions are proposed.

II. Problem Formulation and Approach

We consider a planar robot described in coordinates by control system 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, $q = x \in R^2$ 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 ν and ω , we have $q = \{[\,x^T, \theta]\,^T \,|\, x \in R^2, \theta \in [\,0, 2\pi)\}$, $u = (\nu, \omega) \in R^2$, $F(q,u) = [\,cos\theta\nu \;\; sin\theta\nu \;\; \omega]\,^T$.

Each state of the discrete state space is a triangle and we label each triangle using a finite set of symbols $L = \{l_1, l_2, ..., l_M\}$ and denote the region for triangle l_i by $I(l_i)$.

Problem 1: Construct a set U of state feedback controllers so that, for any string $(l_i1, l_i2, ..., l_im) \in L(DG)$, there exists $u \in U$ driving the robot (1) from any initial state $q_0 \in Q$ so that its observable x moves through the regions $I(l_{i1}), I(l_{i2}), ..., I(l_{im})$ in finite time, and stays in $I(l_{im})$ for all future times.

III. Symbolic Discrete Motion Planning

For discrete path planning, model checking techniques are utilized. Model checking method as a sub-branch of formal methods brings together mathematically based languages, techniques and tools for verifying system behaviors. Symbolic robot motion planning based on model checking generates a counterexample to the negation of the task specification for a robot represented by its transition system model. The transition system encompasses all possible movements of the robot between any two adjacent triangles of the triangulated environment. The system specifications are expressed as temporal logic formulas. The temporal logic language used in this project is Linear Temporal Logic (LTL), where the specifications of this system are defined as reaching all goals while avoiding obstacles and are expressed by the following LTL formula:

$$\phi = \bigwedge_{i \in Goals} \Diamond \pi_i \wedge \bigwedge_{i \in Obstacles} \Box \neg \pi_i$$

where π_i is the proposition that labels goals and obstacles.

The model checker tool used in this project is called NuSMV, which allows for the representation of finite state systems, and for the analysis of specifications expressed in Computation Tree Logic (CTL) and Linear Temporal Logic (LTL), using BDD-based and SAT-based model checking techniques (Cimatti et al. 2002). In order to use discrete logics to reason about continuous systems, we need a finite partition of the continuous state space and this is carried out by using Triangle tool (Shewchuk 1996), which triangulates convex hulls using Delaunay method.

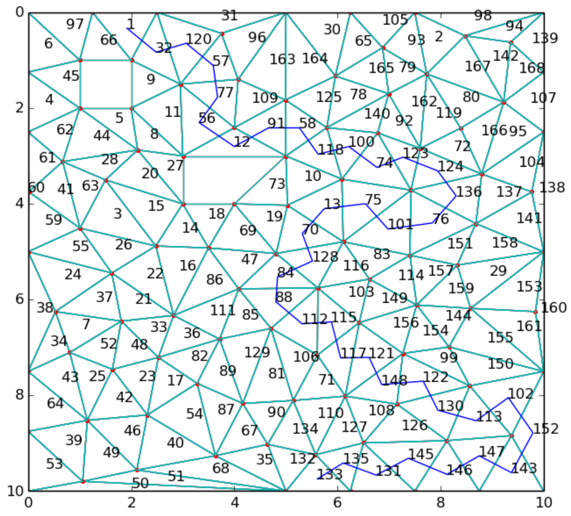


Figure 1: Triangulation and state order.

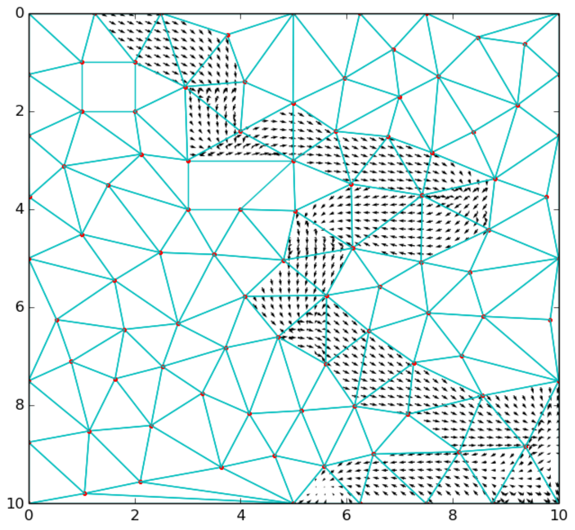


Figure 2: Vector fields for all of the patches.

