Proceedings of the 2002 IEEE
International Conference on Robotics & Automation
Washington, DC • May 2002

A Method for Modifying Closed-Loop Motion Plans to Satisfy Unpredictable Dynamic Constraints at Runtime

Joel M. Esposito * Vijay Kumar

Mechanical Engineering and Applied Mechanics
University of Pennsylvania, Philadelphia, PA 19104

Abstract

In this paper, the problem of motion planning in environments with both known static obstacles and unpredictable dynamic constraints is considered. A methodology is introduced in which the motion plan for the static environment is modified on-line to accommodate the unpredictable constraints in such a way that the completeness properties of the original motion plan are preserved. At the heart of the approach is the idea that Navigation functions are indeed Lyapunov functions; and that the traditional method of forcing the robot to track the negative gradient of field is not the only input which stabilizes the system. This extra freedom in selecting the input is used to accommodate the dynamic constraints. A computational method for selecting the appropriate inputs is given. The method is used to solve two sample problems. The constraints in these cases are used to model collisions with other robots and, in the second example, a team of robots traveling in formation. Finally, some preliminary work on extending the approach to nonholonomic systems is presented.

1 Introduction

Most solutions to the motion planning problem have the desirable attribute of *completeness* meaning that they are guaranteed to find a solution if one exists or report failure if no solution exists. The draw back of such approaches is that, when unexpected changes to the model of the environment occur, there is no satisfactory way of modifying the motion plan online which preserves the completeness properties of the original solution. In contrast, reactive planning ([1]) is a paradigm in which the actions of the robot are simply a function of its sensor inputs and are computed in real time in response to changes in the environment. They have the advantage of enabling robust operation in dynamic and changing environments; however suffer from a lack of completeness. It is difficult to establish performance guarantees except in a very limited set of special cases. In this paper we introduce a new planning methodology which attempts to bridge the gap between these two approaches. We assume that the robot is provided with a map of the static obstacles in the environment up-front and that the robot is capable of generating a corresponding static solution to the planning problem. A list of reactive requirements or constraints whose time dependence is not known in advance are also provided. We present a way of modifying the motion plan for the static environment online to locally accommodate the dynamic constraints whenever possible.

Since we place few restrictions on the dynamic nature of the constraints one cannot always guarantee that the problem can be globally solved. However, we provide a method that can determine if a local solution exists and either compute one, or report failure and alert the high level planner that a global replanning is needed as a last resort. The motivation for this is: (1) a global replanning is expensive and we would like to rely on reactive solutions whenever possible without sacrificing completeness; and (2) since we have no prior knowledge of the time-dependence of the constraints it is impossible to account for them up front.

Similar problems have been addressed in the literature. Game theoretic approaches treat the dynamic constraints as controlled by an adversarial agent and attempt to find the worst case inputs for the system, [2] and [3]. In [4] a method of altering a Navigation function to account for unmodeled obstacles (topological alterations) or poorly modeled obstacle geometries (geometric alterations) is proposed. A method termed *reflexive collision avoidance*, is developed [5] which is essentially an obstacle avoidance controller that accounts for the robot's dynamics. In [6] homotopic deformations to preplanned trajectories are computed which enable the robot to circumvent unmodeled obstacles. A behavior-based or reactive control paradigm which switches between several simple controllers based

*We gratefully acknowledge support from DARPA grant ITO/MARS 130-1303-4-534328-xxxx-2000-0000, and a DoE GAANN grant.

on changes in the environment is advocated in [1]. Our approach differs from these in several ways. Most importantly in some sense, it preserves the completeness properties of the "static solution" by not introducing spurious equilibria and reporting when no local solution exists. It can also account for more general types of dynamic constraints, other than simple obstacle avoidance. It is also relatively cheap from a computationally point of view.

