Robotics: Science and Systems 2006
Philadelphia, PA, USA, August 16-19, 2006

Integrated Planning and Control for Convex-bodied Nonholonomic systems using Local Feedback Control Policies

David C. Conner, Howie Choset and Alfred A. Rizzi
Carnegie Mellon University, Robotics Institute
Pittsburgh, Pennsylvania 15213
Email: {dcconner+,choset+,arizzi+}@ri.cmu.edu

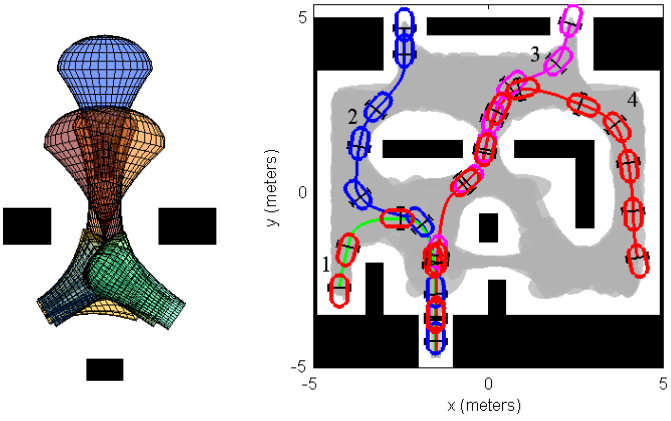
***Abstract*— We present a method for defining a hybrid control system capable of simultaneously addressing the global navigation and control problem for a convex-bodied wheeled mobile robot navigating amongst obstacles. The method uses parameterized continuous local feedback control policies that ensure safe operation over local regions of the free configuration space; each local policy is designed to respect nonholonomic constraints, bounds on velocities (inputs), and obstacles in the environment. The hybrid control system makes use of a collection of these local control policies in concert with discrete planning tools in order to plan, and replan in the face of changing conditions, while preserving the safety and convergence guarantees of the underlying control policies. This work is validated in simulation and experiment with a convex-bodied wheeled mobile robot. The approach is one of the first that combines formal planning with continuous feedback control guarantees for systems subject to nonholonomic constraints, input bounds, and non-trivial body shape.**

I. Introduction

The problem of simultaneously planning and controlling the motion of a convex-bodied wheeled mobile robot in a cluttered planar environment is a challenging problem because of the relationships among control, nonholonomic constraints, and obstacle avoidance. The objective is to move the robot through its environment such that it arrives at a designated goal set without coming into contact with any obstacle, while respecting the nonholonomic constraints and velocity (input) bounds inherent in the system.

Conventional approaches to addressing this problem typically decouple the navigation and control problem [1], [2], [3]. First, a planner finds a path, and then a feedback control strategy attempts to follow that path. Non-trivial robot body shapes complicate the problem of finding a safe path, as the path must be specified in the free configuration space of the robot. This leaves the challenging problem of designing a control law that converges to the path, while remaining safe with respect to obstacles.

We address the coupled navigation and control problem by generating a vector field along which the robot can "flow." Unfortunately, determining a global vector field that satisfies all of these objectives for constrained systems can be quite difficult. Our approach, inspired by the idea of *sequential composition* [4], uses a collection of local feedback control policies coupled with a switching strategy to generate the vector field. Composing these relatively simple policies induces a piecewise continuous vector field over the union of the policy domains. A *policy* is defined as a configuration dependent vector field over a local region of the robot's free configuration space, termed a *cell*. The vector field flow over each local cell leads to the specified policy goal set within the cell. Figure 1-a shows some of the cells, which are defined in three-dimensional configuration space, projected into the x-y plane. Figure 1-b shows four paths induced by invoking the local policies for four different initial conditions in the robot experiments, as will be discussed in Section V.



a) Cell Projections b) Experimental Runs

Fig. 1. a) Projection of three-dimensional cells in the x-y plane (work space). b) Experimental results of four robot runs using the proposed hybrid control framework. By chaining the policies sequentially together, the system can be brought to an overall goal. The paths are induced by the switched policies, which are switched according to the ordering determined by a discrete planner. An explicit desired path is never calculated. The projection of the individual policy domains are shown in light gray.

A discrete transition relation represented by a graph is induced by the transition between the domain of one policy to the domain of a second policy containing the policy goal set of the first. On-line planning, and re-planning under changing conditions, becomes more tractable on the graph, allowing us to bring numerous discrete planning tools to bear on this essentially continuous problem. By sequencing the local policies according to the ordering determined by a discrete

planner via graph search, the closed loop dynamics induce the discrete transitions desired by the discrete plan. The overall hybrid (switched) control policy responds to system perturbations without the need for re-planning. In the face of changing environmental conditions, the discrete graph allows for fast online re-planning(reordering), while continuing to respect the system constraints.

By coupling planning and control in this way, the hybrid control system plans in the discrete space of control policies; thus, the approach represents a departure from conventional techniques. A "thin" path or trajectory is never explicitly planned; instead, the trajectory induced by the closed-loop dynamics flows along the vector field defined by the active policy, which is chosen according to an ordering determined by a discrete planner. A plan over the discrete graph associated with the collection of policies corresponds to a "thick" set of configurations within the domains of the associated local policies.