IV. Continuous Motion Planning

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

Affine Functions in Convex Hulls

Affine functions have an interesting property in polygons (Belta, Isler, and Pappas 2005): it is uniquely determined by its values at the vertices of the polygon and its restriction to the polygon is a convex combination of these values and is given by:

$$f(x) = GW^{-1}\begin{bmatrix} x \\ 1 \end{bmatrix} \tag{2}$$

where,

$$G = [g_1\ g_2\ g_3] \tag{3}$$

and

$$W = \begin{bmatrix} v_1\ v_2\ v_3 \\ 1\ \ 1\ \ 1 \end{bmatrix} \tag{4}$$

are 2×3 and 3×3 real matrices. g_i is the robot velocity at vertice i and v_i is the corresponding vertice postion.

Affine Feedback Control Laws in Convex Hulls

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 state in a triangle through a desired facet in finite time or making the triangle an invariant, which keeps the robot inside a triangle. The following proposition gives a characterization of all affine vector fields that drives the robot from any initial condition in a triangle through a facet in finite time.

Proposition 1(Exit through a facet): There exists an affine vector field driving all inital states in the triangle through the facet F_1 in finite time if and only if the polyhedral sets $V_j^e, j = 1, 2, 3$ are nonempty (Belta, Isler, and Pappas 2005), where

$$V_1^e = V \cap \hat{V}_1^e \tag{5}$$

$$V_j^e = V \cap \hat{V}_j^e, j = 2, 3 \tag{6}$$

with,

$$\hat{V}_1^e = \{g \in R | n_j^T g \leqslant 0, j = 2, 3, and, n_1^T g > 0\} \tag{7}$$

$$\hat{V}_2^e = \{g \in R | n_j^T g \leqslant 0, j = 3, and, n_1^T g > 0\} \tag{8}$$

$$\hat{V}_3^e = \{g \in R | n_j^T g \leqslant 0, j = 2, and, n_1^T g > 0\} \tag{9}$$

Next, there are two cases that we need to keep the robot inside a triangle. First where there is no smooth vector field that drives the robot on a certain specified path which intuitively happens with sharp turns. In this case we need to split the path into two or more smooth paths and the robot needs to stop in the last triangle of each smooth path patch and turns with no forward velocity towards the next vector field for the next smooth path.. The second case is where the robot is driven into the very last triangle and no more transition is needed and the robot stops there and all the goals are achieved. The following proposition characterizes all affine vector fields for which the triangle is an invariant:

Proposition 2(Stay inside a triangle): There exists an affine vector field whose trajectories never leave the triangle if and only if the polyhedral sets $V_j^s, j = 1, 2, 3$ are nonempty (Belta, Isler, and Pappas 2005), where

$$V_j^s = V \cap \hat{V}_j^s, j = 1, 2, 3 \tag{10}$$

with,

$$\hat{V}_j^s = \{g \in R | n_i^T g \leqslant 0, i = 1, 2, 3,\ and,\ i \neq j\} \tag{11}$$

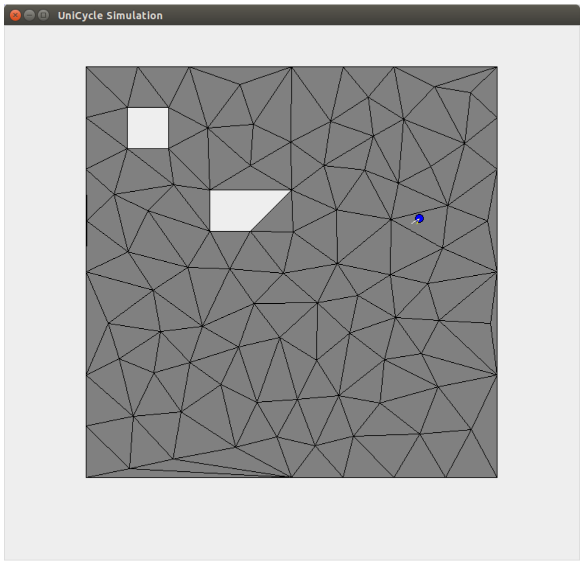


Figure 3: Simulation for a planar unicycle robot.