2 Problem statement

Static Problem (Basic Motion Planning) Given a robot R, with a fully actuated, holonomic, kinematic model $\dot{q} = u$, a goal configuration q_g, and a group of sets O_i where $i = 1, \ldots, N$ describing the known static obstacles in the environment; assign an input $\bar{u}(q,t)$ which steers the robot from any initial position q_0 to the goal q_g (provided q_0 and q_g are in the same connected subset of the workspace) – while not hitting any static obstacles O_i. We refer to the map $\bar{u}(q,t)$ as the *static solution.*

Dynamic Problem In addition to all the data given in the *static problem,* a partially ordered list of inequalities $g_j(q,t) \leq 0$ (where $j = 1, \ldots, M$) which represent the reactive requirements is given. Note that the time dependent portion of the constraint dynamics may not be known in advance. The problem is then to satisfy all the requirements of the *static* solution *while* not violating any of the inequality constraints q_g. The algorithm for generating the *dynamic solution* $u(q, g_1, \ldots, g_M, t)$ should be such that it is guaranteed to produce a solution *locally* consistent with the constraints if one exists and report failure otherwise.

3 Approach

The key observation which we exploit to solve the above problem is that Navigation functions, $V(q)$, which solve the static problem, are actually Lyapunov functions. The traditional control law of $u = -\nabla V$ is not the only input capable of rendering $\dot{V} < 0$; there is in fact a uncountable set of such inputs. We exploit this freedom in choosing u to satisfy the additional dynamic constraints whenever possible. In this section we prove that this set of inputs exist, construct it, and give a computational method for assigning the inputs to the system.

Navigation functions We assume that a Navigation function,V has been constructed which solves the static problem (i.e. a function that steers the robot to q_g while avoiding $O_1, \ldots, O_M$). We choose to base our methodology on Navigation functions[7] because they represent an

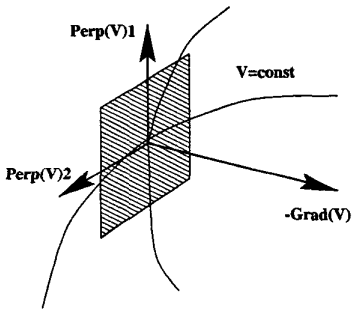

Figure 1: An illustration of the vectors $-\nabla V$, and $[-\nabla V]^{\perp}$ in R^3. Any velocity vector in the same half plane as $-\nabla V$ also decreases $V(q)$.

algorithmically complete *closed loop* solution to the planning problem. Typically, the robot's input is $u = -\nabla V$ which causes the robot to reach the goal and halt while avoiding obstacles (and hence represents what we call a solution to the static planning problem). Navigation Functions can be though of as Lyapunov functions for the system $\dot{q} = u(q)$, where $u(q) = -\nabla V(q)$, because $V(q)$ is positive definite by construction and, by definition of the control policy, the value of V is always decreasing along system trajectories

$$\dot{V} = \nabla V \cdot u(q) = -\nabla V \cdot \nabla V \leq 0. \qquad (1)$$

Sets of stabilizing inputs It is apparent however that this control policy is not unique– any control policy which renders $\dot{V} = -\nabla V \cdot \dot{q} \leq 0$ also solves the planning problem. This fact is observed in [4] and in [8]; the set of all input vectors which decrease some cost-to-go function is termed the "cone of progress". However in both of these contexts the fact is used passively to address sensor uncertainty. Here however we wish to actually construct a parameterized family of control laws which solve the static planning problem.

Proposition 1 *If $q \in R^n$, define mutually perpendicular vector fields $[\nabla V(q)]_i^{\perp}$, where $i = 1, \ldots, n-1$, which are also everywhere perpendicular to $\nabla V(q)$. Further assume each of these vectors has been normalized and is of unit length. See Fig. 1. Then the control law*

$$u_\alpha(q) = -(1 - \sum_{i=1}^{n-1} \alpha_i^2)^{1/2} \nabla V(q) + \sum_{i=1}^{n-1} \alpha_i [\nabla V(q)]_i^{\perp} \quad (2)$$

also solves the static planning problem, provided $\sum_{i=1}^{n-1} \alpha_i^2 < 1$.

