Fast and Frugal Autonomous Sustain and Resupply: Final Report

Submitted to DRDC in support of Contract W7702-07R146/001/EDM

Richard T. Vaughan Jens Wawerla Yaroslav Litus Adam Lein
Alex Couture-Beil Abbas Sadat
Greg Mori

Autonomy Lab, School of Computing Science,
Simon Fraser University, Burnaby V5A 4S6, Canada

email: vaughan@sfu.ca phone: 604 291 5811 fax: 604 291 3045

1 May 2010

Contents

Chapter 1

Overview of this document

The *Fast and Frugal Autonomous Sustain and Resupply Project* was a two-year effort of the SFU Autonony Lab for DRDC Suffield, with Technical Authority Greg Broten, during 2008-2010. This final report collects our research papers from the final year of the project. It is a continuation of the previous report [1], which introduced our approach and research plan, provided a review of the state of the art, and collected the research papers from the first year (2008-2009). For brevity, the material from the first report is not reproduced here.

This document includes only results that have been peer-reviewed at major journals and international conferences, and published with the permission of DRDC. We can submit further informal documentation on other work to DRDC on request. This policy is designed to assure DRDC of the quality of the work described here.

DRDC's Greg Broten visited the Autonomy Lab on 22 March 2010 to review demonstrations of the systems described here plus some preliminary work not yet published.

1.1 Approach and outcomes

Target Problem: *to devise robot control algorithms and implemented code that guide a team of robots to deliver resources between dynamic sources and sinks, minimizing the time that sinks spend needing resources, while maintaining the robots'internal consumables to ensure continued operation.*

We considered the target problem at two levels: (i) a formal optimization problem; and (ii) a real-time robot behaviour generation problem. The robot systems we implemented were based mainly on heuristic methods that find good-enough approximate solutions, but formal analyses and optimal solutions are provided for various sub-problems where feasible.

The results of the work, collected in this document and the Year 1 report, include:

1. several original robot controllers that solve aspects of the target problem;
2. a novel theoretical analysis of the generalized work/recharge action selection problem;
3. new concepts in recharging and rendezvous methods for multi-robot teams;
4. extensive improvements in performance and correctness of the Stage multi-robot simulator, maintaining its leading position;
5. the first demonstration of gaze-and-gesture-based control of multiple robots by an uninstrumented human;
6. a proposal for improving research practices in our community.

A project home page including bibliography and downloadable papers is at
`http://autonomy.cs.sfu.ca/fasr.html`

1.1.1 Results statistics

This project was a major source of support for the following formal publications:

- 3 journal papers [2, 3, 4]
- 10 refereed conference papers [5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
- 2 PhD theses: [15, 16]
- 2 MSc theses: [17, 18]

1.2 Outline

- Chapters 2-4 describe new methods for managing descructive spatial interference in robot teams.
- Chapters 5 and 6 decribe new ideas for managing recharging stations when used with multiple robots.
- Chapters 7 and 8 investigate the application of behavioural ecology foraging theory to task switching for individual robots.
- Chapter 9 presents a very simple and effective "Fast and Frugal" solution to the target problem.
- Chapter 10 is a proposal for improving robotics research methodology, which is followed in most of the other chapters.
- Chapter 11 is a novel human-robot interaction system, inspired by the problem of putting humans "into the loop" of the target problem.

1.3 Acknowledgments

The authors wish to thank Greg Broten at DRDC for his support and guidance throughout the project, and to many editors and anonymous reviewers for helping to improve the individual papers. Warm thanks also to the examiners of the student PhD and MSc theses, and to Gaurav Sukhatme and Owen Holland for their continuing advice and mentorship of Vaughan and the Autonomy Lab. Thanks to our consultants and friends Brian Gerkey and Nate Koenig, and for inspiration, our colleagues at Willow Garage, Bristol Robotics Labs, the University of Southampton and the University of Southern California.

Chapter 2

Adapting to non-uniform resource distributions in robotic swarm foraging through work-site relocation

Adam Lein and Richard T. Vaughan [11]

2.1 Abstract

We describe a simple controller for swarms of foraging robots that reduces mutual spatial interference and adapts to non-uniform resource distributions. We discuss several sources of such non-uniformity, and show that some non-uniform distributions are not well-handled by previously described foraging schemes. Our new method adapts each robot's work site size and location during run time, to reflect the distribution encountered. The controller is very scalable, using only local information and no explicit communication. Simulation studies demonstrate the method's effectiveness.

2.2 Introduction

Central-place foraging is a well-studied task in robotics and ethology [19]. The task requires agents to locate and collect spatially distributed items and deliver them to a "home" location. The delivery rate can be increased by adding more foragers, but typically with diminishing returns due to mutual spatial interference.

This interference effect creates a problem for foraging robot controllers. To maximize performance, we want robots to work where there are items to collect ("pucks" hereafter). Yet we need to keep the robots spread out to minimize interference. When pucks are clustered together, these two goals are mutually exclusive. Figure 2.1 shows an example of this difficult domain that is the focus of this paper.

