Exploiting Redundancy to Implement Multi-Objective Behavior

Yuandong Yang Oliver Brock Roderic A. Grupen
Laboratory for Perceptual Robotics
Department of Computer Science
University of Massachusetts Amherst
Email: {yuandong, oli, grupen}@cs.umass.edu

Abstract

Teams of robots can be redundant with respect to a given task. This redundancy can be exploited to pursue additional objectives during the execution of the task. In this paper, we describe a control-based method to exploit such redundancy for the execution of additional behavior, leading to the improvement of overall performance. The control-based method provides a suitable mechanism for combining controllers with different objectives. The mechanism ensures that the subordinate controllers do not interfere with the superior controllers. Thus it allows to build controllers exhibiting complex behavior from simple primitives, while maintaining their provable performance characteristics. The effectiveness of the framework is demonstrated by experiments with a multi-robot exploration task.

1 Introduction

Distributed control methods for multi-robot teams have been an active area of research. Individual members of a team are controlled independently in such a way that a desired behavior of the group emerges. Robots observe the world, abstract, and act independently and concurrently. The main objective and difficulty in designing distributed controllers is the coordination of team members.

Various methods have been introduced for distributed control tasks. We distinguish two approaches: top-down and bottom-up methods. In top-down methods, the controller is recursively divided into less complex units. The division continues until all units can be physically implemented. The resulting controller is generally well suited to solve the given task, but might fail to generalize easily to a different task.

Bottom-up methods, on the other hand, build small modules first. Higher-level behavior is achieved by combining lower-level modules. Once the lower-level modules are built, they can easily be combined into different controllers addressing varying task requirements. Consequently, bottom-up methods can be considered more versatile and flexible compared to top-down methods.

Within the framework of bottom-up methods, the construction process for controllers requires two steps: the design of versatile modules and their combination into controllers. The resulting modules should be robust, while methods to combine them should maintain their desirable properties. In this paper, we introduce a control-based framework to address these issues.

Our framework allows to specify a "universe" of controllers by combining *objective functions* representing a desired behavior, *state estimators* to perceive information about the state of the world, and *effectors* to modify the state of the world. Each combination of instantiations from these categories can be viewed as a control option. Desired behavior is achieved and its performance can be guaranteed by employing closed-loop disturbance rejection.

The control-based framework presented here also specifies how controllers are combined. The controllers are ranked based on their importance for the overall behavior. Actions of subordinate controllers are projected into the nullspace of superior controllers. This mechanism ensures that the performance of the superior controller remains unaffected, while performing additional behavior whenever possible.

We demonstrate the effectiveness of this control-based method for multi-objective robot control by applying it to the task of exploring an unknown environment with a team of robots. The experiments presented in Section 4 demonstrate how the redundancy of the team can be exploited to add secondary objectives, which significantly improve the overall performance of the team.

2 Related Work

We will restrict our discussion of previous work to those methods classified as bottom-up, as described above. The behavior-based control method [1][6] is one of the successful ones. It replaces behaviors couched in expensive representations with local control decisions and sensor information. The subsumption architecture arbitrates actions using inhibition and suppression mechanism between behavioral modules.

Several people have contributed to the area of behavior-based methods. Maes [5] developed a method by which robots learn how to combine predesigned basic behaviors. Hoff [3] designed algorithms to learn both the basic behaviors with sub-goals and how to combine them to achieve more complex behaviors. Mataric [7] gave an impressive demonstration of behavior-based control on real robots. Two behaviors can be combined into new higher-level behaviors by behavior combination operators. Reinforcement learning is applied to let robots learn how to construct higher level behaviors automatically.

Behavior-based methods have their advantages, but we want to argue that they have some problems. The simple combination of two or more basic behaviors cannot ensure the execution of either behavior. Reinforcement learning can be employed to identify functioning combinations, but the controllers it chooses are unlikely to be generalizable and will probably fail if the environment or other conditions vary.