The set of vector fields u_α represents all the vectors which lie in the same half-tangent space as $-\nabla V$. To prove the

proposition we show the robot (1) reaches the goal and (2) does not hit any obstacles. First we prove that $q \to q_g$:

Proof 1 *Observe that $V(q)$ serves as a common Lyapunov function for the equation $\dot{q} = u_\alpha$ regardless of the values of α_i since*

$$\begin{aligned}\dot{V} &= \nabla V \cdot ((\sum_{i=1}^{n-1} \alpha_i^2 - 1)^{1/2} \nabla V(q) + \sum_{i=1}^{n-1} \alpha_i [\nabla V(q)]_i^\perp) \\ &= -(1 - \sum_{i=1}^{n-1} \alpha_i^2)^{1/2} \nabla V(q) \cdot \nabla V(q) \leq 0\end{aligned}$$

provided $\sum_{i=1}^{n-1} \alpha_i^2 < 1$. Note that the new control law is free of local minima since the $\dot{V} = 0$ iff $q = q_g$ by definition of the navigation function. □

To show the second requirement:

Proof 2 *If the obstacles are defined by a closed surface C let $\hat{n}(C)$ be the unit normal pointing toward the interior of the free space. Then proving that the robot does not hit the obstacles is equivalent to proving $\dot{q} \cdot \hat{n} \geq 0$ using $\dot{q} = u_\alpha$*

$$(-(1 - \sum_{i=1}^{n-1} \alpha_i^2)^{1/2} \nabla V(q) + \sum_{i=1}^{n-1} \alpha_i [\nabla V(q)]_i^\perp) \cdot \hat{n} \quad (3)$$

Recall that navigation functions are uniformly maximal on the boundary of the free space, so $-\nabla V(q)$ is parallel *to $\hat{n}(C)$ for all $q \in C$, so $-\nabla V(q) \cdot \hat{n}(q) > 0$ and $\nabla V(q)^\perp \cdot \hat{n}(q) = 0$. Therefore eq.(3) becomes*

$$(1 - \sum_{i=1}^{n-1} \alpha_i^2)^{1/2} (-\nabla V \cdot \hat{n}) \geq 0 \quad \square$$

Thus all controllers in the set u_α solve the static planning problem. Finally we add that since the set of stabilizing inputs u_α results in a set of closed loop systems which share $V(q)$ as a common Lyapunov function, it can be shown (see [9]) that a system whose right hand side switches between these inputs is also stabilizing, regardless of the nature of the switching sequence. This implies that we are free to choose the values of α online, in a possibly discontinuous or time varying fashion, without affecting the overall stability of the system or the completeness of the solution.

Constraints Since the constraints are unpredictable in nature we do not always take them into account. As an objective measure of when to react to changes in the environment we introduce a quantity Δt_j which is an estimate of the time to constraint activation ($g_j(q(t), t) = 0$)

$$\Delta t_j = \frac{-g_j(q(t), t)}{\dot{g}_j(q(t), t)} \quad (5)$$

This quantity also encodes information about the kinematics of the robot. Small positive values of Δt_j imply that a constraint activation is impending; while negative values (moving away from the level surface $g_j = 0$) or large positive values are not a cause for concern. Thus if $0 \leq \Delta t_j \leq \delta_j$, g_j is added to the list of active constraints, where δ_j is some predetermine constant termed the *look ahead time.*

Once a constraint becomes active, our goal is to select an input which makes $\dot{g}_j \leq 0$. The time derivative of g_j is conveniently expressed as the sum of two quantities:

$$\dot{g}_j(q(t), t) = \frac{\partial g_j}{\partial q} \cdot \dot{q} + \frac{\partial g_j}{\partial t}. \quad (6)$$

the first term represents the robot's own influence on g_j and is assumed to be known; the second represents the dynamic nature of g_j and must be either sensed online or some assumptions must be placed on its value. We assume the robot has an expression for g_j and is equipped with sensor enabling it to measure its value online. In this work we only consider what are referred to in the optimal control literature as *first order constraints*, that is constraints for which $\frac{\partial g_j}{\partial q} \neq 0 \; \forall q$; although the extension for higher order constraints is straightforward.