V. Implementation

For implementation, first we define in Triangle tool (Shewchuk 1996) the polygons that represent the whole environment for the robot with the obstacles that are known a priori. The triangulated environment will then be used to build the transition system. NusMV, the model checker takes the transition system as an input along with the system specifications written in LTL and outputs the set of the states over the triangles. Then by applying the propositions 1 & 2, we are able to construct the vector field for the entire path from starting point to the last goal. The constructed vector fields are then used to generate the control laws for a fully actuated kinematic robot and a unicycle.

VI. Simulation Results

To illustrate the assignment of vector fields, we simulated a fully-actuated and an under-actuated robot in an environment with some obstacles. The robot needs to visit the triangles = [120, 74, 124, 13, 115, 143, 133] without colliding with obstacles. The configuration space and the model checker output which is the sequence of adjacent triangles is shown in Figure 1. Figure 4 illustrates the constructed vector field for the sequence of triangles. Figure 3 shows a simulation snapshot for a unicycle.

VII. Future Works

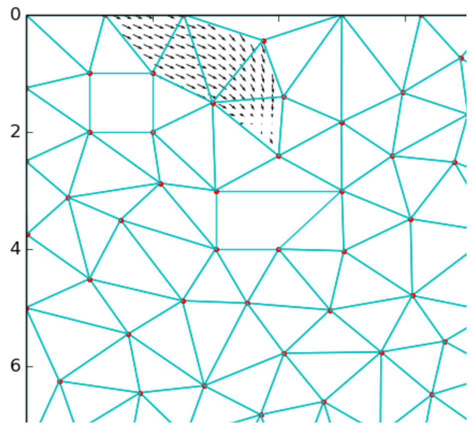
We propose two directions for extending this work. First, considering unknown obstacles in the configuration space for which the robot needs to replan its path towards the unvisited goals iteratively as it detects new obstacles. The second extension is to having decentralized multi-agent system for which local collsion avoidance is then essential.

References

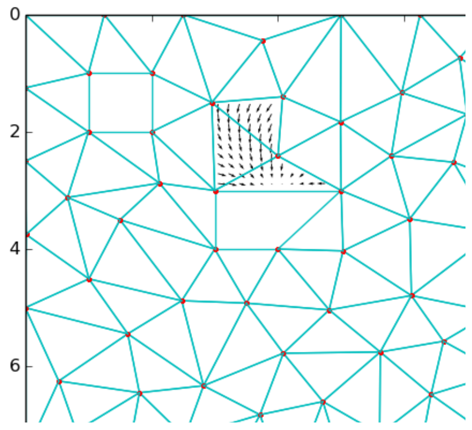
Belta, C.; Isler, V.; and Pappas, G. J. 2005. Discrete abstractions for robot motion planning and control in polygonal environments. *IEEE Transactions on Robotics* 21(5):864–874.

Cimatti, A.; Clarke, E.; Giunchiglia, E.; Giunchiglia, F.; Pistore, M.; Roveri, M.; Sebastiani, R.; and Tacchella, A. 2002. NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking. In *Proc. International Conference on Computer-Aided Verification (CAV 2002)*, volume 2404 of *LNCS*. Copenhagen, Denmark: Springer.

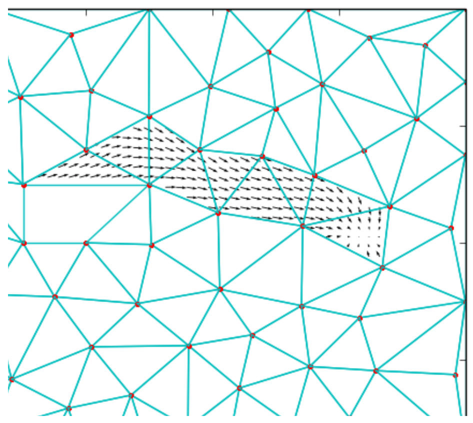
Shewchuk, J. 1996. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. *Applied computational geometry towards geometric engineering* 203–222.



