# A formalism for parallel composition of reactive and deliberative control objectives for mobile robots

Joel M. Esposito Vijay Kumar

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

## Abstract

*In this paper we introduce a methodology for closed loop motion planning which is able to reactively accommodate dynamic constraints, such as avoiding other robots, at run time without nesecatate a global replanning. The methodology is complete in the sense that it is guaranteed to produce a successful motion plan that is (locally) consistent with the constraints if one exists; and reports failure when a feasible solution does not exist, in which case a global replanning is necessary. This is accomplished by noting that given a particular robot and environment, and a Navigation Function, the most common control policy of following the negated gradient is not the unique solution to the planning problem. We construct a parameterized set of acceptable control laws, and use the extra degrees of freedom in the parameter choice to satisfy the constraints by solving an optimization problem. Two example applications solved here are avoiding moving obstacles and traveling in formation; both while trying to reach a goal position.*

## 1 Introduction

The most basic version of the motion planning problem can be stated as:

**Problem 1** *Given a geometric descriptions of the work space, $W$, and a finite number of obstacles $O_1, O_2, \ldots$; and a goal position, $q_g$, for the point robot, $R$, with configuration $q$ and the dynamics*

$$\dot{q} = u(q,t) \tag{1}$$

*design a function $u(q,t)$ that drives $q \rightarrow q_g$ and halts there without colliding with the obstacles; or report failure if no path exists.*

In this paper we introduce a tractable method to solve the a more sophisticated version of this problem, which is to generate a input function $u(q,t)$ that fulfills all the above criteria *while* satisfying dynamic (in)equality constraints of the form $g(q,t) \leq 0$. One important example of such a constraint would be where the robot has an accurate map of the static obstacles in the environment but an unmodeled moving obstacle, human, or other robot enters the workspace and th robot must plan a path around it.

Unlike optimal control approaches, we do not assume that the dynamic behavior of these constraints is known *a priori* (e.g. the human's motion is unpredictable). In fact the goal of our approach is to formulate a global "static solution" to the unconstrained version of the planning problem upon initialization; then, at run-time, locally modify the initial plan as needed to accommodate the dynamic constraints, without requiring a global replanning if possible, in such a way as to preserve the completeness properties of the global "static solution". The motivation for this is: (1) we assume a global replanning is expensive and would like to rely on reactive solutions whenever possible; and (2) since we have no prior knowledge of the time-dependence of the constraints it would be impossible to account for them up front.

Previous approaches to similar classes of problems include:

- game theoretic approaches treat the dynamic constraints as controlled by an adverserial agent and attempt to find the worst case inputs for the system, [1] and [2]. This approach yields a global solution to the problem we are considering but is intractable for all but the simplest applications;
- In [3] a method of altering a Navigation function to account for unmodeled obstacles (topological alterations) or poorly modeled obstacle geometries (geometric alterations) is proposed but it is not applicable

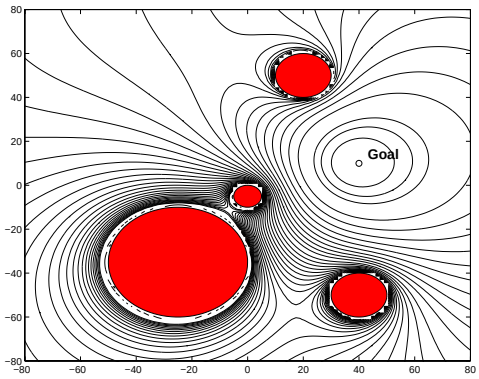


Figure 1: Level sets of a sample Navigation function. The solid circles are obstacles. Note the unique minimum at the goal.

to time dependent changes;

- A method termed *reflexive collision avoidance*, is developed [4] which is essentially an obstacle avoidance controller that accounts for the robot's dynamics, however once the avoidance behavior is activated a global replanning is required to reach the goal.

- In [5] homotopic deformations to preplanned trajectories are computed which enable the robot to circumvent unmodeled obstacles; however this approach is not readily extensible to other types of online constraints

- A behavior-based or reactive control paradigm which switches between several simple controllers based on changes in the environment is advocated in [6] but this approach does not guarantee completeness

Our approach differs from these in several ways. Most importantly, 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 reactive requirements, other than simple obstacle avoidance, such as formation control or cooperative manipulation.

