Cascaded Filter Approach to Multi-Objective Control

Bryan J. Thibodeau, Stephen W. Hart, Deepak R. Karuppiah†, John D. Sweeney, and Oliver Brock

Laboratory for Perceptual Robotics
† Computer Vision Laboratory
Computer Science Department
University of Massachusetts Amherst
{thibodea,shart,deepak,sweeney,oli}@cs.umass.edu

***Abstract*— In this paper we propose a new approach for multi-objective control using a cascade of filters that progressively removes candidate commands which do not satisfy task constraints. The approach is motivated by other control methods that prevent destructive control interactions through null space projections. We apply this approach to a practical leader/follower task in which a mobile robot must address the conflicting objectives of moving to a goal position while avoiding obstacles and keeping a region of the workspace within the field of view of a fixed camera mounted on the platform. We experimentally verify our approach using the Segway Robotic Mobility Platform (RMP), a dynamically stable, differential drive mobile robot.**

I. Introduction

As both the physical capabilities of autonomous robots and the complexity of the tasks we demand of them increase, it becomes imperative for system programmers to have a clear and intuitive framework to implement multiple concurrent behaviors. In general, the robot must be able to make decisions that satisfy multiple, possibly conflicting objectives while utilizing multiple resources with a guaranteed level of performance.

In this paper we propose a new approach for choosing control commands using a cascade of filters that progressively remove candidate commands which do not satisfy task objectives or constraints. The inspiration for this approach has come from analogous work on multi-objective control in the Laboratory for Perceptual Robotics at the University of Massachusetts Amherst [9], [15] in which subordinate controllers can only act without disturbing the progress of higher priority controllers. Our approach also draws inspiration from the domain of multi-objective decision making [12].

We apply this approach to a highly constrained instance of a leader/follower task in which a mobile robot must move to a goal position while avoiding obstacles and keeping a region around the goal within the field of view of a camera mounted on the platform. This behavior is a prerequisite to the task of person tracking and following in the presence of dynamic obstacles, which is critical to many service robotics applications like astronaut assistance in planetary explorations [2] and leader/follower multi-robot search tasks [15], [17]. This is challenging because the task constraints may create complex conflicts between objectives that the control system must resolve.

II. Related Work

A. Multi-Objective Control

The leader/follower task for a team of robots is a well-known problem in the domain of multi-robot formation control [5]. To successfully complete the task, the robots must avoid obstacles while coordinating their motion to maintain communication. Yang [17] presents a null space composition approach for addressing multiple objectives in a leader/follower search task. In our task, we do not consider multi-robot cooperation, however, the multi-objective nature of the problem remains. Other related work includes the AuRA architecture, developed by Arkin *et al.* [1] which computes control actions as weighted, linear combinations of non-linear motor schemata, and the lexicographic method for multi-objective decision making which relies on function optimization [12].

The null space composition approach to multi-objective control applies the concept of null spaces to compose controllers defined by artificial potential fields [9], [13], [16]; subordinate control actions are projected onto the equipotential manifold of superordinate controllers. This null space projection relationship is denoted by the "subject-to" ($\triangleleft$) operator. For example, for two controllers ϕ_α and ϕ_β, $\phi_\beta \triangleleft \phi_\alpha$ means that ϕ_β can only make progress toward its goal in the equipotential manifold of ϕ_α. This guarantees that lower priority control actions do not destructively interfere with the superior controller.

B. Real-Time Obstacle Avoidance

Early approaches for goal seeking while performing collision avoidance are based on potential methods in which local repulsive forces are summed with attractive forces from the goal [10], [11]. The vector field histogram approach [3] builds a polar obstacle density histogram which the robot uses to avoid obstacles. These approaches are susceptible to local minima despite being computationally efficient. Navigation functions such as NF1 [11] and harmonic potential fields [6] overcome this problem by computing globally optimal solutions.

The curvature velocity method [14] and the dynamic window approach [7] use linear weighted objective functions to rank motion commands according to the qualities of their corresponding trajectories. Though these approaches take the dynamic and kinematic constraints of the robot into account when making control decisions, they too suffer from local minima.