In animal foraging literature, resources are frequently described as occurring in "patches" or "food-source locations", or otherwise referred to as clusters implicitly in discussions of isolated food sources distant from a nest site [20, 21, 22]. Similar language and environmental setups are used in ant-inspired artificial agent research, often in the context of pheromone trail-following [23, 24].

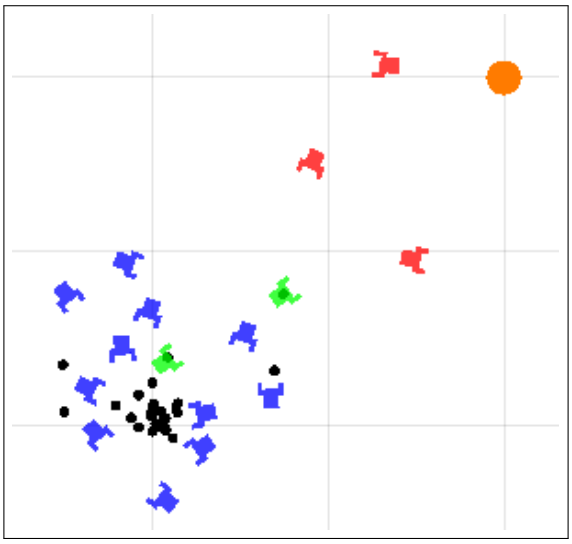


Figure 2.1: Robots in simulation foraging around a cluster of pucks. Robots are indicated by boxes with "arms", and pucks by solid black circles. Blue/dark robots are searching, green/light robots are homing, and red/medium robots are returning to their work areas. The larger circle in the northeast corner is the home zone.

Research in robot foraging has addressed scenarios in which resources are uniformly distributed and where resources are tightly clustered. The dominant approach for the former is "bucket brigading", which separates robots in space to minimize mutual interference [25, 26, 27]. The dominant approach for the latter is ant-inspired trail-following, which directs robots to resource patches. Trail-following robots are separated in time rather than space, and *requires* that robots are able to navigate close together without significant mutual interference.

In [28], we extended the bucket brigading method introduced in [25]. Our adaptive bucket-brigade foraging algorithm (described in Section 2.6) allowed robots to vary the size of their work-site dynamically in response to interference with other robots. We showed that this could increase the performance and scalability of the foraging team.

Here we further extend adaptive bucket brigade foraging to allow robots to relocate their work zones to more productive areas. This allows the distribution of robots to reflect the current distribution of pucks, while still separating them in space to maintain resistance to interference.

This paper narrows the gap between previous work in bucket brigading, characterized by its assumption of high risk of spatial interference, and uniformly distributed resources, and ant-like foraging, with its assumptions of relatively very low spatial interference and tightly clustered resources. We discuss some ways that clusters may form spontaneously in a foraging system, further motivating cluster-friendly methods.

We describe and examine our novel adaptive forager in a series of simulation experiments.

2.3 Related work

Foraging strategies that seek to reduce spatial interference have been studied extensively. Liu *et al.*, in [26], produced an adaptive controller to dynamically decide between "resting" and "foraging" operations, in order to maximize energy income. Here interference was controlled by adapting the size of the foraging population.

Fontán and Matarić [27] investigated foraging with small teams (2, 3, and 4 robots) of real robots which forage in separate territories to reduce spatial interference. Territories were

assigned *a priori* and covered all available space without overlap. In contrast, in our work, work areas (the analogue of territories) are assigned dynamically and adaptively.

Shell and Matarić [25] pioneered the bucket-brigade approach to foraging in large-scale multi-robot systems. They discussed interference in multi-robot systems. Instead of allowing all foragers to explore the entire environment, they restricted each forager to straying no further from its starting location than a universal, preset value: the search radius. That study found a relationship between the number of workers and the performance of the team as a function of search radius. In the paper, we use a similar algorithm.

Bucket-brigade foraging had previously been explored, and its efficacy demonstrated in real robots [29, 30]. Bucket-brigade foraging in ants has also been studied (and explicitly named) in [21]. Gordon *et al.*studied spatial division of labor in red harvester ants [31].

Notably, Schmickl and Crailsheim, in [32], simulated robots acquiring food from a localized source *and* investigated the possibility of *positive* effects of spatial interference. Their robots communicated using trophallaxis (transferring food from one robot to the next directly) and avoided non-trophallactic collisions with one another. The use of trophallactic contact induced a gradient of food concentrations, and robots could measure the local gradient and use the information to to navigate uphill. However, their research did not explicitly look at the effect that using a localized source had on the performance of their algorithm.They did not address alternate distributions of the "dirt", such as the uniform distribution.