Huber and Grupen [4] introduced the idea of control-based methods. They can be still classified as behavior-based methods, but it is more formalized. Huber successfully applied the method to learning to control legged-robots [4]. The robots learn how to walk without falling, which is guaranteed by the virtue of discrete event specifications.

In this paper, we extend this technique following the work of Sweeney [9] that treats multiple mobile robots with multiple concurrent objectives; namely searching a multi-room floor plan while maintaining in a connected communication network.

3 Exploiting Redundancy using Control-Based Methods

3.1 Specifying Controllers

A set of controllers C can be described using a vocabulary of objectives Φ, sensors S, and effectors E:

$$C = \Phi \times S \times E$$

A specific controller c then is given by the tuple (ϕ, s, e) with $\phi \in \Phi$, $s \in S$, and $e \in E$. The objective function ϕ is measured using the sensor s and continuous closed-loop control is employed to optimize ϕ using the effector e. Closed-loop control ensures robust performance by continuous disturbance rejections. Note that this framework is very general and that computational resources, sensors, and effectors do not have to reside in physical proximity. Our notation for such a controller, also called control primitive, is given by

$$\phi|_e^s, \text{ with } \phi \in \Phi, s \in S, e \in E.$$

Applying this formalism to the domain of teams of mobile robots we will use the notation $\phi|_i^s$throughout the remainder of this paper. The objective function ϕ is represented by an artificial potential field. Robot i uses a sensor s to descend the gradient of ϕ.

3.2 Combining controllers

Control primitives as described above perform single objectives, represented by ϕ in a robust manner. Our goal is to combine many such primitives into a complex controller without sacrificing robustness. We determine a ranking of control primitives in such a way that the behavior of a subordinate controller should never interfere with a superior controller. This can be accomplished by projecting the actions resulting of a subordinate controller into the nullspace of the superior controller.

If the task of a robot is specified by the vector $\dot{x}$ and the robot is redundant with respect to that task, the robot can perform a set of actions not affecting task performance. These actions are said to be in the nullspace of the Jacobian matrix J, associated with the task.

The control of the robot is composed from two components: one representing task behavior, $J^{\#}\dot{x}$, and a second one representing additional, subordinate behavior K. The subordinate behavior is projected into the nullspace $(I - JJ^{\#})$ of J to not affect the execution of the superior task:

$$\dot{\theta} = J^{\#}\dot{x} + (I - JJ^{\#})K, \tag{1}$$

where $J^{\#}$ designates the Moore-Penrose inverse of J, I is the identity matrix, $K = \frac{\delta p}{\delta x}$ represents the gradient of the potential function p of a subordinate controller, and $(I - JJ^{\#})$ is the nullspace projection of J.

We use the notation $\phi_1 \triangleleft \phi_2$, or ϕ_1 "subject to" ϕ_1, to express the nullspace relationship between two primitives ϕ_1 and ϕ_2. Primitive ϕ_1 is subordinate and must be performed in the nullspace of superior primitive ϕ_2. Consequently, actions resulting from ϕ_1 will

only be performed if they do not affect the execution to ϕ_2.

The approach of combining control primitives permits to safely pursue multiple objectives as long as subordinate controllers do not affect the behavior of superior ones. The control primitives are constructed from sets of objective functions, sensors, and effectors; by employing closed-loop disturbance rejection the robustness of the primitive as well as the robustness of the combined controller can be guaranteed. We can continue this process of combining controllers, until the resulting nullspace does not suffice any more to perform a certain desired behavior.

3.3 Exploiting Redundancy

We will now apply the framework presented in the previous section to teams of mobile robots. In such teams each member can be treated as a sensor-effector pair – it has the ability to sense and change the environment. If the team resources exceed the requirement of the task, we say that the team exhibits *redundancy* with respect to that task. For example, in a single-objective exploration task, the robot always follows the steepest descent of the potential function representing the task in order to minimize the exploration time. A team of robots, however, will be redundant with respect to the exploration task.