The dynamic window approach [7] provides a robust framework for high-speed obstacle avoidance and goal-seeking behavior in unknown dynamic environments. The *dynamic window* contains the velocities of a robot that are achievable in one servo tick given its kinematic configuration and dynamic constraints. This limits the search space of trajectories to those that are physically realizable. The approach chooses the next motion command by searching over the set of translational and rotational velocity tuples (v, ω) in the dynamic window. The search space is further reduced to *admissible* velocities; those that produce collision-free trajectories. Figure 1 illustrates the achievable and the admissible tuples in the space of translational and rotational velocities.

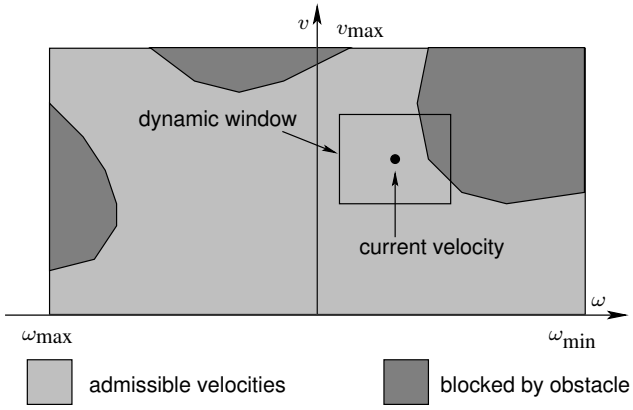


Fig. 1. The search space for the dynamic window approach.

A dynamic simulation is performed for each velocity command within the current dynamic window, under the assumption that the robot achieves the velocity and holds it constant for the remainder of the trajectory. Certain properties such as proximity to obstacles or progress toward the goal are evaluated for the resulting trajectory. A quality value for each trajectory is computed using a weighted linear combination of the properties determined by the dynamic simulation. The trajectory with the highest quality is used to generate motor commands. However, the robot may still get caught in local minima.

The global dynamic window approach [4] overcomes the limitation by incorporating a global planner - the navigation function NF1 [11] - over the discretized workspace, measuring the L_1-norm of the robot to the goal. Descending the gradient of this potential field brings the robot to the goal without hitting obstacles.

The global dynamic window approach guarantees safety by eliminating non-admissible velocities from the search space. We can further shape robot behavior by eliminating regions of the search space that do not satisfy additional objectives. We formalize this notion in the following section.

III. Cascaded Filter Approach

A. Filters

We define a filter to be a function ψ that takes a nonempty set of candidates I and returns a set of candidates $O \subseteq I$ that satisfy the criteria of the filter. In the simplest case, a *threshold* filter ψ selects a candidate c from I if and only if some property of c is above (or below) a threshold.

$$\psi(I, \tau, p, \oplus) \triangleq \{c | c \in I, p(c) \oplus \tau\}, \tag{1}$$

where $p(c)$ is property p of candidate c, $\oplus$ is a comparison operator and τ is a threshold.

Another type of filter is the *best effort* filter $\psi^\star$ which we define as

$$\psi^\star(I, \tau, p, \oplus) \triangleq \psi(I, \tau, p, \oplus) \cup \{c | c \in I, \forall c' \in I \; p(c) \oplus p(c')\}. \tag{2}$$

If no candidate c exists after applying a threshold filter, then the best effort filter selects the element(s) that are closest to the threshold.

An analogy can be made between filters as defined above and controllers as used in null space control compositions. The set returned by a filter is analogous to the null space of a controller, as both define a space in which subordinate filters or controllers can work without disturbing superior filters or controllers. This allows us to use terminology developed for null space control compositions to define cascades of filters. Specifically we can use the "subject-to" ($\triangleleft$) operator introduced above in the context of filter compositions.

For filters ψ_α and ψ_β, $\psi_\beta \triangleleft \psi_\alpha$ means that the input to ψ_β is the output of ψ_α. Thus ψ_β cannot return any control decision not returned by ψ_α, so ψ_β cannot violate constraints introduced by ψ_α.