As a static solution to the planning problem we use Navigation Functions, a special class of potential fields. Navigation Functions, [7], are scalar fields, $V(q)$, defined over the free configuration-space which possess a unique minimum value of $V = 0$ at the goal configuration of the robot, $q_g$; and attain a maximum value of $V = 1$ at the boundaries of the free space. This concept is illustrated in Figure 1 which shows the level surfaces of $V(q)$ for an example environment. By employing a feedback control law of the form

$$u(q) = -\nabla V(q) \tag{2}$$

which tracks the direction of steepest descent in $V(q)$ the planning problem is solved. The Navigation function, $V$, and the associated control law, eq.(2), are a desirable solution to the planning problem for a variety of reasons including: they can account for the Lagrangian dynamics of the robot, the prescribed inputs are always bounded and $C^2$ smooth, they are free of local minima, and, perhaps most importantly, they compute a closed loop solution to the planning problem. In [7] recipes for constructing Navigation Functions on increasingly complex spaces are presented, ultimately however there is a limit to how general these environments are. However a variety of algorithms for constructing "Numerical Navigation Functions" are known (see [8]). These functions possess most of the desirable properties of Navigation Functions and permit the construction of Navigation-like functions on arbitrary spaces (however since they are inherently discrete they cannot be $C^2$ smooth). From an implementation standpoint they can be used on the raw output of a occupancy map created by a robot mapping its environment.

In Section 2.1 a parameterized set of control laws is constructed which accomplish the same result as eq.(2) – every member of this set is guaranteed to bring the robot to the goal while avoiding obstacles. This implies there is some freedom in selecting a control law from this set. In Section 2.2, we formalize the "reactive requirements as inequality constraints. In Section 2.3, we use the freedom in selecting the control law to satisfy the constraints by formulating an optimization problem. Two applications of this framework, obstacle avoidance and traveling in formation, are solved as example problems in Section 3. Finally in Section 4, we comment on further applications of the framework.

# 2 Approach

## 2.1 Families of navigation functions

As mentioned in Section 1, Navigation functions define a scalar field $V(q)$ with a unique minimum over the free space, such that if the robot follows the negated gradient of this field, eq.(2), it is guaranteed to reach the goal and halt without hitting any obstacles. 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

$$\dot{V} = \nabla V \cdot u(q) = -\nabla V \cdot \nabla V \leq 0. \tag{3}$$

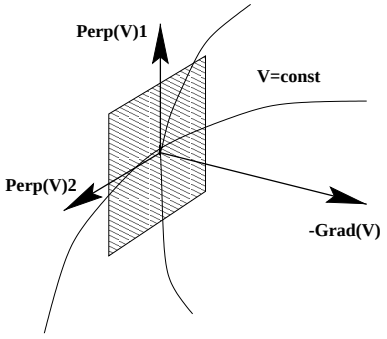


Figure 2: An illustration of the vectors $-\nabla V$, and $[-\nabla V]^{\perp}$ in $R^3$. Any velocity vector in the same half plane as $-\nabla V$ also decreases $V(q)$.

Using Lyapunov arguments, completeness of the navigation functions as motion planning tools can be shown, since $\nabla V(q) = 0$ only when $q = q_g$.

It is apparent however that this control policy is not unique– any control policy which renders $\dot{V} \leq 0$ also solves the planning problem. This fact is observed in [3] and in [9]; 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 Figure 2. Then the control law*

$$u_\alpha(q) = -(1 - \sum_{i=1}^{n-1} \alpha_i^2)^{1/2} \nabla V(q) + \sum_{i=1}^{n-1} \alpha_i [\nabla V(q)]_i^{\perp} \tag{4}$$

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

**Proof 1** *As proof that these control laws drive $q \to q_g$, and do not introduce any local minima observe that $V(q)$ serves as a common Lyapunov function for the equation $\dot{q} = u_\alpha$ regardless of the values of $\alpha_i$ since*

$$\begin{aligned} \dot{V} &= \nabla V \cdot (-(1 - \sum_{i=1}^{n-1} \alpha_i^2)^{1/2} \nabla V(q) + \sum_{i=1}^{n-1} \alpha_i [\nabla V(q)]_i^{\perp}) \\ &= -(1 - \sum_{i=1}^{n-1} \alpha_i^2)^{1/2} \nabla V(q) \cdot \nabla V(q) \leq 0 \end{aligned} \tag{5}$$