Other authors are closing the natural/artificial gap between ants and robots. Kube and Bonabeau, in [33], modeled a group of ants cooperatively carrying a single food item with real robots in a cooperative box-pushing task. They produced a coordinate movement effect without direct communication between robots.

Bucket-brigade foraging has been studied in insect societies as a form of task partitioning. Anderson et al., in [34], described a form of bucket-brigading seen in several species of ant and termite in which insects pass resources directly from one to the next until it reaches the nest. They gave situations in which bucket-brigading would be especially effective, such as situations in which material must be passed along a narrow passageway, or when many insects foraging around the same source create a bottleneck.

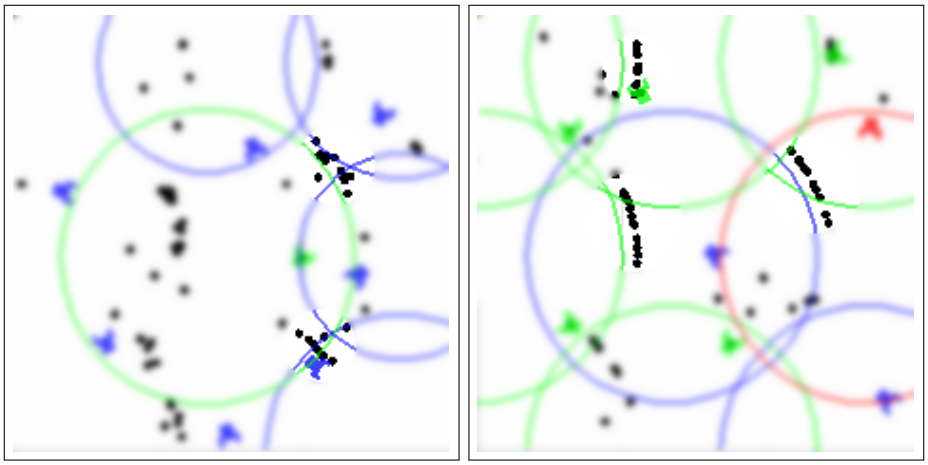
Trail-following behavior in robotics research is well-studied. Vaughan et al. in [35] implemented trail-following in robots in a transportation task (similar to foraging) using low-bandwidth wireless communication. Real-world implementation of ant-like trail following shows that mutual spatial interference is a severe limitation. Later work by the same authors addressed this problem directly by quickly resolving conflicts over navigation space [36].

2.4 Simulated robotic foraging

Our experiments were performed in a dedicated multi-robot foraging simulator. The environment is a square region with a home location in the northeast corner. Space is approximately continuous, and there are no obstacles in the environment apart from the robots themselves; robots avoid collisions with the environment's boundaries and with one another. Pucks are distributed throughout the environment, either uniformly or in a cluster depending on the experiment; pucks are modeled as points and do not "interfere" with one another. Robots are initially placed at random intervals throughout the environment.

Robots are equipped with 12 short-range proximity sensors capable of detecting walls and other robots up to 1m away; the sensors can differentiate between walls and robots. Each robot has a gripper which can lift, carry and drop a single puck. Another sensor detects when a puck is under the gripper and therefore grippable. This is the only way the robots can sense pucks. The robots measure 15cm $\times$ 15cm, approximately 1/28,000 the area of the world.

Except for the ability to place clusters of pucks (see below), this is the same experimental setup as used in [28].



(a) (b)

Figure 2.2: Clusters forming spontaneously (a) in neglected regions and (b) near the boundaries of foragers' work areas. The large circles indicate the boundaries of robots' work areas. The areas outside the clusters are blurred to highlight the clusters.

2.5 Clustering

Much of the previous work with foraging in robotic swarms assumes that the distribution of pucks is initially uniform. However, in real-world foraging scenarios, it is possible that the resources to be collected are distributed non-uniformly, especially in patches or clusters. In this work, we examine scenarios where there is initially a single cluster of pucks. Clusters are of interest for two reasons, first because some resources naturally or inherently are produced or deposited in clusters (apples on a tree; a bucket of tennis balls; a pile of rocks), and secondly because clusters may emerge as a side-effect of the actions of the robots and the peculiarities of their controllers.

The controller developed in [25] allowed robots to keep track of the distance and bearing to their starting location, through odometry. The inevitable error and unbounded drift in odometry caused the robots' work areas to drift over time. The authors noted that this had the added effect of ensuring coverage of otherwise neglected areas of the environment.

In our original experiments with global, non-drifting localization, we noticed that some areas of the environment were indeed neglected (see Figure 2.2a), in that no robot's work area covered them. Clusters of pucks spontaneously formed in these neglected areas. Even when total coverage is achieved, if some areas are more frequently visited for drop-off than pick-up then a clustering effect will exist.

As an example of this effect, we observed that clusters of pucks tended to spontaneously form near the edges of robots' work zones, visible in the simulator screenshot depicted in Figure 2.2(b).