However, there are important differences between cascaded filters and cascaded controllers that clearly distinguish them. Controllers generate a policy through configuration space, but filters do not generate policies, they remove some subset of the control decisions that constitute their input. In general, it is only possible to use a composition of controllers to generate policies when a continuous artificial potential for task objectives can be computed. If a potential field is not available for a task or the gradient of the potential field cannot be computed in real time, and the properties of control decisions must be determined through some other means such as dynamic simulation, cascaded filters provide an alternative that maintain some of the desirable properties of compositions of controllers.

IV. Implementation

The task for our robot is to travel to a goal location while avoiding obstacles, and keeping the goal within the field of view of the fixed on board camera. The scenario that we use

for experimentation is a *blocked corridor*. The robot starts at one end of a narrow corridor with the goal region to be tracked at the other. As the platform approaches the goal region, an obstacle appears between the robot and the goal, effectively sealing the corridor.

In order for the robot to successfully complete its task without allowing the goal region to leave the field of view of the camera, it must back away and maneuver around the outside of the corridor so that it keeps the goal region within the field of view of the camera. We have identified the following behaviors as necessary for the completion of this task: forward motion, backward motion and stopping. Distinct states are required because the different behaviors require different prioritizations of objectives. Each of these states is defined by a particular filter cascade which corresponds to the prioritization of the possibly conflicting objectives.

The inputs to the filter cascades will be the set of all rotational and translational velocities the platform is capable of attaining. For all cascades, the highest priority filter will select a discrete set of trajectories parameterized by (v, ω). Part of this set will contain achievable velocities determined via the dynamic window [7]. The other part of this set is composed of non-achievable velocities that are opposite the current direction of motion. These trajectories are marked as non-achievable and are never used to directly generate motor commands. We will refer to this filter as the *expanded dynamic window*. By examining the set returned by a filter cascade for non-achievable trajectories we can determine when it is useful to change the state of the robot.

A. Properties of Trajectories

We define a set of properties of trajectories that we use to define our filters for the tracking task. All of these properties, except *achievable* and *translational velocity* are determined via a dynamic simulation of the trajectory.

$l(\cdot)$: **Length** is the distance until a collision with an obstacle.

$\theta(\cdot)$: **Orientation time** is the time that will pass until the subject is no longer in the field of view of the camera.

$\gamma(\cdot)$: **Gradient error** is the difference in the orientation of the front of the robot and the direction of the NF1 gradient after following the trajectory for some predetermined length of time.

$\delta(\cdot)$: **Potential drop** is the maximum drop in NF1 potential between the current position of the robot and any point along the trajectory.

$\Omega(\cdot)$: **Goal** is a Boolean function that indicates if a trajectory ever enters the goal region.

$a(\cdot)$: **Achievable trajectory** is a Boolean function that indicates whether or not a trajectory is achievable based upon the dynamic window.

$v(\cdot)$: **Translational velocity** of the tuple (v, ω) that generates the trajectory.

B. Filters for Tracking

Specific objectives can be realized by properly parameterizing filters with the properties defined above. The values of the threshold τ always have the same units as the property specified in the filter.

a) Expanded Dynamic Window: The filter Ψ_{EDW} returns a subset of the achievable trajectories from the set of all possible trajectories plus some non-achievable trajectories that are opposite the current direction of travel.

b) Safety: This filter removes trajectories that would be unsafe. By setting τ to be the distance required to bring the robot to a full halt under maximum braking, we can guarantee that all of the trajectories in the output are safe with respect to obstacle avoidance.

$$\Psi_{\text{S}} \quad = \quad \psi(I, \tau, l, \geq)$$

c) Viewpoint: This best effort filter removes trajectories in which the target quickly passes out of the field of view of the camera. For example, setting $\tau = 3$ seconds will, if possible, select trajectories that keep the target in the field of view for at least 3 seconds. If this is not possible, it will select the trajectory that will keep the target in the field of view for the longest period of time.