This approach offers guaranteed convergence over the union of the local policy domains. For the method to be complete, the entire free configuration space must be covered by policy domains (cells). This is a difficult problem due to the multiple constraints on the robot dynamics. Even constructing the free configuration space in the first place is a difficult problem. This latter difficulty is avoided by deploying policies using workspace measurements to guarantee that the policy domain lies in the free configuration space. Herein, the focus is on deploying a "rich enough" collection of policies that enables the navigation problem to be addressed over the bulk of the free configuration space.

This paper describes the basic requirements that any local policy must satisfy to be deployed in this framework; each local policy must be provably safe with respect to obstacles and guarantee convergence to a specified policy goal set, while obeying the system constraints within its local domain. We develop a set of generic policies that meet these requirements, and instantiate these generic policies in the robot's free configuration space. Finally, a concrete demonstration of controlling a real mobile robot using a collection of local policies is described.

II. Related Work

Our approach maintains the coupling between planning and control, unlike conventional approaches that decouple the problem into a path planning problem and a path-following control problem. If the path planning does not consider the nonholonomic constraints, this decoupled approach often leads to highly oscillatory paths that require great control effort [5], [6]. Motion planning approaches that do consider nonholonomic constraints typically assume a discrete set of feasible motions, and search for a shortest feasible path composed of arcs of feasible motions [7], [8], [9], [10]. In another decoupled example for nonholonomic systems, open-loop steering methods address the point-to-point navigation problem in the absence of obstacles [5], [11].

The proposed hybrid control approach is based on the technique of sequential composition [4]. This technique enables the construction of switched control policies that have guaranteed behavior, and provable convergence properties. In its original form, sequential composition uses positively invariant state regulating feedback control policies; the policy domains of attraction are used to partition the state space into cells.

The overall control policy induced by sequential composition is fundamentally a hybrid control policy [12], [13]. The method of composition based on the relationship between policy goal sets and domains obviates the need for complex stability analysis of the form given in [14]. The stability of the underlying control policies guarantees the stability of the overall policy [4]. Disturbances are robustly handled provided their magnitude and rate of occurrence is relatively small compared to the convergence of the individual policies.

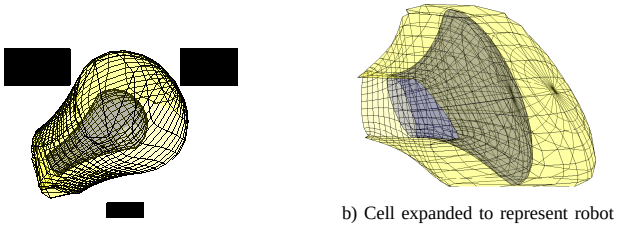
Sequential composition-like techniques have been applied to mobile robots. Many examples are for systems without nonholonomic constraints, where the policies are defined over simple cells – polytopes and balls – in configuration space [15], [16]. Sequential composition has also been used to control wheeled mobile robots using visual servoing [17], [18]. In these cases, the local control policies were designed based on careful analysis of the system, its constraints, and the problem at hand. Other researchers have focused on automated deployment of generic policies for idealized (point) fully-actuated systems in simulation [19], [20], [21].

III. General Framework

As the robot moves through its planar workspace, its configuration evolves on the manifold $SE(2)$; the configuration is locally represented as $q = \{x, y, \theta\} \in \mathcal{Q} \approx SE(2)$. We assume a kinematic model of the system, meaning the system velocity is directly controlled. The nonholonomic constraints inherent in wheeled mobile robots determine a linear mapping $A(q) : \mathcal{U} \rightarrow T_q\mathcal{Q}$ between the control input $u \in \mathcal{U} \subset \mathbb{R}^2$ and the configuration velocity; that is $\dot{q} = A(q)\, u$ naturally respects the nonholonomic constraints. The input space $\mathcal{U}$, assumed to be a bounded subset of $\mathbb{R}^2$ without holes, represents the control inputs available to the system.

We address the navigation and control problem for these systems by developing a collection of feedback control policies, over local regions of configuration space that are termed cells. Let Φ_i denote the i^{th} policy in the collection, and let $\Xi_i \subset \mathcal{Q}$ denote the policy's associated cell. Define the cells in a local region of $\mathbb{R}^3$ that represents a local chart of the configuration space manifold $SE(2)$; the cells are restricted to compact, full dimensional subsets of $\mathbb{R}^3$ without holes. It is assumed that the boundary of the cell, $\partial\Xi_i$, has a well defined unit normal, $n(q)$, that exists almost everywhere. The policy goal set is defined within the closure of each cell. The feedback control policy is defined by a mapping $\Phi_i : \Xi_i \rightarrow \mathcal{U}$ such that the flow along the vector field induced by $A \circ \Phi_i : \Xi_i \rightarrow \mathcal{T}_q\mathcal{Q}$ leads to the designated goal set.

This paper restricts discussion to a new class of "flow-through" policies [20], where the goal set is defined on the



a) Extent of robot body in workspace

b) Cell expanded to represent robot body extent

Fig. 2. The cell boundary (dark inner surface) may be expanded to account for the non-trivial robot shape. The minimum signed distance from the projected silhouette curve of the expanded cell to the closest obstacle is the closest approach of the system to an obstacle under the control of the associated policy. Note, the open face of the goal set is not shown. The goal set will be contained within the domain of another policy, so expanding this set is unnecessary.

boundary of the cell. Other types of policies can be readily added to the proposed framework provided they meet the general conditions described below.