The existence of non-uniform resource distributions, typically clusters caused either by the asymmetry inherent in the initial supply of resources, or by asymmetrical pick-up and drop-off frequency, motivated us to study adaptive approaches to foraging that work well under these conditions.

In our experiments, we looked at two distributions of pucks:

- the "hat" distribution, in which pucks are uniformly distributed, but only appear within a fixed circular area (the "cluster") of radius $\frac{1}{c} \times 25$m, and

- the uniform distribution (effectively $c = 0$).

Here c is the "clustering parameter", which indicates how tightly clustered the pucks are.

2.6 Foraging in a bucket-brigade

In the bucket brigade foraging controller of [25], each robot stays within the same fixed radius of its work-space location. This restriction means that robots approximately maintain their initial uniform distribution, keeping the robots spread out and limiting interference. Our adaptive method allows the robot's distribution to change over time in response to the environment. We describe the original method and our extensions in this section.

2.6.1 Basic bucket-brigading algorithm

Robots can be in one of three states: searching, homing, or returning. The initial state is searching.

- A robot in the searching state searches its "work area", the circle defined by the robot's initial location and a fixed work-radius. The search method is implementation-specific; the details of how a robot goes about locating a puck are not relevant to the application of the bucket-brigade strategy. In our implementation, robots followed a random walk except when avoiding obstacles and steering to stay within the work area.

  On detecting a puck, the robot picks it up and transitions from the searching state to the homing state.

- A robot in the homing state drives in the direction of the global home location (supplied to the robot) while avoiding collisions with other robots.

  On reaching the home zone, or on leaving its work area, the robot drops its puck and transitions to the returning state.

- A robot in the returning state, drives towards the center of its work zone. On arriving within *half* their search radius from this location, the robot returns to the searching state.

In any state S, if a robot must avoid an obstacle it will transition to an avoiding state until obstacle avoidance is no longer necessary, at which point it will return to state S, as per a subsumption architecture [37].

The result of this strategy is that very few robots actually deliver pucks to the home zone. Assuming robots are uniformly distributed in space, and assuming a sufficiently large population, one or more robots will have a work area that overlaps the home zone. These robots will transport pucks just outside the home zone into the home zone. Other robots will have work areas that overlap these robots' work areas, and this pattern of overlap will continue so that most of the environment is covered by a work area that is "connected" to the home zone, and pucks will travel toward the home zone in a manner suggested by the algorithm's name—a bucket-brigade.

2.6.2 Adding adaptive work-zone size

Our first extension is to allow each robot to adapt its work-zone radius with experience. While the robot is avoiding collisions with other workers, it decreases the radius at a preset rate. Otherwise, the radius increases at a (possibly different) preset rate. Allowing different robots to use different work area radii gives the team more flexibility in adapting to the distribution of workers in the environment. We have previously shown that this simple extension improves performance and allows larger population sizes to work efficiently in uniform puck distributions [28].

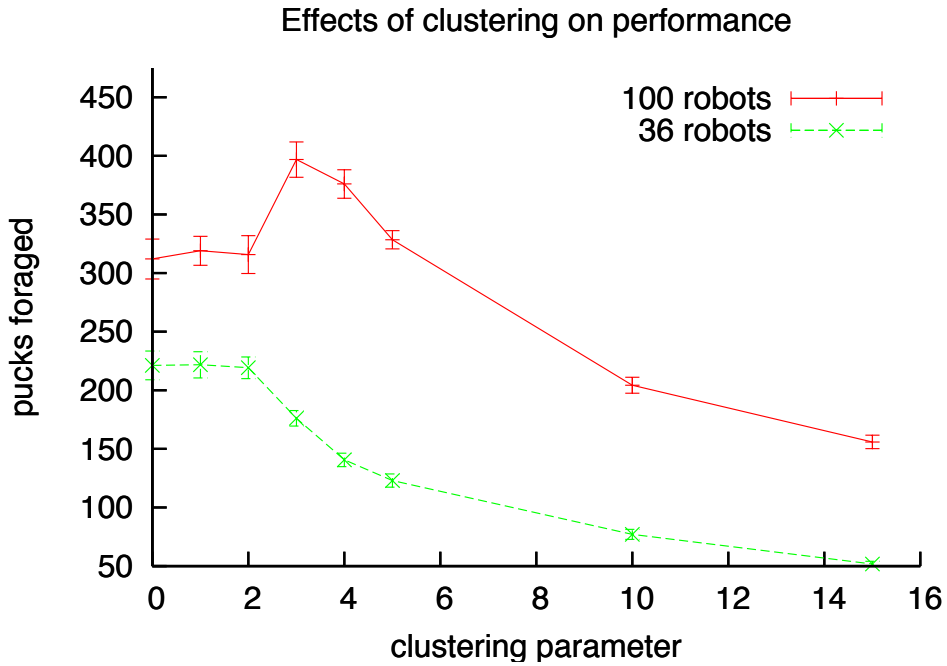


Figure 2.3: Performance of the standard, non-adaptive bucket-brigade algorithm degrades significantly as the degree of clustering increases. As the "clustering parameter" increases, the cluster gets smaller but has the same number of pucks in it. Mean performance over thirty trials are plotted with 95% confidence interval.