$$\Psi_{\text{V}} \quad = \quad \psi^{\star}(I, \tau, \theta, \geq)$$

d) Gradient Alignment: This filter eliminates trajectories in which the front of the robot is not well aligned with the local NF1 gradient. It can be seen as the combination of a threshold filter and a best effort filter that effects only achievable trajectories. This filter will always have at least one achievable trajectory in its output if there is at least one in its input. If the front of the robot is well aligned with the gradient, the filter's output will contain non-achievable trajectories that can be used to determine the state of the robot.

$$\Psi_{\text{GA}} = \psi(I, \tau, \gamma, \leq) \cup [\psi^{\star}(I, \tau, \gamma, \leq) \triangleleft \psi(I, \text{true}, a, =)]$$

e) Potential Drop: This filter selects the trajectories with the biggest potential drop along their paths.

$$\Psi_{\text{PD}} \quad = \quad \psi^{\star}(I, \infty, \delta, \geq)$$

f) Goal: The purpose of this filter is to select trajectories that bring the robot to the goal position with a velocity less than some threshold velocity.

$$\Psi_{\text{G}} = \begin{cases} \psi^{\star}(I, \tau, v, \leq) \triangleleft \psi(I, 1, \Omega, =) & \text{if } \psi(I, 1, \Omega, =) \notin \emptyset \\ I & \text{otherwise} \end{cases}$$

C. System Description

We designed a finite state machine with five states to address our task. Figure 2 shows the finite state machine and the events that trigger state transitions. Motion commands are generated from a trajectory in C, which is the set of candidates returned by the filter cascade of the current state. The trajectory used for motion control is obtained by applying a simple, state-dependent objective function to the candidates in C.

State 1 corresponds to forward motion. The associated filter cascade is

$$\Psi_{\text{PD}} \triangleleft \Psi_{\text{V}} \triangleleft \Psi_{\text{G}} \triangleleft \Psi_{\text{S}} \triangleleft \Psi_{\text{EDW}}.$$

The element $c \in C$ with the greatest $v(c)$ is chosen to generate motor commands, with ties broken arbitrarily. This filter cascade generates motions that are safe with respect to obstacle avoidance, exhibit goal seeking behavior, and maintain the viewpoint constraint. The last filter in the cascade ensures progress toward the goal.

State 2 corresponds to stopping in order to change from forward motion to backward motion and state 5 corresponds to stopping in order to change from backward motion to forward motion. The associated filter cascade is

$$\Psi_{\text{V}} \triangleleft \Psi_{\text{S}} \triangleleft \Psi_{\text{EDW}}.$$

The element $c \in C$ with the least absolute $v(c)$ is chosen to generate motor commands, with ties broken arbitrarily. This particular filter cascade is chosen since we are only concerned with not violating our tracking constraint while safely stopping.

States 3 and 4 correspond to backward motion. The reasons for differentiating these states are described below. The filter cascade is

$$\Psi_{\text{GA}} \triangleleft \Psi_{\text{V}} \triangleleft \Psi_{\text{G}} \triangleleft \Psi_{\text{S}} \triangleleft \Psi_{\text{EDW}}.$$

The element $c \in C$ with the greatest negative $v(c)$ is chosen to generate motor commands, with ties broken arbitrarily. This filter cascade is identical to the filter cascade in state 1 except for the lowest priority filter Ψ_{GA}. When moving backward, we are not trying to make progress toward the goal, rather, we want to better position the robot to make forward progress in the future. This can be accomplished by aligning the robot with the NF1 gradient, so Ψ_{PD} is replaced with Ψ_{GA}.

The transition from state 1 occurs when all of the candidate trajectories returned by the filter cascade are non-achievable, indicating that forward motion is no longer desirable. The transitions from states 2 and 5 occur when the robot has come to a halt. The transition from state 3 to state 5 corresponds to situations where there are no backward trajectories returned by the filter cascade. State 4 transitions to state 5 when there is any forward velocity returned by the filter cascade. The transition from state 3 to state 4 occurs when the distance traveled in state 3, d, exceeds some threshold τ_d. This helps limit the distance the robot travels backward. This is desirable because the robot has no backward-facing sensors and we do not want to move too far from our target.