*provided $\sum_{i=1}^{n-1} \alpha_i^2 < 1$. Hence the robot is guaranteed to reach the goal and halt there. Note that the new control law is free of local minima since the equality in eq.(5) is only achieved at $q_g$.*

*The second requirement for path planning is that the robot never collide with the obstacles. If the obstacles are defined by a simple closed curve $C$ and $\hat{n}(C)$ is the unit normal pointing toward the interior of the free space; then if $-\nabla V(q)$ is parallel to $\hat{n}(C)$ for all $q \in C$, $-\nabla V(q) \cdot \hat{n}(q) = 1$ then it is trivial to show that the vector field $u_\alpha(q)$ also satisfies this requirement on $C$*

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

*Since $(1 - \sum_{i=1}^{n-1} \alpha_i^2)^{1/2} \geq 0$ the new vector also points away from the obstacle.*

It is well know (see for example [10]) that if a set of stable differential equations share a common Lyapunov function, as the set of equations $\dot{q} = u_\alpha$, $\forall \alpha_i$ such that $\sum_{i=1}^{n-1} \alpha_i^2 \leq 1$, share $V(q)$ as a Lyapunov function, then a system whose dynamics switches between the right hand sides of those differential equations shares the same stability properties as the original set of systems. This is true regardless of the nature (sequence, frequency, etc.) of the switching. This ensures that *any program for $\alpha$ results in a system which is globally asymptotically stable to the origin*, including one where alpha time varying or possibly even discontinuously.

## 2.2 Constraints

The robot has a series of reactive requirements which cannot be accommodated up front because they are based on events we have no *a priori* knowledge of. These reactive or low-level requirements are expressed as a series of $M$ inequality constraints of the form $g_j(q,t) \leq 0$. Certainly the constraint is active when $g(q) = 0$; but to be more tolerent to sensor noise, we may want to consider the constraint in planning before $g(q) = 0$. As a rule for deciding if a constraint should be considered active we introduce a quantity $\Delta t_j$ which is an estimate of the time to constraint activation ( $g_j(q(t),t) = 0$ )

$$\Delta t_j = \frac{-g_j(q(t),t)}{\dot{g}_j(q(t),t)} = \frac{-g_j(q(t),t)}{L_{u_\alpha} g + \frac{\partial g}{\partial t}} \tag{6}$$

where $L_{u_\alpha} g$ is the Lie derivative of the constraint along trajectories of the robot. Small positive values of $\Delta t_j$ 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 $\delta_j$ is some predetermine constant termed the *look ahead time*.

Note that the time derivative of $g_j$ is conveniently expressed as the sum of two quantities:

$$\dot{g}_j(q(t), t) = \frac{\partial g_j}{\partial q} \cdot \dot{q} + \frac{\partial g_j}{\partial t}. \tag{7}$$

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.

These inequalities can model a wide variety of constraints such as the proximity to a moving object or the deviation from a desired position in a formation. It may be useful, when there are many constraints, to place a partial ordering on the constraints which indicate their relative priorities. That is, in the event it is not possible to comply with all of the constraints, one could designate which take precedent.

## 2.3 Computational issues

In the absence of additional constraints, the nominal input is $u_\alpha = -\nabla V$ (*i.e.* $\alpha_1 = 0, \ldots \alpha_{n-1} = 0$). 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. Obviously if $g_j \leq 0$ is desired, when constraint $g_j$ is active it is imperative that $\dot{g}_j \leq 0$. Thus at each time step the problem can be phrased as an optimization problem with a set of inequality constraints. 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 $G_t = \frac{\partial G}{\partial t} \in R^P$

**Problem 2**

$$\min_{\alpha_i \in (-1,1)} \frac{1}{2} \sum_{i=1}^{n-1} \alpha_i^2 \tag{8}$$

*such that*

$$G_q u_\alpha \leq -G_t \tag{9}$$

*where the inequality is evaluated componentwise;* $u_\alpha$ *is defined in eq.(4) and* $V(q)$ *is computed using one of the algorithms mentioned in Sect. 1*

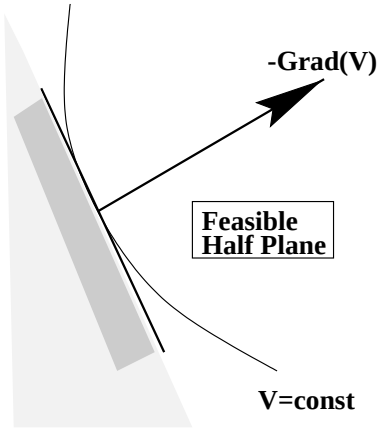


Figure 3: In the absence of constraints the set of feasible directions is the half plane containing $-\nabla V$