In the remainder of the paper we will demonstrate how we use the framework introduced in this section to exploit the redundancy encountered in robot teams. We present three different concurrent controllers consisting of multiple control primitives. Each controller solves the exploration task. The controllers differ in additional, subordinate primitives, relying on redundancy to improve the overall performance during the exploration task. Exploration is performed in a multi-room floor plan represented by a grid, measuring 32 by 32 cells. The task of the robots is to explore all the cells with infrared sensors while maintaining a line-of-sight constraint. This constraint requires the robots to maintain an unobstructed line of sight and to exceed a given distance from each other. This constraint is motivated by communication requirements between robots using radio signals with limited range.

The robots need to plan its path to all unexplored regions while avoiding hitting any obstacles. We use harmonic functions[2, 8] to generate the motion of the robots. Harmonic functions have desirable properties in dynamic environments. All the obstacle cells are assigned a potential value of 1 and the goal cell is assigned a potential value of 0. We use dynamic programming to compute the potentials.

The following three controllers were used to demonstrate that redundancy can be exploited to increase performance. Each of them combines at least two behaviors for an exploration task performed by a team of two robots.

(A) Line-of-sight exploration controller: The line-of-sight exploration controller is implemented in Sweeney's work [9]. The exploration of the robots is frontier-based [10]: the robots are always heading to the biggest boundary between free space and unexplored space using artificial potentials and gradient descent. Equation 2 describes the line-of-sight exploration controller in the framework introduced above:

$$\Phi|^{s}_{\{L,F\}} = \{\phi|^{s}_{L} \triangleleft \phi|^{LOS_L}_{F}\}, \tag{2}$$

where s refers to the infrared sensors of the robots, LOS stands for line of sight, and L, F refer to the leader and follower, respectively. The subordinate exploration primitive $\phi|^{s}_{L}$ of the leader is performed in the nullspace of (or "subject to") the primary line-of-sight control primitive $\phi|^{LOS_L}_{F}$. Any motions resulting from the exploration primitive will be projected into the nullspace of the line-of-sight primitive. Thus, the line-of-sight constraint will be always maintained.

The additional line-of-sight constraint might increase exploration time with respect to a single-robot exploration, because the leader has to ensure that the follower is able to maintain visibility. This controller will serve as our reference controller. In what follows we propose modifications to this controller with the goal of speeding up the exploration process. Improvement will be measured relative to the line-of-sight exploration controller, or reference controller.

(B) Line-of-sight exploration with collaborative search behavior: In this modification of the reference controller, the goal of the follower to perform its own exploration by maintaining the line-of-sight constraint to the leader. The line-of-sight constraint bounds the distance between the leader and the follower from above. The exploration primitive for the follower also imposes a lower bound. The follower is moving towards an unexplored region within those bounds, allowing it to assist the leader in exploring the maze while still maintaining the line-of-sight constraint. This also forces the motion of the follower to be mapped into the nullspace of the line-of-sight controller.

Equation 3 describes the line-of-sight controller with collaborative search behavior using the notation introduced earlier. A third control primitive $\phi|^{s}_{F}$ is added (see Equation 2). The collaborative search primitive is subject to the follower performing the line-of-sight control primitive. The motion of the follower

will be projected into the nullspace of the line-of-sight primitive.

$$\Phi|^s_{\{L,F\}} = \{\phi|^s_L \lhd \phi|^{LOS_L}_F \rhd \phi|^s_F\} \quad (3)$$

The controller in the equation 3 can be rewritten as separate controllers for the leader and follower: $\phi|^s_L \lhd \phi|^{LOS_L}_F$ and $\phi|^{LOS_L}_F \rhd \phi|^s_F$. The first part can be interpreted as the leader exploring the environment, while "pulling" the follower along. The second part represents the follower "pushing" the leader ahead by proceeding into unexplored regions in which the line-of-sight constraint is maintained. The follower's explorative motion is mapped into the nullspace of $\phi|^{LOS_L}_F$, the line-of-sight controller, preventing any motions violating the primary LOS constraint.