A. Generic Policy Requirements

In order for a local policy to be valid, it must have several properties. First, to induce safe motion, the cell must be contained in the free configuration space of the robot so that collision with any obstacle is not possible while the configuration is within this cell. Second, under the influence of the policy, the induced trajectories must reach the goal set without departing the associated cell from any initial configuration within the cell. Third, the system must reach the designated goal set in finite time for any initial condition within the cell. Finally, to be practical, the cells must have efficient tests for point inclusion.

The first condition is that the cell must be contained in the free configuration space, $\mathcal{FS}_{\mathcal{Q}}$, of the convex-bodied robot; that is $\Xi_i \subset \mathcal{FS}_{\mathcal{Q}} \subset \mathcal{Q}$. Any configuration within a valid cell is free of collision with an obstacle. Testing that the cell is contained in $\mathcal{FS}_{\mathcal{Q}}$ would seem to require constructing $\mathcal{FS}_{\mathcal{Q}}$, and then performing tests on this complicated space. This complexity can be avoided by testing the cells based on workspace measurements. Consider the composite set of points in the workspace obtained by taking the union, over the set of cell boundary configurations, of the points occupied by the robot body. The cell is contained in free configuration space if no points in this composite set intersect an obstacle (see Figure 2-a).

For a given cell, this composite set can be determined analytically by defining a surface that accounts for the the maximum extent of the robot body at each configuration on the cell boundary (Figure 2-b). This surface, which can be thought of as an expansion of the given cell, is projected into the workspace (Figure 2-a). If no points in the surface projection intersect an obstacle, then the cell lies in the free configuration space of the system, and the system cannot collide with an obstacle while remaining in the cell. Given a piecewise analytic representation of the robot body boundary and a parametric representation of the cell boundary, an exact piecewise parametric representation of the expanded cell surface can be determined directly. See [22] for details.

The second condition is a generalization of *positive invariance*, termed *conditional* positive invariance [23]. Whereas earlier versions of sequential composition were restricted to convergent control policies with positive invariant domains, we allow policies that cause the system to *flow-through* a designated goal set and not come to rest within the goal set. For a conditionally positive invariant domain, the configuration remains in the domain of the policy until it enters the designated goal set. Coupled with the fact that the cell is contained in free configuration space, conditional positive invariance guarantees the policy is safe with respect to collision.

To maintain conditional positive invariance, the system must be able to generate a velocity that, while satisfying the system constraints, keeps the system configuration within the cell and away from the boundary. On the cell boundary away from the goal set, the velocities are restricted to the negative half-space defined by the outward pointing unit normal at the boundary point; that is, there must exist a $u \in \mathcal{U}$ such that

$$n(q) \cdot A(q)\, u < 0\,, \tag{1}$$

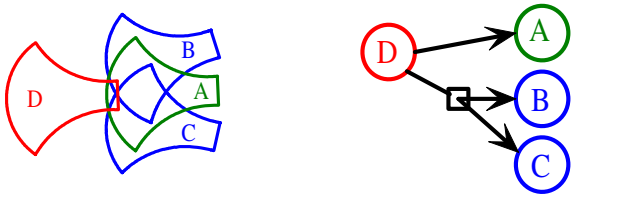
where $A(q)$ encodes the nonholonomic constraints. For flow-through policies, the aim is to to drive the system configuration through the goal set; hence, some points in the goal set must satisfy the analogous necessary condition $n(q) \cdot A(q)\, u > 0$.

The third requirement is that a valid policy must bring any initial configuration within the cell to the specified goal set in finite time. In general, determining this for constrained systems can be difficult and must be considered for each type of policy in conjunction with its cell and goal set specification.

In summary, policies that respect the system constraints, have simple inclusion tests, are completely contained in the free configuration space, are conditionally invariant, and whose vector field flow converges to a well defined goal set in finite time may be deployed in this sequential composition framework. Given a specific system model and input set $\mathcal{U}$, these conditions limit the size and shape of the associated cell in the free configuration space. In Section IV, we describe a family of policies that satisfy these requirements.

B. Discrete Abstraction

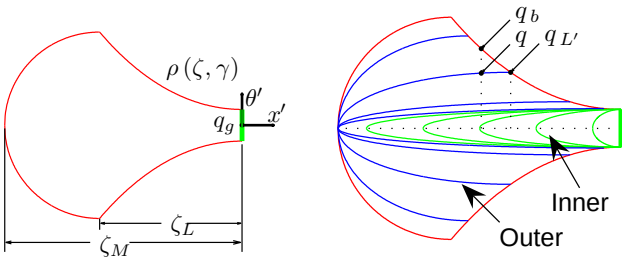
Sequential composition defines a specific relationship between the policies. Finite time convergence coupled with conditional positive invariance induces a transition relation between a given policy domain (cell) and its associated goal set. This transition relation, coupled with having the goal set of one policy contained in the domain of another policy, induces a discrete transition relationship between the two policies. Given two control policies, Φ_j is said to *prepare* Φ_i, if the goal set of Φ_j is contained in the domain of Φ_i [4]. This prepares relationship, denoted $\Phi_j \succeq \Phi_i$, induces a graph structure by defining a *directed* edge between the two nodes corresponding to the two policies.



a) Prepares relationship between a collection of policies.

b) Graph representation of the induced discrete abstraction.