Computational issues The objective then is to introduce a computational method for selecting an input, from the set of all inputs u_α (which, by construction, solve the static problem), that forces the derivative of any active constraints to be strictly non-positive. A constraint g_j is considered active if $0 \leq \Delta t_i \leq \delta_i$; at any given time $P \leq M$ constraints are active. Let $G = [g_1 \; \ldots \; g_P]^T \in R^P$ be the constraint vector and $G_q = \frac{\partial G}{\partial q} \in R^{P \times N}$ and $G_t = \frac{\partial G}{\partial t} \in R^P$. In the absence of additional constraints, we assume the nominal input is $u_\alpha = -\nabla V$ (*i.e.* $\alpha_1 = 0, \ldots \alpha_{n-1} = 0$). The problem can be phrased as

$$\min_{\alpha_i \in (-1,1)} \frac{1}{2} \sum_{i=1}^{n-1} \alpha_i^2 \quad (7)$$

such that $G_q u_\alpha \leq -G_t$ where the inequality is evaluated componentwise; u_α is defined in eq.(2) and $V(q)$ is computed using one of the algorithms mentioned in Sect. 1.

This problem is computationally identical to the "direction finding" sub-problem which is solved as part of the nonlinear programming method called the Method of Feasible Directions (see [10] for example). Well studied and efficient techniques are available for solving it. Most of the approaches involve using a series of projections of $-\nabla V$ onto the constraint directions to determine if a solution exists and to compute the one "closest" to the optimum.

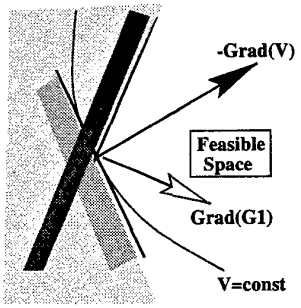

Figure 2: The addition of a constraint, g_1, with no time dependence further constrains the set of directions to the union of the half spaces containing $-\nabla V$ and ∇g_1

Geometric insight behind the problem can be gained from recognizing that the j^{th} inequality defines a *cone*, c_j (or the complement of a cone) with its apex at the origin in the tangent space of the body fixed frame; while the set of vectors $U = \{u_\alpha : \sum_{i=1}^{n-1} \alpha_i^2 \leq 1\}$ defines a half space. Figures 2 illustrates this in R^2.

It should be said that if $U \bigcap c_1 \bigcap \cdots \bigcap c_j = \emptyset$ there is no input that can simultaneously solve both the *static* planning problem and satisfy the reactive objectives. The algorithm used to calculate the inputs is capable of recognizing this and reports that a high level replanning is required; or that some reactive constraints must be discarded according to some predetermined priority rankings until a feasible solution exists. If the cone is not empty, an infinite number of solutions exist and the optimization problem can be solved at each step. Since this need only be solved at points along the trajectory its cheaper than a global replan however its may not be globally optimal.

4 Applications

Our framework is general enough that it can be used to solve a fairly diverse group of applications, the most obvious of which is dynamic obstacle avoidance. However more general types of constraints can be specified as well.

Obstacle avoidance We consider a situation in which the robot has a perfect map of the static obstacles in the environment; however, the presence moving obstacles (humans or other robots) complicates the problem. We assume the robot can measure the position and velocity of these moving obstacles but has no *a priori* knowledge of their trajectories. If $\hat{q}_1(t), \ldots, \hat{q}_M(t)$ are the position vectors of the M dynamic obstacles, and r_j are their radii. The dynamic constraints for the robot are

$$g_j = -(\|q - \hat{q}_j(t)\|^2 - (r + r_j)) \leq 0. \qquad (8)$$

Fig. 3 illustrates such a scenario. Important things to note about this example are: (1) R has no prior knowledge of the trajectory of the convoy; (2) the avoidance is performed in an online, purely reactive fashion without having to recompute the static plan; and (3) at all times during the execution $\dot{V} < 0$, so the completeness of the navigation function is preserved (no local minima or limit cycles are introduced).