In Figure 1 the traces of the two robots performing exploration using this controller are shown. In the middle of the figure, we can see that the follower moves to the right of the leader, assisting in the exploration of an unexplored region. Such behavior reduces exploration time.

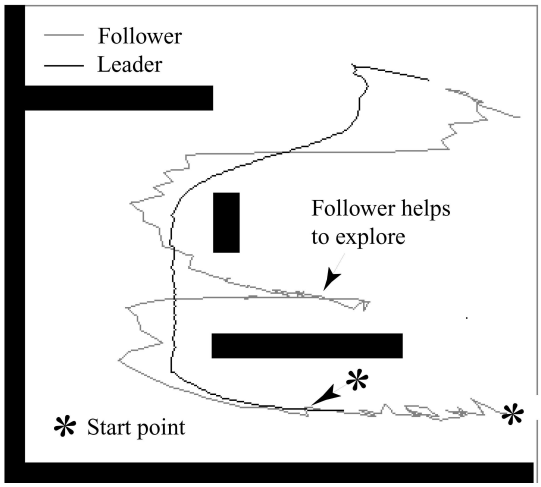

Figure 1: The traces of leader/follower with collaborative search

(C) **Line-of-sight exploration with collaborative search and role-change behavior:** Based on controller B, we try to further improve the performance by adding an additional behavior. We employ the ability of the leader and the follower to change roles during the exploration process. A role change can save exploration time when a change in direction occurs. Consider the example of exploring a room. At first, the follower follows the leader into that room. After the leader has completed the exploration, both robots need to exit the room. The leader's motion will force the follower out of the room. If the robots switch roles instead, however, the leader does not have to travel the distance to pass the follower and the exploration can resume faster. A role-change is always in the nullspace of the line-of-sight controller because it does not change the distance between the leader and follower.

The decision to change roles is based on the harmonic potential. We notice that the paths of the robots are mostly determined by obstacles in proximity. Once these have been detected, the corresponding potential surface for the leader and the follower will looks similar. As a result, the paths chosen by the two robots will be almost identical. A monitor is added to the line-of-sight exploration controller, monitoring if the leader and the follower are on similar paths and if the follower is ahead of the leader with respect to the direction of motion. If this situation occurs for longer than a predetermined interval, the leader and follower change roles. Note that the role-change always is in the nullspace of the line-of-sight behavior.

4 Experimental Results

We applied the three controllers introduced in the previous section to exploration tasks in 1920 randomly generated mazes. The result is shown in Table 1. The exploration time is measured in terms control cycles of the simulator required to complete the exploration. In each control cycle, the robots communicate with each other and execute an action. Figure 2 shows the traces of the robots controlled by the three controllers in an identical maze. For the line-of-sight controller, an average 870 of control cycles were needed to complete the exploration. Because the follower is fast enough to follow the leader, the exploration time with line-of-sight constraint is only a marginally longer than the exploration with a single robot.

We evaluate the three controllers presented here by comparison relative to the reference controller. The comparison is shown in Table 1.

The line-of-sight controller with collaborative search behavior exhibits an average improvement of 30% compared to the reference controller. This improvement is a result of the follower providing additional information about obstacles, effectively increasing the sensing range of the leader, and thus assisting with the exploration within line-of-sight distance. The standard deviation in the improvement can be explained by the fact that the environment initially is completely unknown. A short-term gain does not necessarily brings a long-term gain. Exploration decisions made locally might not be optimal globally. But

	LOS	LOS with CS	LOS with both CS and RC
average exploration control cycles	857.2	578.2	565.9
standard deviation	209.2	115.7	112.8
average improvement(percent)	N/A	30.0	31.5
standard deviation(percent)	N/A	17.9	17.4

Table 1: Comparison of three controllers in 1920 simulated exploration tasks. LOS stands for line-of-sight exploration; RC stands for role-change; CS stands for collaborative search.