Fig. 3. This figure shows the relation between a set of policies and the discrete abstraction. In this example, Φ_D prepares Φ_A and $\bigcup\{\Phi_B, \Phi_C\}$.



a) Cell Definition

b) Level Set Definition

Fig. 4. Schematic representations of the generic cell. Some of the important parameters are labeled.

Often the size of a valid cell that satisfies the policy requirements is tied to the size of the specified goal set; therefore, to enable larger cells, it is useful to consider policies where no single policy domain covers the designated goal set. Therefore, we extend the conventional definition of *prepares* given in [4] from a relation between two policies, to a relation between a policy and a set of policies. A selected policy, Φ_i, *prepares* a set of policies if the goal set of the selected policy, $\mathscr{G}(\Phi_i)$, is contained in the union of the domains of the policies in the set, that is $\Phi_i \succeq \{\Phi_j\}$ if $\mathscr{G}(\Phi_i) \subset \bigcup_j \mathscr{D}(\Phi_j)$.

Although the flow along a vector field is mathematically determinate, from the perspective of the discrete transition relation, the single policy could result in a transition to any of the policies in the union. This introduces indeterminacy into the graph structure that can be represented as an *action* with multiple *outcomes*. This relationship between prepares and the discrete abstraction is shown in Figure 3. Given a collection of policies, the specification of the prepares relationship between policies induces a directed, generally cyclic, graph.

The discrete graph abstracts the continuous dynamics of the problem into a finite set of transitions between policies. Thus, a fundamentally continuous navigation problem is reduced to a discrete graph search. If the current configuration is contained in the domain of a policy corresponding to a node the graph, and a path through the graph to the overall goal node exists, then the given navigation problem is solvable with the collection of currently deployed policies. By executing the feedback control policies according to an ordering determined by a discrete planner, the trajectories induced by the local policies solve the specified navigation problem. To facilitate ordering, each edge in the graph is assigned a cost. Given policies that meet the above requirements, the proposed technique is amenable to any number of discrete planning tools, including those that employ model checking and temporal logic specifications [24].

IV. Local Policy Definition

In defining the local policies, there are several competing goals. First, we desire policies with simple representations that are easy to parameterize. Additionally, we desire policies that have simple tests for inclusion so that the system can determine when a particular policy may be safely used. On the other hand, for a given policy goal set, we want the policy domain to capture as much of the free configuration space as possible given the system constraints. That is, we want the policies to be *expressive*. This paper focuses on how far a set of relatively simple parameterizations can be pushed. Starting with a class of parameterized representations for the cell boundary, we specify a vector field that satisfies the policy requirements. Here, the focus is on the general description of the generic cells; refer to [22] for details.

A. Cell Definition

The generic cells are defined relative to a local coordinate frame with coordinate axes $\{x', y', \theta'\}$. We attach the local frame to the goal set center such that the local x'-axis is an outward pointing normal with respect to the policy goal set, and arbitrarily restrict the policy goal set to lie within the y'-θ' plane. The cell is positioned by specifying the goal set center $q_g = \{x_g, y_g, \theta_g\}$ with respect to the world frame such that the θ'-axis is parallel to the world θ-axis and the x'-axis is aligned with the robot direction of travel at q_g. This is shown in Figure 4-a. With this setup, the cells have a central axis defined by the negative x'-axis.

We define the generic cell boundary in local coordinates with *two* smooth two-surfaces embedded in $\mathbb{R}^3$. The intersection of the cell boundary with a plane orthogonal to the central axis is a simple closed curve that may be parameterized by the orientation around the x'-axis. This suggests a cylindrical parameterization for the generic cell boundary. Let ζ be a scalar encoding the depth of the cell along the negative x'-axis, and let γ be a scalar encoding the angle about the local x'-axis. Define a generic cell boundary point, p, in these local coordinates as

$$p(\zeta,\gamma) = \begin{bmatrix} -\zeta \\ \rho(\zeta,\gamma)\cdot(\cos\beta\,\cos\gamma - c\,\sin\beta\,\sin\gamma) \\ \rho(\zeta,\gamma)\cdot(\sin\beta\,\cos\gamma + c\,\cos\beta\,\sin\gamma) \end{bmatrix}, \quad (2)$$

where c is an eccentricity parameter and β specifies a rotation of the policy goal set about the x'-axis. The function $\rho(\zeta,\gamma)$ governs the radius in the local cylindrical coordinate system.

There is freedom in defining $\rho(\zeta,\gamma)$ provided (1) from Section III-A is satisfied for all points on the surface. We choose a ρ function that has two continuous, piecewise-smooth segments that correspond to the two surface patches – a

funnel and a cap – defining the cell boundary. The segments are shown in Figure 4. In the portion corresponding to the funnel, let $\rho_f(\zeta,\gamma)$ be a monotonically increasing function in ζ that governs cell growth as ζ increases away from the policy goal set; $\rho = \rho_f$ in this portion. The portion of $\rho(\zeta,\gamma)$ corresponding to the cap section monotonically decreases from the maximal value of ρ_f to zero at the maximal extent of ζ. Formally, define the complete function as

$$\rho(\zeta,\gamma) = \begin{cases} \rho_f(\zeta,\gamma) & 0 < \zeta \leq \zeta_L \\ \rho_f(\zeta_L,\gamma)\ \frac{\sqrt{(\zeta_M-\zeta_L)^2-(\zeta-\zeta_L)^2}}{\zeta_M-\zeta_L} & \zeta_L < \zeta \leq \zeta_M \end{cases}.$$

The γ term in ρ allows asymmetry into the cells [22]. Note, ρ_f can have several internal parameters that govern the rate of expansion and shape of the cell. These internal parameters provide some freedom to shape the cell to fit its environment.