2.6.3 Effects of clustering on the bucket-brigade controller

Given the controller described above, we found that if the initial placement of pucks is clustered instead of uniformly distributed, then performance decreases. Using the simulation environment shown in Figure 2.1 we ran 30 trials for a range of cluster parameters c, and for two robot populations. Figure 2.3 shows the results. The more tightly clustered the pucks (larger c), the fewer pucks are delivered over the length of the experiment. The success of the non-adaptive bucket brigade method depends on exploiting the uniform distribution of pucks.

2.6.4 Where to start searching

Recall that the bucket-brigade foraging algorithms require each robot to maintain an approximation of its start location. They begin their puck search near this point each time. When puck density is uniform, one location is as good as another for puck searching, and the start location serves only to spread robots out. In non-uniform puck distributions, however, placing the center of the work zone in a puck-rich neighborhood is likely to reduce search time and improve performance. Our second adaptive modification attempts to improve the position of the work zone center over time.

As in Shell and Matarić's original method [25], our robots' estimates of their work areas drift on a slow random walk. We improve on this by adding non-random, deliberate work-zone *relocation* as follows.

Upon finding a puck, a robot will relocate its estimate of its starting location a certain portion of the way along the line from its current position to the location of the puck. This proportion is an adjustable *relocation parameter* ρ; a value of $\rho = 0$ (no relocation) corresponds to non-relocating bucket-brigade foraging, in which the robot's estimate of its starting location

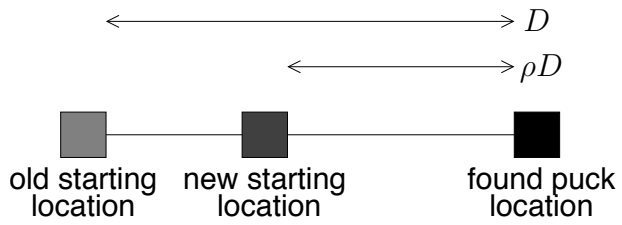


Figure 2.4: If D is the distance between the old starting location and the newly found puck, then ρD is the new distance; ρ (for $0 \leq \rho \leq 1$) is the relocation parameter.

remains fixed (upon finding pucks; it is still subject to drift). A value of $\rho = 1$ (complete relocation) indicates that robots will return to the last place they found a puck before they begin searching again. All robots use the same value of ρ (though there is no a priori reason requiring this). This process is illustrated in Figure 2.4 and described algorithmically in Algorithm 1.

Require: $0 \leq \rho \leq 1$ is the *relocation parameter*
Require: $r_Z \geq 0$ is the *range to zone center* from odometry
  if I found a puck **then**
    Pick up the puck.
    $r_Z \leftarrow (1 - \rho) r_Z$
    Switch to the homing state.
  end if

Algorithm 1: Procedure for robots in the searching state.

This tends to cause robot work zones to move towards areas of high puck density, reducing search time and potentially improving performance.

2.7 Experiments

2.7.1 Simulated environment

We performed a series of simulation experiments to examine the new method. Our hypothesis is that relocation can improve performance by adapting to the current puck distribution.

The simulator was a modified version of the one used in [28]. The foraging environment was a 25m × 25m enclosure with a quarter-circular home area, of radius 1m, in the northeast corner. Robots were initially positioned at random locations (different in each trial).

Pucks are initially placed at random locations throughout the environment (different in each trial), either in a uniform distribution, or according to the clustering model described in Section 2.5. In either case, on average there were two pucks per square meter (for a total of 1,250 pucks available to be foraged). Upon delivery to the home zone, a puck is replaced at random according to the same initial distribution.

2.7.2 Parameter space

Experiments were run with all possible combinations of the following parameter values:

- Clustering parameter: we tested several small values ($c = 1, \ldots, 5$), and a few larger values ($c = 10$, $c = 20$).
- Robot population: could be either a "small swarm" of 36 (0.25 robots per square meter) or a "large swarm" of 100 (0.69 robots per square meter).

#robots	Clustering	ρ	Performance		x p
			μ	σ	
36	0	0	221.000	33.837	< 0.0001
36	0	1	130.300	19.701	
36	5	0	120.633	15.940	< 0.0001
36	5	1	179.800	14.372	
36	10	0	76.900	11.973	< 0.0001
36	10	1	146.133	12.260	
36	15	0	56.100	9.386	< 0.0001
36	15	1	124.367	17.604	
100	0	0	313.833	36.545	0.0022
100	0	1	347.433	53.187	
100	5	0	335.667	36.456	0.0002
100	5	1	367.500	33.470	
100	10	0	208.033	17.629	< 0.0001
100	10	1	328.067	19.311	
100	15	0	150.033	15.751	< 0.0001
100	15	1	282.833	24.552	

Table 2.1: Analysis of performance for extreme values of the relocation parameter ρ. p-values for the significance of the difference between the $\rho = 0$ and $\rho = 1$ cases' performances are shown (mean over thirty trials).