V. Experimental Results

A. Robotic Platform

In our experiments, we use the Segway Robotic Mobility Platform (RMP), shown in Figure 3. It is a dynamically stable, differential drive mobile platform, capable of speeds of 12 km/h. A 2.4 GHz Mobile Pentium 4 laptop with 1 GB of RAM running Linux is mounted on the RMP. We interfaced

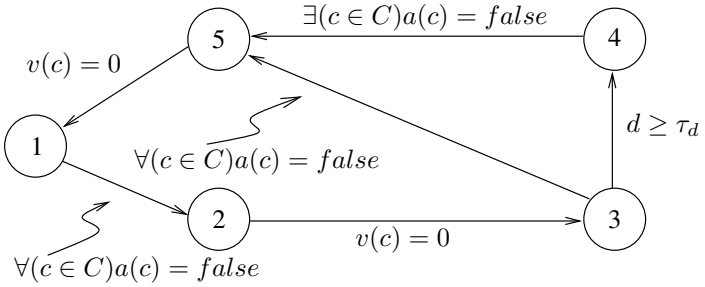


Fig. 2. State Transition Diagram. Each transition is labeled with the conditions that trigger the transition. C is the set returned by the filter cascaded in the originating state. $v(c)$ is the current velocity of the robot. d is the distance traveled since entering the originating state 3.

Fig. 3. The Segway RMP equipped with a laser range finder and a camera. The PVC pipes are a safety system that protects the on board equipment from damage in case of system failure.

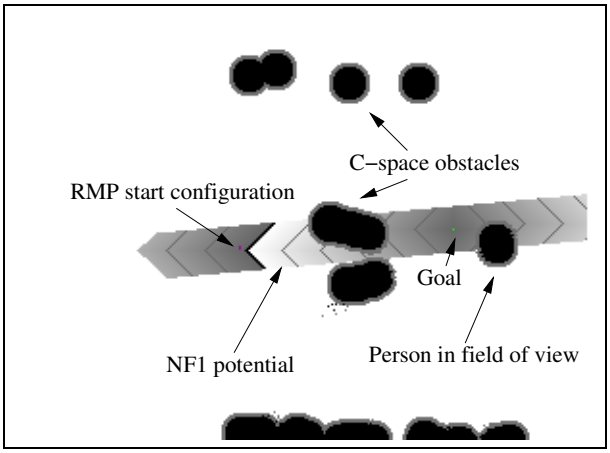
with the RMP hardware through a Kvaser PCMCIA CAN interface. The RMP also has an 802.11b wireless hub for off board communication.

We attached a SICK laser range finder and a Sony color camera to the top of the platform. For these experiments we treat the camera as if it were fixed. The laser range finder has a 180° field of view in the horizontal plane. The camera, affixed to a mast near the center of the platform, has a field of view of approximately 70°, and is used to track a target. We impose a viewpoint constraint that limits the position of the goal in the camera image to be less than $\pm$ 34.4° (600 milliradians) from the center of the image. In this work, it is assumed that all obstacles are tall enough to be detected by the laser, but short enough not to occlude the view of the camera. We use the Player/Stage [8] mobile robot control architecture to drive the RMP.

Figures 4(a)–4(f) show images of the experiment and snapshots of the configuration space maps generated for the RMP while executing the task. In these configuration space maps, black areas are obstacles and white areas denote free space. The NF1 potential function, computed for a limited portion of the configuration space, as in [4], is shown as a gray region. The global minimum of this function is at the goal.