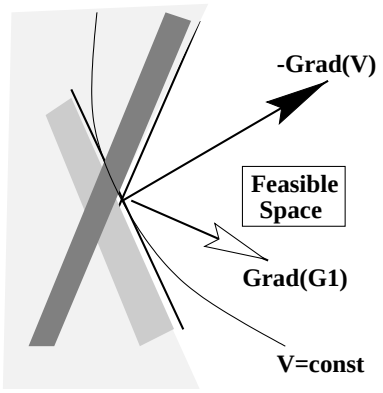


Figure 4: The addition of a constraint, $g_1$, with no time dependence further constrains the set of directions to the union of the half spaces containing $-\nabla V$ and $\nabla g_1$

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 3–5 illustrate this in $R^2$.

If $U \bigcup c_1 \bigcup \cdots \bigcup c_j = \emptyset$ there is no input that can simultaneously solve both the planning problem and the reactive objectives objectives. This implies 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 not a global optimal.

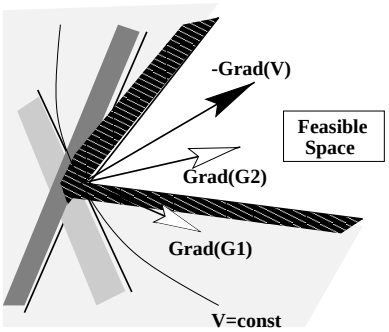


Figure 5: A time dependent constraint $g_2$ would reduce the feasible directions to the union of the feasible space in Figure 4 with the interior of a cone.

Since we are proposing to solve the problem online, at each time-step, a discussion of the required computational effort is warranted. First note that the objective function is convex since it is quadratic. In the case of $G_t \leq 0$ the design space is also convex, implying that the problem admits a computationally efficient solution.

Although convexity is an important condition for computational feasibility, the dimension of the problem and the constraints also plays a role in determining the complexity of the problem. This a minimization problem in $n-1$ variables and $P+1$ constraints; this can be solved readily using a sequential quadratic programming algorithm. However it is well known in the optimization literature that if $P+1 \leq n-1$ it is advantageous from a computational point of view to convert to the *dual problem*.

The dual problem is an *unconstrained* optimization problem in $P+1$ variables. This reduction may be important for higher dimensional problems. It should also be noted that, from our point of view, a sub-optimal solution is perfectly acceptable provided the constraints are not violated. This fact further enables fast computation.

# 3 Sample Applications

Our framework is general enough that it can be used to solve a fairly diverse group of applications such as:

- static/dynamic obstacle avoidance
- formation control
- cooperative manipulation using force closure
- synthesis of switching controllers

We give detailed examples of the first two applications later in this section. The last two examples are mentioned briefly in the next section as future work.

## 3.1 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

$$g_j = -(\|q - \hat{q}_j(t)\|^2 - (r + r_j)) \leq 0 \qquad (10)$$

and

$$\dot{g}_j = -(2\|q - q_j(t)\| \cdot u_\alpha - 2\|q - q_j(t)\| \cdot \dot{q}_j) \qquad (11)$$

and if $q \in R^2$

$$\dot{q} = u_\alpha(q) = (\alpha^2 - 1)\nabla V(x) + \alpha[\nabla V(x)]^\perp \qquad (12)$$

where $V$ is some suitable navigation function which accounts for the static obstacles in the scene.

Such an example is pictured in Figure 6, which appears on the last page of this paper following the References. The six frames are snapshots in time. In the upper left frame, the robot, $R$, at the far right is attempting to reach the goal (the small circle at the far left). However a convoy of four moving robots (lower center of the frame) is proceeding across its path. There are also stationary obstacles (large solid discs) in the scene. $R$, begins by creating a static plan which accounts for the stationary obstacles. $R$ has no *a priori* knowledge of the future positions of the convoy. At this time the convoy is far away enough that the constraints are not considered active $R$'s objective is to reach its goal, while avoiding the moving convoy, without having to recompute a static plan, if possible. If at anytime it is not feasible to achieve both of these objectives than the algorithm is to report failure; at which time a global replanning could be initiated.