- Relocation parameter: could be 0.00, 0.25, 0.50, 0.75, or 1.00.

2.7.3 Performance metric

We make two claims; first, that clustering of pucks hurts the performance of existing algorithms, such as bucket-brigading, which ignore the distribution of the pucks. This is supported by the data discussed in Section 2.6.3 and displayed in Figure 2.3. The second claim is that we can adapt the foraging algorithm to work well in the presence of non-uniformity by using the relocation system described in Section 2.6.4 and illustrated in Figure 2.4. To support the second claim, we compare the performance of foragers in clustered-resource situations with and without relocation.

In each experiment, robots foraged for one hour of simulated time. At the end of the hour, the total number of pucks delivered to the home zone was counted.

Performance is reported as the average number of pucks collected over thirty trials for each possible choice of parameters as given in Section 2.7.2. A 95% confidence interval was determined, assuming that real performance is normally distributed.

2.8 Results and discussion

We found that relocation can have either a beneficial or an adverse effect, depending on the setting. The results support our hypothesis that relocation would allow robots to adapt to a tightly clustered distribution of pucks: swarms employing complete relocation outperformed those that did not relocate their starting locations, nearly doubling performance in the small-swarm (36 robots) scenario. These results are statistically significant; see Table 2.1.

The results are illustrated graphically in Figure 9.3.

We also found that relocation did indeed permit the distribution of robots' starting locations to adapt to the puck distribution. See Figure 2.6 for an illustration.

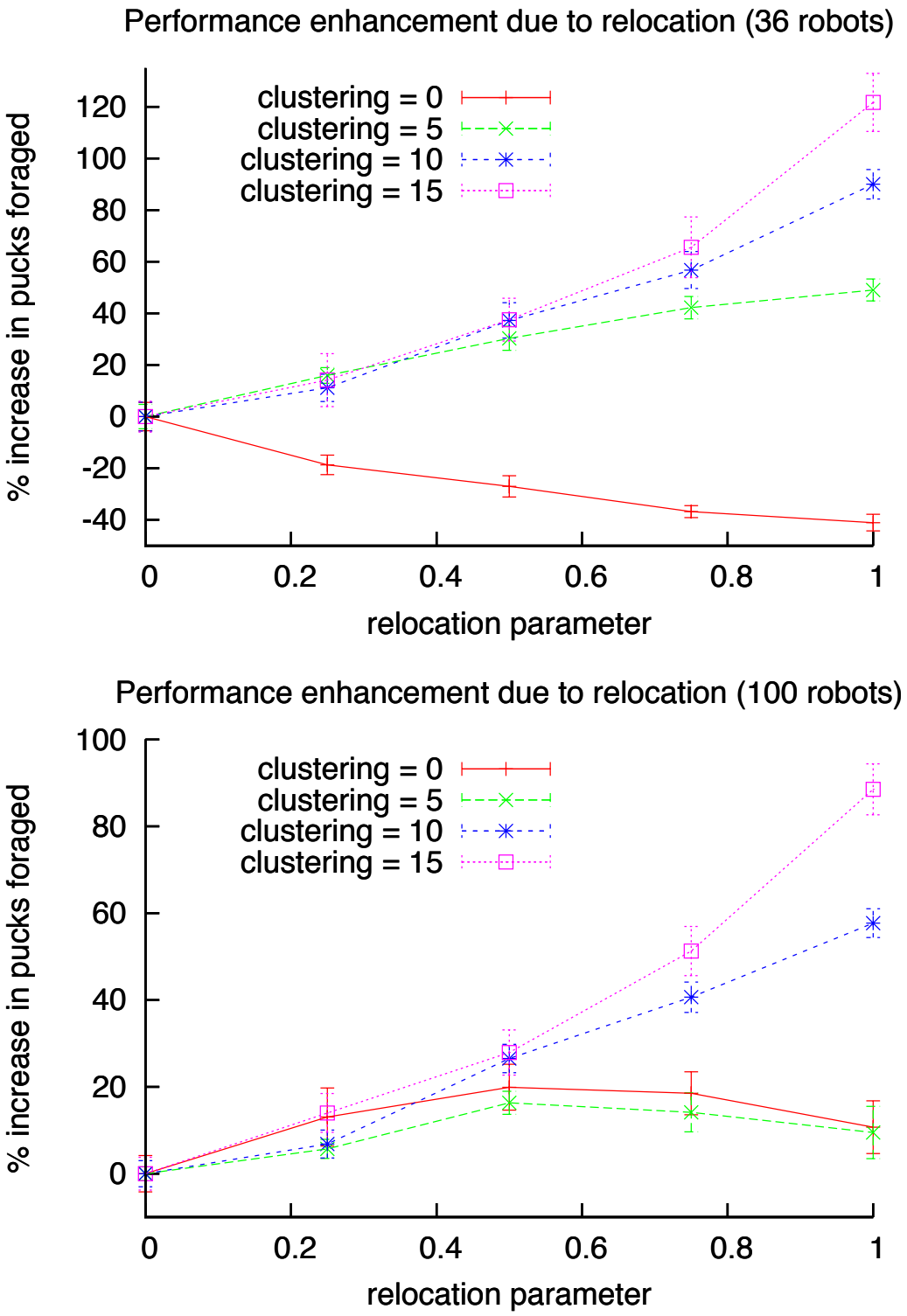
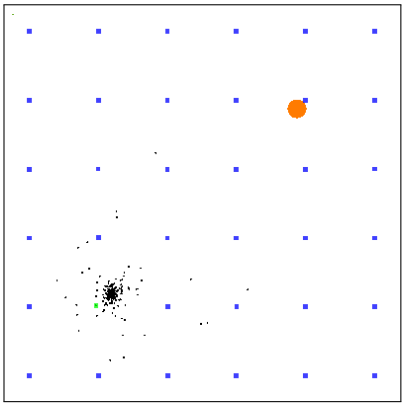
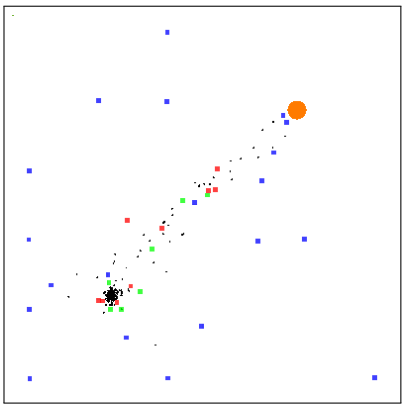