For the cell to be valid, (1) must be satisfied over the entire surface, and the analogous condition must hold on the policy goal set. For the cell boundary, $n = \frac{D_\zeta p \times D_\gamma p}{\|D_\zeta p \times D_\gamma p\|}$, which is a piecewise smooth function. Restating requirement (1) in terms of the local parameters and an inequality relation,

$$\arg\max_{\zeta,\gamma} \left[\arg\min_{u \in \mathcal{U}} \left[n(\zeta,\gamma) \cdot A\left(p(\zeta,\gamma)\right) u \right] \right] < 0 \qquad (3)$$

must be satisfied. In other words, in the worst case, the system must be able to generate a velocity that is inward pointing with respect to the cell boundary to satisfy the conditional positive invariance requirement. Although non-linear, the function is piecewise smooth and generally "well-behaved" for the mapping $A(q)$ found for most robot models; therefore, it is feasible to verify that (3) is satisfied off-line during the policy deployment phase.

Given a cell, the system must check if the current configuration, q, is inside the cell. Let $\{\zeta_q, \gamma_q, \rho_q\}$ be the coordinate values in the cell's local cylindrical coordinate frame. The corresponding point, $q_b = \{\zeta_q, \gamma_q, \rho_b\}$, on the cell boundary is given by

$$\rho_b = \rho(\zeta_q,\gamma_q)\ \frac{\sqrt{1+c^2-(c^2-1)\cos(2\gamma_q)}}{\sqrt{2}}. \qquad (4)$$

If $\rho_q < \rho_b$ and $0 < \zeta_q < \zeta_M$, the configuration is in the cell.

B. Vector Field Definition

We use a family of level sets based on the parameterization of the cell boundary to define the control vector field that flows to the policy goal set. First consider the case $\zeta_M = 0$ and $\zeta_L = 0$, where the cell boundary corresponds to the policy goal set. Increasing ζ_M, while fixing $\zeta_L = 0$, generates a family of level sets that grow out from the goal; these are termed the *inner* level sets, as shown in Figure 4-b. By fixing ζ_M at its maximum value and increasing ζ_L, the *outer* family of level sets grows; these are also shown in Figure 4-b.

For a configuration within the cell, the corresponding level set – either inner or outer – that passes through that configuration must be determined. These level sets are governed by the cap (second) portion of $\rho(\zeta,\gamma)$. First, rewrite this cap portion in terms of new variables ζ'_M and ζ'_L. The appropriate level set is determined by values for ζ'_M and ζ'_L that satisfy

$$\rho_f(\zeta'_L,\gamma_q)\ \frac{\sqrt{(\zeta'_M-\zeta'_L)^2-(\zeta_q-\zeta'_L)^2}}{\zeta'_M-\zeta'_L} - \rho_q = 0. \qquad (5)$$

For configurations within the inner family of level sets, $\zeta'_L = 0$ and ζ'_M can be determined in closed-form from (5). If the configuration is within the outer family of level sets, then $\zeta'_M = \zeta_M$ and we must determine the value of ζ'_L that satisfies (5). Unfortunately, ζ'_L cannot be found in closed form. Fortunately, (5) is a monotonic function of ζ'_L, which admits a simple numeric root finding procedure. The corresponding level set intersects the cell boundary at $q_{L'}$ as shown in Figure 4-b. Given the values of ζ'_M and ζ'_L which determine the corresponding level set, the level set normal $n(q)$ is defined by a closed-form analytic function.

The level set normal defines a constrained optimization over the input space to determine the input value. In the simplest case, let

$$u^* = \arg\min_{u \in \mathcal{U}} \left[n(q) \cdot A(q)\, u \right] \quad \text{s.t. } n(q) \cdot A(q)\, u < 0. \qquad (6)$$

Given the assumption that the level sets satisfy (3), a solution is guaranteed to exist.

Using u^* as the control input drives the system from the outer level sets to the inner level sets, and then continuously on to the goal. Forcing both the inner and outer level sets to satisfy (3) at every point in the cell simplifies the proof of finite time convergence. By virtue of (3), a solution to (6) is guaranteed to exist and will bring the system configuration to a "more inward" level set; thus, the system moves a finite distance closer to the goal. Although a formal analytic proof is lacking, experience shows that if the outermost level set corresponding to the cell boundary satisfies (3), all interior level sets will also satisfy (3).

Given the input u^*, the reference vector field over the cell is simply $A(q)\,u^*$. In reality, only u^* is important; the vector field is induced by the flow of the system given the control inputs, and is never explicitly calculated.

C. Policy Deployment

To deploy a policy in this sequential composition framework, we must specify the parameters that govern the location, size, and shape of the cell. Given a specification of these parameters, the requirements from Section III-A must be verified. Future work will develop automated deployment methods; we currently use a trial-and-error process to specify parameters.