Formation control Consider a situation in which a group of robots must travel from the respective starting configurations to their goal configurations; however they are to do so in formation whenever possible. If at anytime it is not feasible to achieve both of these objectives than it should report failure, break away from the formation and proceed to its goal. By a formation we mean that the robots must try to achieve and maintain some predetermined relative separation and bearing from each other. Such behavior is desirable in many applications, for example in the case of unmanned air vehicles, formation flight results in greater fuel economy. In other cooperative tasks close proximity of teammates is crucial.

In such situations we can assign one robot the role of leader and assume the follower robots can measure the position and velocity of the leader but have no *a priori* knowledge of its motion plan. Consider robot-i and let $q_i(t)$ be its position vector. This dynamic constraint is expressed as

$$g = -\|q_i - q_i^f(t)\|^2 \leq 0 \qquad (9)$$

where q_i^f is the desired position of q_i in the formation. Fig. 4, shows a simulation of such an example.

5 Nonholonomic systems

In practice most mobile robots are nonholonomic; therefore extending the methodology outlined in the first part of this paper to systems with velocity constraints is important from both a theoretical and practical point of view.

Dipolar fields A dipolar field is a type of scalar field with the special property that systems which track its negated gradient have integral curves which are feasible trajectories for nonholonomic mobile robots. This idea is introduced in [11]. The basic dipolar scalar field is $V(q) = |q_1|/(q_1^2 + q_2^2)$. As is the case for Navigation functions, various coordinate transformations are used to create a field which steers the system around stationary obstacles. Using this methodology, potential fields can be built for nonholonomic systems in static environments which share all of the completeness properties of Navigation Functions. This idea is developed further in [12] where an associated

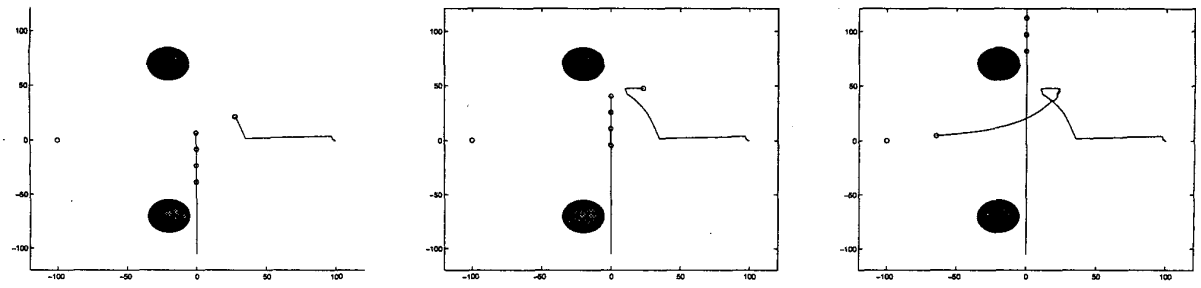

Figure 3: (L) The Robot (on the right) proceeds to the goal (on the left). (C) The robot tries to steer to the right around the moving convoy but is blocked by a static obstacle. Finally (R) the robot loops around to pass the convoy safely on the left.