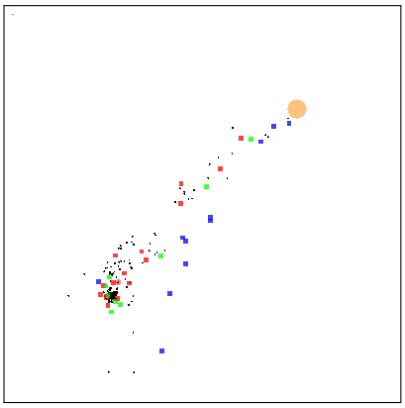
Figure 2.5: Performance of relocating compared to non-relocating foragers, averaged over thirty trials, with 95% confidence interval, over a range of relocation parameters. In tightly clustered puck distributions, the graph shows a upward trend in performance as the relocation parameter is increased.



(a) Initial



(b) 20 minutes



(c) 40 minutes

Figure 2.6: Distribution of pucks (dark circles) and robots' starting locations (squares) initially and after 20 and 40 minutes. As pucks are spread out via bucket-brigading, the foragers automatically relocate to the "trail" of pucks from the cluster to the home zone. For the clarity of this illustration, robots were initially placed on a grid.

2.9 Future work

In focusing on adaptive work site relocation, we discovered a wealth of complexity arising out of a non-uniform resource distribution. In [28] we provided a simple, one-dimensional mathematical model of such distributions that form as a result of the action of robots using the bucket-brigade algorithm for transporting pucks. We would like to develop a more general model that includes the variety of sources of clustering. We would also like such a model to incorporate the energy requirements of the workers, as discussed in [26].

Furthermore, we would expect that relocation of work sites would arrange the robots approximately in an *ideal free distribution* as described by Fretwell and Lucas in [38]. That is, in an environment with multiple clusters, the robots will distribute their starting locations so that the number of robots working at each cluster is proportional to the number of pucks in the cluster. Further investigation is needed to confirm this hypothesis.

The problem of spatial interference motivated the bucket-brigade foraging algorithm as a means to *spatially separate* the workers[25]; temporal separation has been studied in both ecology[39] and robots. We would pursue a descriptive (or even better–generative) model that can describe or produce spatial and temporal foraging in a unified framework.

Wawerla and Vaughan modeled the time-value of labor in [40]. Fundamentally, all robotics enterprises must be judged in the context of a real-world economy; future work in foraging swarms should take this dimension into account. This would provide greater weight to pucks foraged early on, which highlights the value of early recruitment of workers into active foraging.

Also, since we have shown that relocation can be beneficial or detrimental to performance in different environments, we would like, as we did in [28], to develop an adaptive mechanism for selection of the relocation parameter.