We begin deploying a policy by specifying that the policy goal set is completely contained within the domain of other policies; this allows the prepares graph to be defined later. First, we specify q_g to lie within the target policy cell (set of cells); this can be verified using (4) above. We initialize the policy goal set by specifying the function $\rho(0,\gamma)$ and elliptical parameters β and c from (2). We test that the policy goal set boundary does not intersect the boundary of a target cell (union of target cells) by testing the minimum distance between a

point on the policy goal set boundary and the corresponding point of the boundary of the target policies. We iteratively adjust the parameters, including the policy goal set center q_g, as needed until the policy goal set is fully contained in the target cell(s). During this iterative process, we also test that the policy goal set is valid with respect to the mapping $A(q)$ and chosen input set.

Given a valid policy goal set, we grow the cell by specifying the parameters of $\rho(\zeta, \gamma)$ that determine the rate of cell expansion and cell shape, including ζ_L and ζ_M. During this process, we must confirm that the cell remains in the free configuration space and that (3) is satisfied. Given these conditions, the requirements that we can reach the goal in finite time and have a test for cell inclusion are automatically satisfied given the methods for defining the cell and vector field. To test that (3) is satisfied, we use numerical optimization to find the maximum value of (3) over the $\{\zeta, \gamma\}$ parameter space. Given a violation of (3), we iteratively adjust the parameters as needed to ensure the requirements are met.

To verify that a policy is safe with respect to obstacles, we use the expanded cell described in Section III-A. In these experiments, we graphically verify that the projection of the expanded cell is free of intersection. We sample the $\{\zeta, \gamma\}$ parameter space with a fine resolution mesh, and calculate the corresponding points on the expanded cell surface. We project the resulting expanded cell surface mesh into the workspace and visually check for obstacle intersection, as shown in Figure 2-b. This step can be automated by determining the mesh edges that contribute to the silhouette curve based on a simple facet normal test, and projecting those edges to the workspace. The distances to closest obstacles can be tested against the maximum error determined by the chosen mesh resolution. The parameters governing cell size and shape are adjusted, and both the collision and (3) tests are repeated.

Once all the policies are deployed, the final step is to specify the prepares graph; this step is completely automated. The automated graph builder checks the prepares relationship for each policy against all other policies in the deployment; that is we test for policy goal set inclusion against other cells or sets of cells. For the policies presented here, the prepares test can be done by verifying that all points on the 1D policy goal set boundary are contained in the domains of neighboring cells. For policies that have a prepares relationship, we assign an edge cost heuristically based on the path length between the goal center and the target cells associated goal center, and a factor for the policies relative complexity.

For the policies deployed in these experiments, this manual deployment process generally takes between two and 30 minutes to fully deploy each policy. The deployment time depends mainly on the tightness of the corridors and sharpness of the turns being navigated. The final graph generation step is fully automated; it took several hours to run for the deployment used in this paper. Once this step is complete, the deployment is ready to use with the robot. The up-front cost of generating the deployment is mitigated by the ability to reuse the policies by planning multiple scenarios on the discrete graph.

Fig. 5. Laboratory robot.

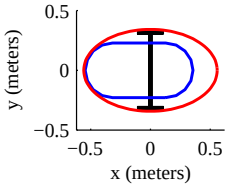


Fig. 6. Bounding ellipse.

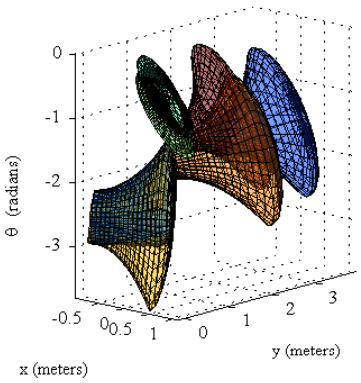


Fig. 7. A three-dimensional view of some of the policies in the current deployment.

V. Current Results

The techniques described in Sections III and IV were validated on our laboratory robot. This standard differential-drive robot, shown in Figure 5, has a convex, roughly elliptical body shape. To simplify implementation of the expanded cell, the robot body and wheels were tightly approximated by an analytic ellipse centered in the body coordinate frame (shown in Figure 6). The car-like kinematic model has the form

$$A(q) = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix}.$$

The inputs are forward velocity (m/s) and turning rate (radians/s). Although the robot is capable of zero-radius turns, the steering rate was limited to model a traditional bounded steering car. This limits the rate of expansion of the cells based on (3); for a zero-radius turn the cell shape would only be limited by the obstacles.

We allow multiple input sets, $\mathcal{U}_i$, to account for changing conditions. The system changes direction by switching between "Forward" and "Reverse" input sets; each having an "Aggressive" and "Cautious" set of values. The numerical values are based on the velocity limits of the motors, and scaled back for cautious modes and the reverse input sets. For computational convenience, this paper is restricted to convex polygonal $\mathcal{U}_i$; this restriction is not fundamental. Each cell is evaluated with respect to a specific choice of $\mathcal{U}_i$.

A. Implementation

For these experiments, we define a set of polygonal obstacles in a $10\,m \times 10\,m$ world, with several corridors and openings as shown in Figure 8. A total of 271 policies were deployed using the techniques described in Section IV; of these, 15 separate policies prepared 31 different unions of policy domains according to the extended definition of prepares. A small number of cells are shown in Figure 7.