In the second and third frame (upper center/right), $R$ now considers the constraints active and attempts to circumvent the convoy by veering to the right. By frame 4 (lower left) however the presence of the static obstacle has prevented $R$ from passing on the right and it doubles back around to the left (frame 5 lower center) to pass safely behind the convoy and proceed to the goal (lower right).

Important things to note about this scenario 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 the completeness of the navigation function is preserved, that is to say the robot is guarantee to reach the goal eventually without getting stuck in

local minima. Hence this represents a provably correct way of composing "goal seeking" and "obstacle avoidance" behaviors in parallel. In a similar fashion the same framework can be used to plan in situations where the convoy are simply static obstacles which were not accounted for up front, by simply setting $\frac{\partial q}{\partial t} = 0$. We could also use this approach in situations where the other robots are actively chasing $R$.

### 3.2 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. 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. Let $q_l(t)$ be the position vector of the leader robot. If robot-$i$ is to follow the leader with a separation distance of $d_i$ and a bearing of $\phi_i$ then the desired position of robot-$i$, such that the formation is kept is $q_i^f(t) = q_l(t) + [d_i\cos(\phi_i), d_i\sin(\phi_i)]^T$. This dynamic constraint is expressed as

$$g = -\|q - q_i^f(t)\|^2 \leq 0 \tag{13}$$

and

$$\dot{g} = -2\|q - q_i^f(t)\| \cdot u_\alpha + 2\|q - q_i^f(t)\| \cdot \dot{q}_i^f \tag{14}$$

and $u_\alpha(q)$ has the same form as in eq.(4).

The robots objective is to reach its goal, while maintaining the 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. Figure 7, which appear on the last page of this paper, depicts snapshots of such a scenario. In the first frame (left) the robots begin in some configuration which is not the desired formation, in the second frame (center) each of the robots has determined that it is feasible to get into a formation and does so. The desired configuration is diamond shaped. They continue to drive in the correct formation until, one at a time, each of the robots gets to a point where it is no longer feasible to stay in formation while getting to their respective goals whereupon they break from the formation (right). The then proceed independently to their individual goals. Note that this is achieved in a completely decentralized manner and the decision of when to get in formation or break from it is done online.

## 4 Conclusion

In this paper, a *provably correct method for the parallel composition reactive objectives*, such as obstacle avoidance or following a leader, *with deliberative behaviors*, such as goal seeking, is presented. This is achieved by noting that the traditional control law for solving static motion planning problems, used in conjunction with a Navigation function, is not unique. We derive a parameterized family of control laws which all share a Navigation function as a common Lyapunov function, and hence all represent solutions to the static planning problem. The extra degrees of freedom in the selection of the parameters are then used to locally accommodate reactive requirements, modeled as inequality constraints, online, by solving an optimization problem at each time step.

The merit of our approach is that the algorithm represents a way of locally modifying a motion plan online to accommodate these constraints, while preserving performance guarantees (*i.e.* always reaching the goal), in such a way as to avoid a global replanning whenever possible. If no such solution exists the algorithm reports failure and can alert the high level planner that a global replan is needed.

Future work focuses on modeling other types of applications such as cooperative manipulation and synthesizing switched controllers. The idea behind the first extension is that, in grasping applications, the constraint that the applied contact forces should lie within the friction cone is quite naturally model as an inequality constraint, in the framework of Section 2.2. In fact the notions of force and form closure rely on the concept of *positive spanning* which is very similar in spirit to the ideas behind this work. For the second extension, the basis for switched control of robotic systems, is that the robot has a suite of controllers to choose from and one must design a rule for selecting the appropriate controller at any given time to accomplish certain objectives. In such a situation one could replace our parameterized set of control laws, $\{u_\alpha : \sum_{i=1}^{n-1} \alpha_i^2 \leq 1\}$ with the discrete set of control laws at the robot's disposal $u_j \in \{u_1, u_2, \dots\}$. Then, rather than solving the continuous optimization problem to select the best value of $\alpha$ as in Section 2.3, one must solve a discrete optimization problem to find the controller $u_j$ which is closest to the nominal direction $-\nabla V$ and satisfies the constraints in eq.(9).

## Acknowledgments

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

## References

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

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

[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] J.-C. Latombe, *Robot motion planning*. Boston: Kluwer Academic Publishers, 1991.

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

[10] M. Branicky, *Studies in Hybrid Systems: Modeling, Analysis and Control*. PhD thesis, MIT, Cambridge, MA, 1995.