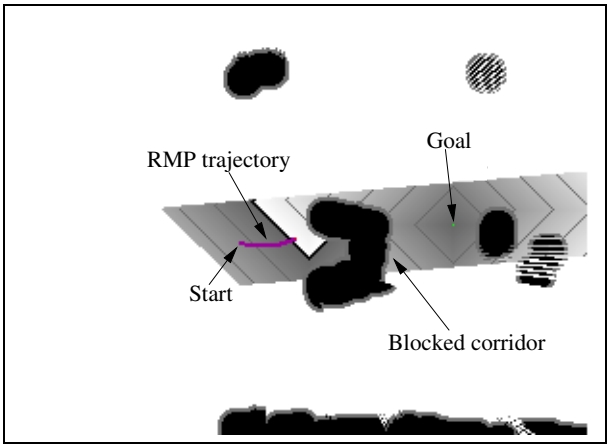
Figures 4(a) and 4(b) show the start and goal positions

(a)

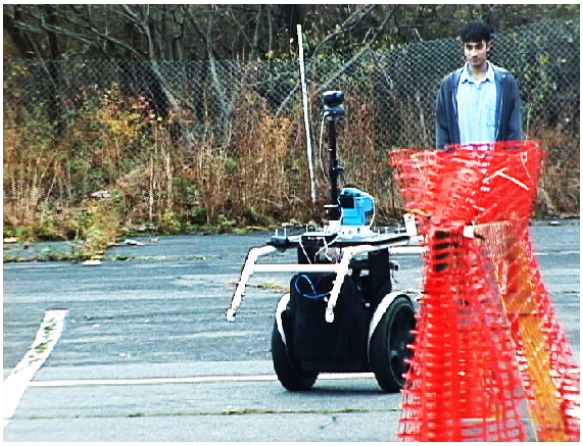


(b)

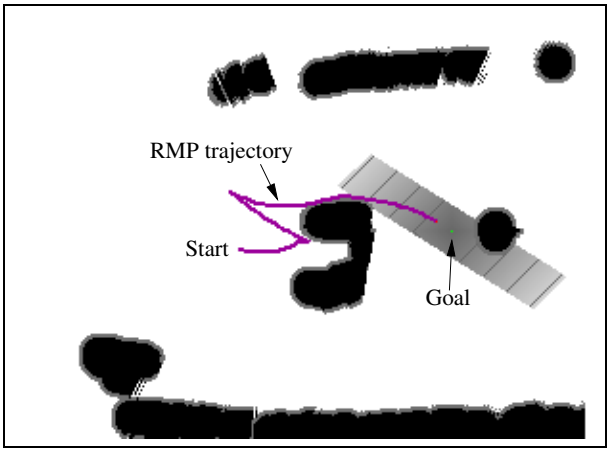
(c)



(d)



(e)



(f)

Fig. 4. (a) This picture shows a still shot from the video feed from the camera mounted on the RMP at the beginning of the experiment. The person standing in the middle of the field of view is slightly behind the goal region. The most direct path to this region is through the corridor. (b) Configuration space of the RMP : dark areas are the obstacles, and white area is the free space. The NF1 potential function is superimposed in this space and its global minimum is at the goal. (c) This video capture shows the robot as it starts to back out the corridor. (d) The far end of the corridor is blocked, forcing the RMP to stop. To maintain the viewpoint constraint, the RMP must now choose a trajectory that takes it out of the corridor while keeping the goal location within the field of view. (e,f) The RMP successfully reaches the goal after maneuvering around the blocked corridor.

assigned to the RMP for the task. We can see that a clear path exists through the corridor that satisfies all constraints imposed by the filter cascade. As the RMP attempts to pass through the corridor a dynamic obstacle blocks its path (Figures 4(c) and 4(d)). At this point in the experiment the objective of moving quickly toward the goal conflicts with the objective of keeping the goal location in the field of view of the camera. Since none of the forward trajectories can take it safely to the goal without violating the viewpoint constraint, the RMP comes to a stop by transitioning to state 2. After that, it immediately transitions to state 3 where the filter cascade imposes backward motion to take it out of the corridor. By moving backward, the viewpoint constraint continues to be satisfied. After backing far enough away, the RMP reaches a position where it is again possible to move forward toward the goal. Figure 4(f) shows that the RMP successfully reaches the goal after maneuvering around the corridor. Figure 5 is the plot of orientation error versus time and shows that the viewpoint constraint was met throughout the run. This experiment demonstrates that the cascaded filter method can successfully control a robot in real time in the presence of conflicting constraints.