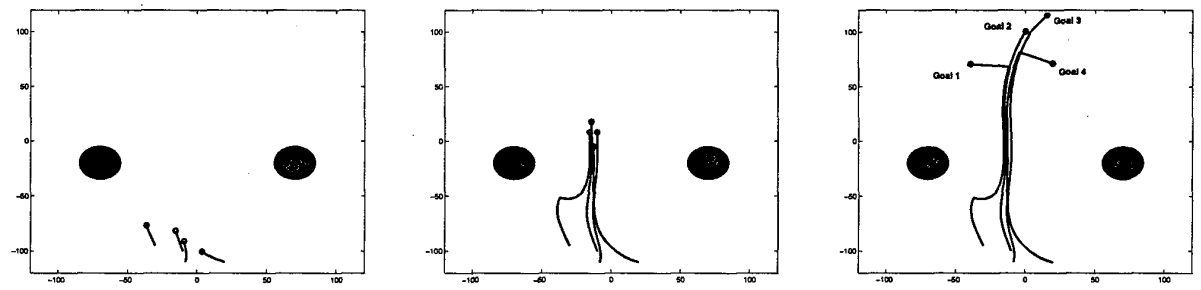

Figure 4: (L) The robots assume formation. (C) Traveling in formation. (R) Breaking off to pursue individual objectives.

controller is introduced that stabilizes θ so that the nonholonomic system is able to track the gradient direction. This controller is developed through a backstepping approach, by looking at how a holonomic system would move under the influence of the dipolar field and then deriving a controller for the nonholonomic system which would mimic the trajectory of the holonomic system.

Difficulties Recall in the case of holonomic systems, the idea behind the method introduced in this paper was to select inputs u such that

$$\dot{V} = \frac{\partial V}{\partial q} u < 0 \quad and \quad \dot{G} = \frac{\partial G}{\partial q} u + \frac{\partial G}{\partial t} < 0,$$

however, in the case of underactuated nonholonomic systems, these inequalities would become

$$\dot{V} = \frac{\partial V}{\partial q} F u < 0 \quad and \quad \dot{G} = \frac{\partial G}{\partial q} F u + \frac{\partial G}{\partial t} < 0,$$

where $F \in \Re^{N \times K}$. When $dim(u) < dim(q)$ (*i.e.* $K < N$) its becomes more difficult to ensure a that a feasible direction is contained within $span(F)$.

Results Because the nonholonomic constraints on mobile robots are prohibitively restrictive we do not directly apply our methodology to nonholonmic mobile robots. Instead we proceed as [12], by assigning $V(q)$ to be a dipolar field. We then proceed to computationally solve the *dynamic* planning problem outlined in Sect. 3 as if the system was holonomic. We then backstep the ideal holonomic input to obtain an input to the nonholonomic system. In doing so the completeness properties of the *static* solution are preserved; however it is difficult in practice to ensure the dynamic constraints are satisfied at all times. In theory it is always a possible to mimic the ideal trajectory of the holonomic system by having high enough gains and a large enough lookahead time but it is difficult to actually select these values since they are problem dependent.

This idea was first simulated, the results of which are shown in Fig. 5. In the scenario depicted there a unicycle type robot, with the kinematic model

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega, \qquad (10)$$

has an initial map of the environment which does not contain any obstacles. It uses the dipolar potential field and controller from [12] as a static solution. The left frame of the figure shows that the nominal trajectory would have steered the robot on a collision course with an obstacle that is unmodeled initially but detected at run time. The right panel shows the trajectory of the robot using the methodology outlined in this paper, once the robot detects the ob-