Given the collection of deployed policies and the associated graph, the node associated with the policy centered in the

lower corridor shown in Figure 8 is specified as the overall goal of these experiments. An ordering of the policy graph is generated using a Mini-max version of D* Lite [25] as the discrete planner. Initially, the planner generates a global ordering of the deployment graph based on the edge costs determined during graph generation. If a valid path through the ordered graph from a given policy node to the designated goal node does not exist, then the node (and corresponding policy) is flagged as invalid. As the planner executes, the effects of invalid nodes propagate through the graph; as a result, a path to the goal from a given node is guaranteed to exist if the node remains valid. D* Lite provides an efficient way to replan in the case where a policy status is changed to invalid after the initial planning stage; for example if a passage way is found to be blocked as the robot approaches.

During execution, the hybrid policy tests the current estimate of configuration to see which policy should be executed. Given a configuration estimate, the search begins by testing successors designated by the discrete planner for the previously executed policy; if the current configuration is included in one of these domains, the associated policy is activated. If a designated successor containing the current configuration is not found, and the previously executed policy is still valid, the previous policy continues to execute until the next update cycle. If the previously executed policy is no longer active, whether due to a system perturbation or replanning in response to an environmental change, the current implementation searches the graph according to the ordering defined by the discrete planner.

The policy switching is completely automatic once the initial start command is given. The system stops when it reaches the policy designated as the goal or, if the configuration is not contained within a valid policy domain. If a valid policy is found, the control is calculated as described in Section IV-B. On our robot, the forward velocity and turning rate calculated by the control policy is converted to wheel speeds used by the low-level PID motor control loop.

B. Experimental Results

Seven representative tests on the robot are presented; a total of 19 tests were conducted. The actual data was logged every 0.1 seconds during the robot experiment, and is plotted with the obstacles shown in the figures. On the plots, the robot symbols are shown every five seconds of travel time. Although the policies were initially validated in simulation, the results presented here are for actual robot runs.

For this round of experiments, the robot lacked an external localization system, therefore the configuration estimates are based on pure dead-reckoning. As a result of dead-reckoning error, inherent in all wheeled-mobile robots, the robot would have crashed into some obstacles had they been physically present. Therefore, we ran the robot in an open space using the deployment of policies that considered obstacles as shown in the figures.

Figure 1-b shows the results of four of the robot runs from four different initial conditions and the same goal node. To

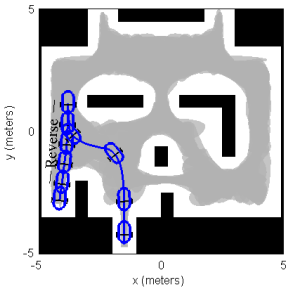
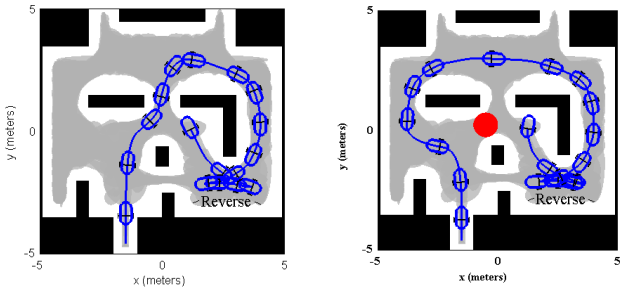


Fig. 8. Path #5 requires the robot to back out of the corridor, and then automatically switch to forward motion.



a) Path #6 b) Path #7

Fig. 9. Paths #6 and #7 show the execution of a K-turn to allow motion around a sharp corner; the paths diverge in response to two policies that are invalidated during run #7.

emphasize, the paths shown are from an actual robot run, and are the result of policy ordering that is based on assigned edge costs in the discrete graph; an explicit desired path is never calculated. The paths are labeled (#1 - 4) clockwise from the lower left.

The path labeled #5, shown in Figure 8, begins near the same position as path #1 (shown in Figure 1-b); however, the orientation is approximately 180 degrees different. As the deployment also includes policies with the reverse input set, the robot backs up and then moves forward to the goal. This automatic policy switching requires no operator intervention.

Paths #6 and #7, shown in Figure 9, have two features that demonstrate the flexibility of the approach. First, like path #5, they demonstrate automatic forward/reverse switching. In this case, the deployment does not include policies that are expressive enough to turn the lower right corner in one continuous motion. The robot initiates a K-turn, again with the policy switching automatically dictated by the policy ordering. While this points to the need for future research in designing more expressive policies, it also validates the basic approach as the combination of simple policies with discrete planning is still capable of generating expressive motions.

The second feature demonstrated by paths #6 and #7 is that of re-planning in the discrete space of available policies. The robot starts from slightly different initial conditions, but begins to converge to the same vector field flow near the K-turn. Just after the K-turn during run #7, the two policies crossing the circular obstacle shown in Figure 9-b are flagged as invalid.

This triggers a re-planning step using D* Lite that reorders the policies, thereby inducing the robot to take "the long way around" via path #7. This re-planning occurred in real time, while the hybrid control policy was executing on the robot.

VI. Conclusion

The results of early simulations and the actual robot experiments presented in this paper demonstrate the validity of the approach, and provide a powerful demonstration of the inherent flexibility of planning in the space of control policies. Policies of the type outlined in this paper enable a large class of discrete planners, including those that can handle temporal logic specifications, to work with real, non-ideal, convex-bodied systems operating in real environments. The sequential composition techniques advocated in this paper are generally applicable to a wide variety of control policies, beyond the flow-through policies described in this paper, provided the basic requirements that we outline are met.