	$\frac{1}{1}$ sensor range	$\frac{3}{4}$ sensor range	$\frac{1}{2}$ sensor range
infrared range in grid cells	8	6	4
average exploration control cycles	578.2	764.9	1204.9
standard deviation	115.7	120.0	187.1
average improvement(percent)	30.0	22.1	11.8
standard deviation(percent)	17.9	15.6	16.6

Table 2: Comparison of three line-of-sight controllers with collaborative search behavior, using differant infrared sensor ranges.

even within the range of the standard deviation there is considerable improvement compared to the reference controller.

The controller with collaborative search *and* role-change behavior resulted in an average improvement of 31.5%. This improvement is not significant relative to the controller with collaborative behavior. The small magnitude of this improvement can be explained as follows: Each time leader and follower change roles, they only save the exploration distance between the leader and the follower. These distances are small compared to the total required travel for exploration. Consequently, the exploration time gained by role-switching is marginal compared to the overall exploration time.

The degree to which the proposed controllers are able to reduce the overall exploration time of a given environment is dependent upon the sensing capabilities of the robots. To demonstrate this we performed experiments with different sensing ranges for the infrared sensors. Table 2 compares the performance of the line-of-sight exploration controller with collaborative for various sensor ranges. If the range is shortened to $\frac{3}{4}$ of the range for the experiments reported in Table 1, we only observe an average improvement of 22% with respect to the reference controller with the same sensor range. Further reducing the range to $\frac{1}{2}$ of the original one, reduces the average improvement to only 11%. This is due to the fact that in order to maintain the line-of-sight constraint, leader and follower have to be able to perceive each other. Reducing the range of the infrared sensors also reduces the distance they can be apart without violating the line-of-sight constraint. This limits the capacity of the follower to perform independent exploration, thus deteriorating the overall performance.

These results presented in this section demonstrate that by exploiting the redundancy of teams of robot with respect to a given task using a control-based nullspace scheme, we can significantly improve the overall performance. Note that while the experiments described above are performed with only two collaborating robots, the general framework extends to an arbitrary number of additional robots. These could either have to maintain a line-of-sight constraint with the leader or adopt other followers as their leaders.

5 Conclusion

In this paper, we presented a formal framework to combine control-based methods to exploit the redundancy of a team of robots with respect to a given task. The framework allows the incremental integration of control primitives. By performing each additional behavior in the nullspace of the original controller, we can guarantee that the added behavior will not affect the performance of existing controllers. As a consequence, added controllers can be used to implement more complex behavior and to improve the overall performance in a robust manner.

We presented experiments, demonstrating the effectiveness of the proposed approach. The experiments apply the framework to the exemplary application of team-based exploration of unknown environments. Augmenting a reference controller for this task

with additional behaviors, we were able to show significant improvements in exploration speed. The experiments conclusively show that our framework is capable of combining various behaviors in a prioritized manner to result in more sophisticated and advantageous overall behavior of the team.

Acknowledgments

The authors would like to thank John Sweeney for their helpful insights and discussion in preparing this paper. Work on this paper has been supported in part by MARS/SDR, NSF CDA-9703217, DARPA/ITO DABT63-99-1-0022, and DABT63-99-1-0004.

References

[1] Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence Journal*, 47:139–159, 1991.

[2] C. Connolly and R. Grupen. Harmonic control. In *Proceedings of the IEEE International Symposium on Intelligent Control*, pages 503–506, Glasgow, Scotland, Aug. 1992.

[3] J. Hoff and G. Bekey. An architecture for behavior coordination learning. In *IEEE International Conference on Neural Networks*, pages 2375–2380, Perth, Australia, Nov 1995.

[4] M. Huber and R. A. Grupen. A control structure for learning locomotion gaits. In *Proceedings of Seventh International Symposium on Robotic and Applications*, Anchorage, AK, May 1998. TSI Press.