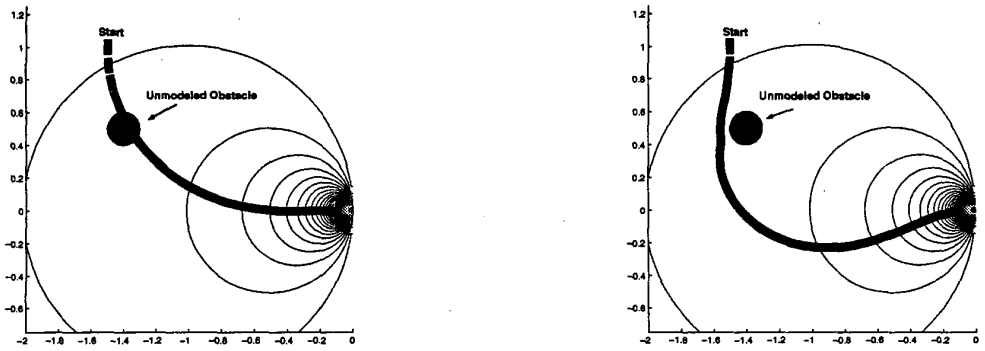

Figure 5: (L) The robots nominal trajectory would cause it to bump into the unmodeled obstacle (dark circle). (R) The dynamic planning methodology enables the robot to steer around the unexpected obstacle on-the-fly.

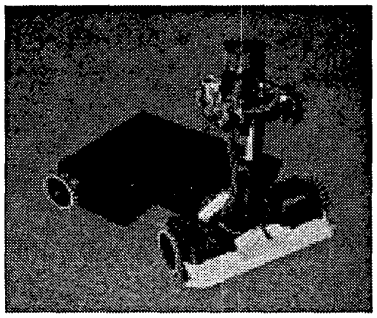

Figure 6: The experimental platform is equipped with IR range sensors and an onboard computer.

stacle it is able to steer around. The concentric circles in the figure are level sets of the potential field. One can see that at all points along the trajectory, the potential function is decreasing, thereby ensuring stability.

The differential drive robot (see Fig. 6) on which the controller is being implemented is a Cye Research robot (trademarked by Probotics Inc.), which has been fitted with infrared range sensors and a laptop computer.

6 Conclusion

In this paper, the problem of motion planning in environments with both known static obstacles and unpredictable dynamic constraints is considered. A methodology is introduced in which the motion plan for the static environment is modified on-line to accommodate the unpredictable constraints in such a way that the completeness properties of the original motion plan are preserved. A computational algorithm was introduced to compute inputs for holonomic systems. A primary concern is identifying the class of environments and constraints for which the algorithm is guaranteed to succeed. Other future work will be centered on extending the preliminary results for nonholonomic systems, implementing the technique on an experimental platform and focus on modeling other types of applications such as cooperative manipulation.

References

[1] R. Arkin, *Behavior Based Robotics*. Cambridge, MA: The MIT Press, 1998.

[2] J. M. Esposito and V. Kumar, "Closed loop motion planning for mobile robots," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (San Francisco, CA), pp. 1020–1025, May 2000.

[3] S. Lavalle and S. Hutchinson, "Path selection and coordination of multiple mobile robots via nash equilibria," *Proceedings of 1994 International Conference on Robotics and Automation*, pp. 1847–1852, 1994.

[4] D. E. Koditschek, "The geometry of a robot programming language," *Workshop on the Algorithmic Foundations of Robotics*, vol. 3, pp. 263–268, 1994.

[5] T. S. Wikman, M. S. Branicky, and W. S. Newman, "Reflexive collision avoidance: A generalized approach," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 3:31–36, May 1993.

[6] O. Brock and O. Khatib, "Real time replanning in high-dimensional configuration spaces using sets of homotopic paths," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2000.

[7] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[8] M. Erdmann, "Understanding action and sensing by designing action-based sensors," *International Journal of Robotics Research*, vol. 14, no. 5, pp. 483–509, 1995.

[9] K. Narendra and J.Balakrishnan, "A common lyapunov function for stable LTI systems with commuting A matricies," *IEEE Transactions on Automatic Control*, vol. 39, pp. 2469–2471, 1994.

[10] G. Vanderplaats, *Numerical Optimization Techniques for Engineering Design*. McGraw–Hill, 1984.

[11] H. G. Tanner and K. J. Kyriakopoulos, "Nonholonomic motion planning for mobile manipulators," *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pp. 1233–1238, 2000.

[12] H. G. Tanner, S. Loizou, and K. J. Kyriakopoulos, "Nonholonomic motion planning for mobile manipulators," *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.