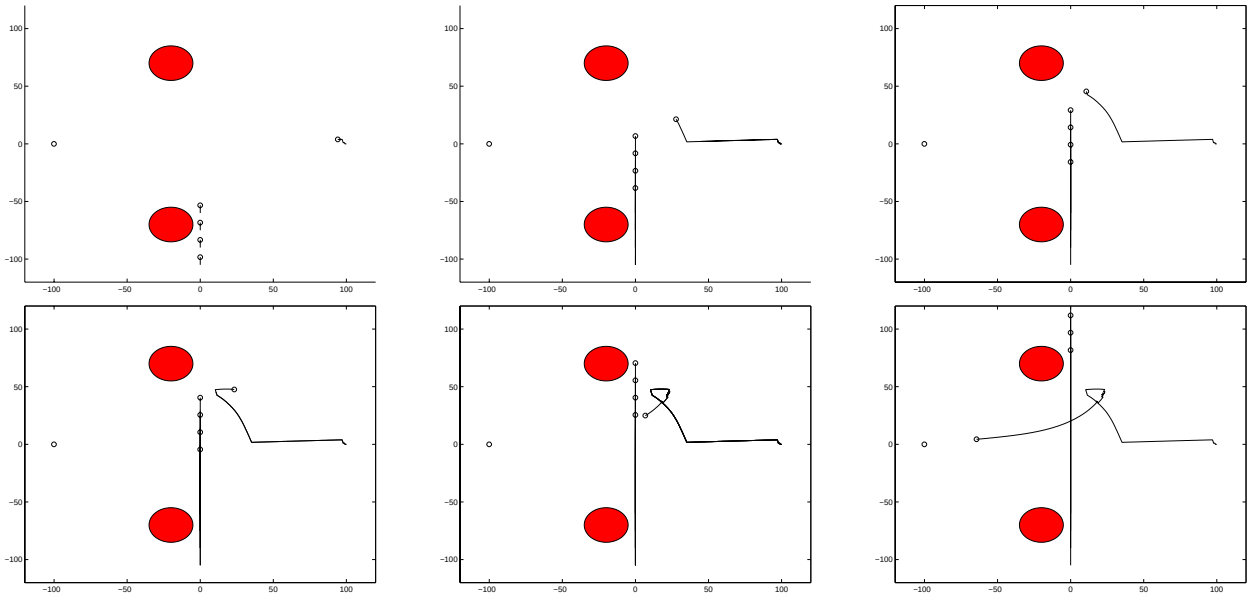

Figure 6: Snap shots of the obstacle avoidance example. Beginning with the upper left hand frame time increases from left to right/top to bottom. Frame 1: Robot (right) proceeds to the goal (left). Frame 2 and 3: The robot tries to steer to the right around the moving convoy. Frame 4: The right pass is blocked by a static obstacle. Frame 5 and 6: the robot loops around to pass the convoy safely on the left.

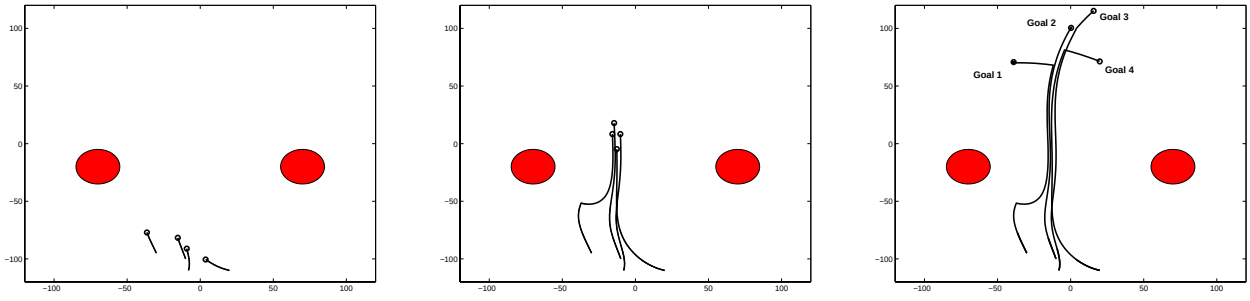


Figure 7: Snap shots of the formation control example. Time increases from left to right. Frame 1: the robots attempt to assume formation. Figure 2: traveling in formation. Frame 3: breaking off from the group to pursue individual objectives.