The approach provides consistent results and maintains the provable guarantees of the local control policies, while enabling the flexibility to replan in real time. By not trying to cover "every nook and cranny" of the free configuration space, the number of policies is kept reasonable at the cost of completeness. Limiting the size of the discrete planning problem enables faster planning in the space of control policies. While the method is not complete, the ability to plan and re-plan efficiently coupled with the guaranteed closed-loop results validate the approach. Future work on automating the deployment of local policies will address the completeness issues associated with this approach by automatically deploying additional policies on-line.

As the ultimate success of the strategy depends on accurate estimates of configuration, future experiments will incorporate vision-based localization that allows the system to estimate its configuration based on the position of known landmarks. This will allow testing the control strategy with a fully integrated autonomous system operating amongst physical obstacles.

Our immediate research plans are to extend the results to more complicated configuration spaces and mappings $A(q)$, such as those for the 4-variable Ackermann steered car and a diff-drive towing a trailer. We are also striving for more expressive local policies, and improvements to the deployment scheme to allow faster, more automated deployment. We are also implementing temporal logic based planning [24]. Finally, we expect the benefits presented in this paper to become even more obvious as we expand to systems with second-order dynamics and bounds on available acceleration.

References

[1] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[2] J. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer Academic Publishers, 1991.

[3] J. P. Laumond, Ed., *Robot Motion Planning and Control*. Springer-Verlag, 1998.

[4] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *International Journal of Robotics Research*, vol. 18, no. 6, pp. 534–555, 1999.

[5] J. P. Laumond, S. Sekhavat, and F. Lamiraux, *Robot Motion Planning and Control*. Springer-Verlag, 1998, ch. Guidelines in Nonholonomic Motion Planning for Mobile Robots, pp. 1–54.

[6] S. Sekhavat and M. Chyba, "Nonholonomic deformation of a potential field for motion planning," in *Proceedings of IEEE International Conference on Robotics and Automation*, May 1999, pp. 817–822.

[7] J. Barraquand and J. Latombe, "Nonholonomic multibody robots: Controllability and motion planning in the presence of obstacles," *Algorithmica*, vol. 10, pp. 121–155, 1993.

[8] S. M. LaValle, "From dynamic programming to RRTs: Algorithmic design of feasible trajectories," in *Control Problems in Robotics*, A. Bicchi, H. I. Christensen, and D. Prattichizzo, Eds. Berlin: Springer-Verlag, 2002, pp. 19–37.

[9] B. Mirtich and J. Canny, "Using skeletons for nonholonomic path planning among obstacles," in *Proceedings of IEEE International Conference on Robotics and Automation*, May 1992, pp. 2533–2540.

[10] M. Vendittelli, J. Laumond, and C. Nissoux, "Obstacle distance for car-like robots," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 4, pp. 678–691, 1999.

[11] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[12] T. A. Henzinger, "The theory of hybrid automata," in *Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS)*. IEEE Computer Society Press, 1996, pp. 278–292.

[13] M. S. Branicky, "Studies in hybrid systems: Modeling, analysis, and control," Ph.D. dissertation, MIT, Dept. of Elec. Eng. And Computer Sci., June 1995.

[14] R. DeCarlo, M. Branicky, S. Pettersson, and B. Lennartson, "Perspectives and results on the stability and stabilizability of hybrid systems," *Proceedings of the IEEE, Special Issue on Hybrid Systems*, vol. 88, no. 7, pp. 1069–1082, July 2000.

[15] 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*, vol. 4, April 2000, pp. 4021 – 4026.

[16] L. Yang and S. M. Lavalle, "The sampling-based neighborhood graph: An approach to computing and executing feedback motion strategies," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 419–432, June 2004.

[17] G. Kantor and A. A. Rizzi, "Feedback control of underactuated systems via sequential composition: Visually guided control of a unicycle," in *Proceedings of 11th International Symposium of Robotics Research*, October 2003.

[18] S. Patel, S.-H. Jung, J. P. Ostrowski, R. Rao, and C. J. Taylor, "Sensor based door navigation for a nonholonomic vehicle," in *IEEE International Conference on Robotics and Automation*, Washington,DC, May 2002, pp. 3081–3086.

[19] C. Belta, V. Iser, and G. J. Pappas, "Discrete abstractions for robot planning and control in polygonal environments," University of Pennsylvania, Tech. Rep. MS-CIS-04-13, 2004.

[20] D. C. Conner, A. A. Rizzi, and H. Choset, "Composition of Local Potential Functions for Global Robot Control and Navigation," in *IEEE/RSJ Int'l. Conf. on Intelligent Robots and Systems*, Las Vegas, NV, October 2003, pp. 3546 – 3551.

[21] S. R. Lindemann and S. M. LaValle, "Smoothly blending vector fields for global robot navigation," in *IEEE Conference on Decision and Control*, Seville, Spain, 2005.

[22] D. C. Conner, A. A. Rizzi, and H. Choset, "Integrated planning and control for convex-bodied non-holonomic systems using local feedback control policies," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-06-34, August 2006.

[23] G. A. Kantor and A. A. Rizzi, "Sequential composition for control of underactuated systems," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. TR-03-23, December 2003.

[24] G. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Hybrid controllers for path planning: A temporal logic approach," in *IEEE Conference on Decision and Control*, Seville, Spain, 2005.

[25] M. Likhachev and S. Koenig, "Speeding up the parti-game alorithm," in *Advances in Neural Information Processing Systems 15*. MIT Press, 2003.