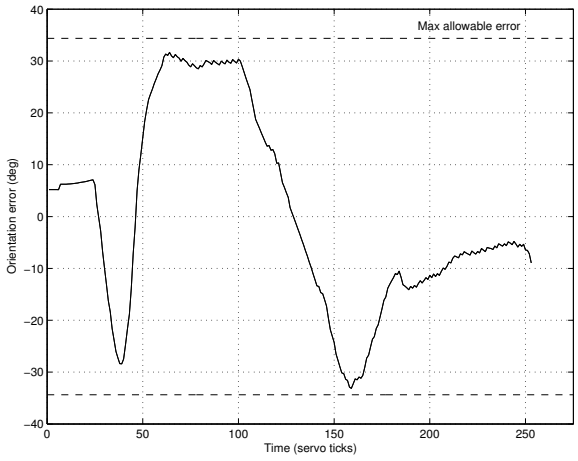


Fig. 5. Plot of orientation error versus time. Orientation error remains within the imposed constraint throughout the run.

VI. Conclusion

We present the filter cascade method for multi-objective control and apply it to a viewpoint-constrained, goal-seeking task. Multiple objectives can be addressed by creating prioritized cascades of filters; lower priority objectives are forced to choose only those actions that satisfy higher priority objectives. The approach provides a simple and intuitive design process for multi-objective control problems by translating task specifications into filters. We have experimentally verified the approach using the Segway RMP mobile robot.

Acknowledgment

This work was supported in part by NSF CDA-9703217, and NASA NAG9-1445#1. The authors would like to thank Andy Fagg, Rod Grupen, Emily Horrell, and Shichao Ou and others in our laboratory for their help in running the experiments. We would also like to thank Bob Barnicle and Construction Services of University of Massachusetts Amherst for lending the obstacles.

References

[1] R.C. Arkin and T. Balch. AuRA: Principles and practice in review. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2), 1997.

[2] E. M. Atkins, J. A. Lennon, and R. S. Peasco. Vision-based following for cooperative astronaut-robot operations. In *Proc. IEEE Aerospace Conference*, Big Sky, MT, March 2002.

[3] J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. In *IEEE Transactions on Robotics and Automation*, volume 7(3), pages 278–288, 1991.

[4] O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 341–346, Detroit, MI, 1999.

[5] Y. Uny Cao, Alex S. Fukunaga, and Andrew B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomuos Robots*, 4:1–23, 1997.

[6] C. Connolly and R. Grupen. Nonholonomic path planning using harmonic functions. Technical report, University of Massachusetts. Amherst, MA, 1994.

[7] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. In *IEEE Robotics and Automation Magazine*, volume 4(1), pages 23–33, March 1997.

[8] B. P. Gerkey and R. T. Vaughan. Most valuable player: A robot device server for distributed control. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1226–1231, Hawaii, 2001.

[9] M. Huber and R. A. Grupen. A feedback control structure for on-line learning tasks. *Journal of Robots and Autonomous Systems*, 22(3-4):303–315, 1997.

[10] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Intl. J. of Robotics Research*, 5(1):90–98, 1986.

[11] J. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[12] Paolo Pirjanian and Henrik I. Christensen. Behavior coordination using multiple-objective decision making. *Sensor Fusion and Decentralized Control in Autonomous Robotic Systems*, 3209:78–89, October 1997.

[13] R. Platt, A. H. Fagg, and R. Grupen. Nullspace composition of control laws for grasping. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, 2002.

[14] R. Simmons. The curvature-velocity method for local obstacle avoidance. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, volume 4, pages 2275–82, Minneapolis, 1996.

[15] John D. Sweeney, TJ Brunette, Yunlei Yang, and Roderic A. Grupen. Coordinated teams of reactive mobile platforms. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Washington, D. C., 2002. IEEE.

[16] John D. Sweeney, Huan Li, Roderic A. Grupen, and Krithi Ramamritham. Scalability and schedulability in large, coordinated, distributed robot systems. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Taipei, Taiwan, 2003. IEEE.

[17] Yuandong Yang, Oliver Brock, and Roderic A. Grupen. Exploiting redundancy to implement multi-objective behavior. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Taipei, Taiwan, 2003.