[5] P. Maes and R. A. Brooks. Learning to coordinate behaviors. In *National Conference on Artificial Intelligence*, pages 796–802, 1990.

[6] M. Mataric. Behavior-based control: Main properties and implications. In *Proceedings of the IEEE International Conference on Robotics and Autonomation, Workshop on Architectures for Intelligent Control Systems*, pages 46–54, 1992.

[7] M. J. Mataric. Interaction and intelligent behavior. Technical Report AITR-1495, MIT, 1994.

[8] E. Prestes, E. Silva, P. M. Engel, M. Trevisan, and M.A.P. Idiart. Exploration method using harmonic functions. *Journal of Neurophysiology*, 40:25–42, July 2002.

[9] J. Sweeney, TJ Brunette, Y. Yang, and R. Grupen. Coordinated teams of reactive mobile platforms. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 299–304, 2002.

[10] B. Yamauchi. A frontier based approach for autonomous exploration. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 146–151, Monterey, CA, July 1997.

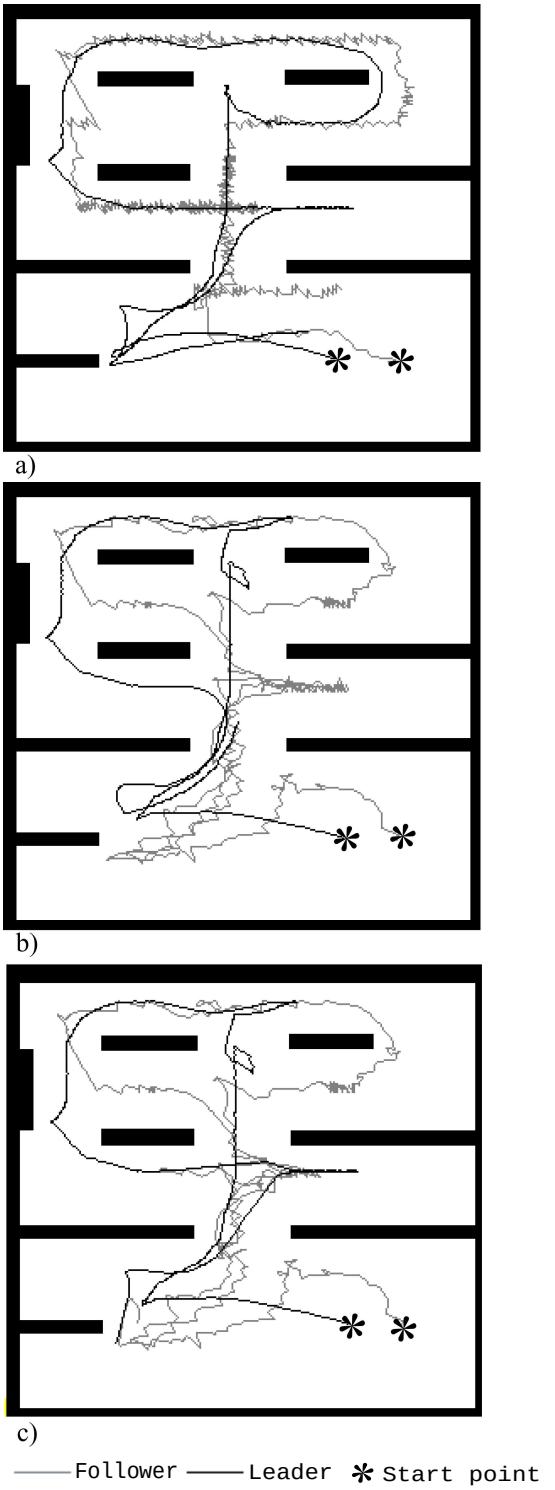

Figure 2: The traces of leader and follower controlled by the three controllers in the same maze: a) line-of-sight exploration, 720 control cycles; b) line-of-sight exploration with collaborative search, 531 control cycles; c) line-of-sight exploration with both collaborative search and role-change, 492 control cycles and 3 role-changes.