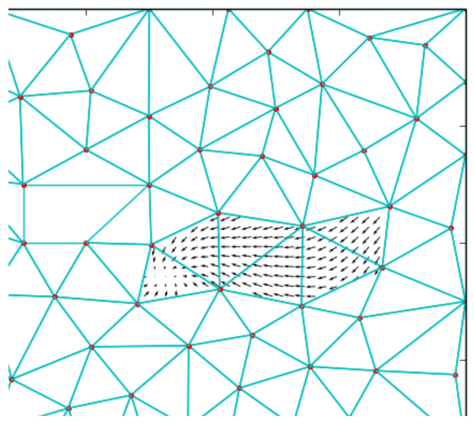
(a)



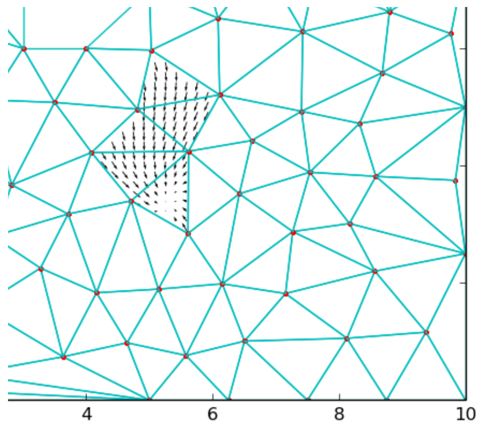
(b)



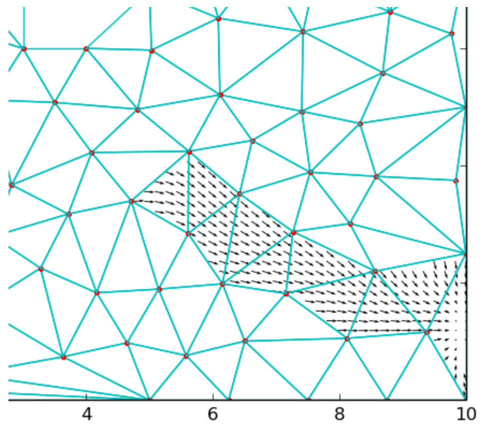
(c)



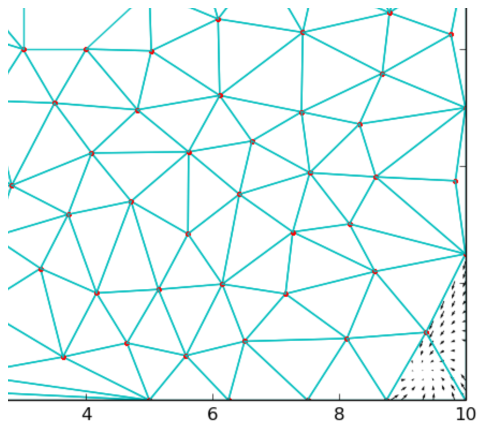
(d)



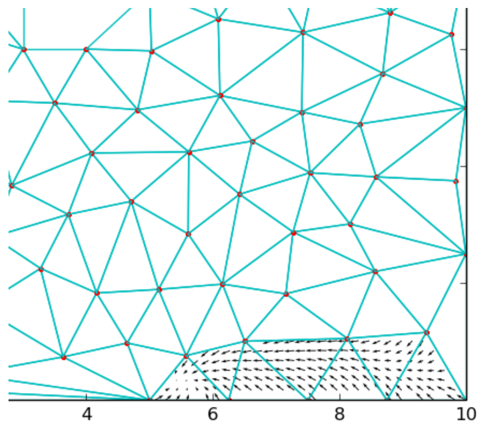
(e)



(f)



(g)



(h)

Figure 4: Vector fields for each patch.