Chapter 3

SO-LOST: An ant-trail algorithm for multi-robot navigation with active interference reduction

Seyed Abbas Sadat, Richard T. Vaughan [41]

Abstract

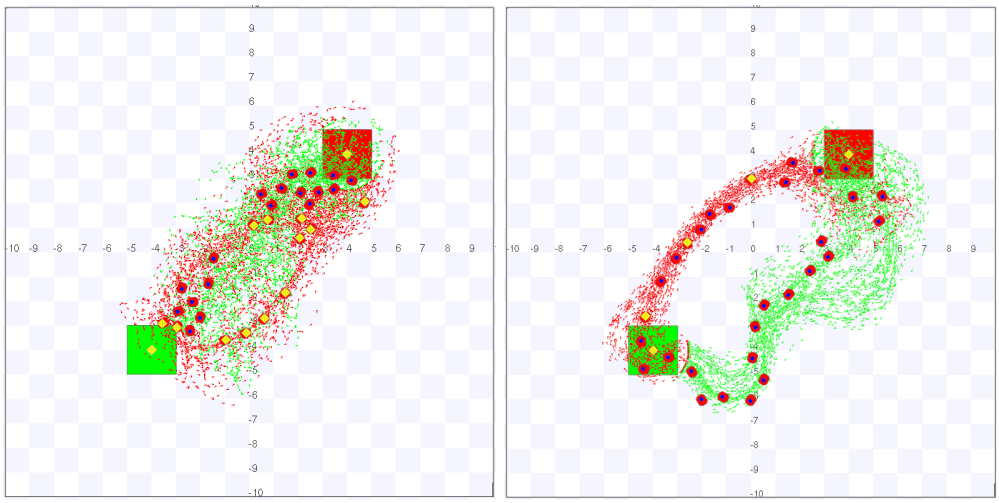
We consider a group of autonomous robots which perform the classical task of transporting resources from a source to home. The robots use ant-like emergent trail following to navigate between home and source. When trails lie close together, spatial interference between robots navigating in opposite directions reduces overall system performance. This paper proposes a navigation strategy which is effective in separating trails with different goals. The results of simulation experiments indicate that the performance of robots is usefully increased compared to original algorithm in constrained environments.

3.1 Introduction

This paper presents a navigation strategy to reduce interference in ant-inspired foraging-and-trail-following robot systems. We consider the classical *resource transportation* task, in which a team of robots works to transport resources in an initially unmapped environment. Robots start from a home position and search for a supply of resources. On reaching the source, they receive a unit of resource and must return home with it, then return to fetch more resource repeatedly for the length of a trial. Achieving this task reliably with robots will meet a real-world need. It is a canonical multi-robot task since the work is inherently parallelizable. The critical factor limiting scalability is mutual spatial interference between robots.

Our earlier work [42, 43] examined an implementation of ant-inspired trail following that is suitable for imperfectly-localized mobile robots. In our "localization-space trails" (LOST) algorithm, robots generate and share trail data structures composed of waypoints specified by reference to task-level features that are shared by all robots. The trails are continuously refined online, and maintain the ant-algorithm property [44] of converging to near-optimal paths from source to home.

Trails are labelled with their destination, and the trail to the current goal destination is followed. In previous work, the other trails were ignored during navigation. However, trails may overlap in space and robots navigating to different goals may interfere with each other's progress. We argued previously that an emergent property of LOST is that it can produce trails that are



(a) LOST (b) SO-LOST

Figure 3.1: Trails formed in an obstacle-free "empty" environment using LOST and SO-LOST. SO-LOST has separated the trails, achieving better throughput due to reduced interference.

separated in space [42] thus reducing interference. In this paper we describe a modification to LOST called Spread-Out LOST (SO-LOST) that greatly improves this effect, creating trails that are share parts of the environment while being far enough apart to reduce interference. The result is superior performance in most of the cases we examine. The innovation is that the robots' trail-following behaviour is subtly modified to avoid competing trails, with the emergent effect that trails are iteratively spread out until intereference is largely avoided.

We do not know of any biological system that uses similar behaviour to tackle the spatial interference problem and the new navigation algorithm in this paper is a synthetic technique that improves the efficiency of a biologically-inspired path finding and sharing algorithm used in multi-robot systems. The advantage of these type of synthetic behaviors has been studied before (e.g. [45]).

3.2 Related Work

Various different robot implementations of ant-like trail following have been presented. Real chemical marks were first used to produce true stigmergic trail-following in [46]. Also recently, Fujisawa et al. [47] carried out a study out of communication in a swarm of robots using pheromone and proposed a behavior algorithm for robots to search for prey and attract other robots. They used ethanol as pheromone in their real robot experiments. The challenge of chemical and sensor engineering makes these methods often impractical. A more parsimonious method was invented by [48] where virtual pheromone trails are implemented by directional infra-red messages transmitted from robot to robot. Robots echo received messages, incrementing a contained hop-count which is used to estimate the distance to the message source. In both chemical and IR-mediated methods, the local "gradient" is sensed directly from the environment. If robots are mutually localized, virtual trails can be created from global waypoints, which are communicated by wireless network. We showed that this scheme can be robust to large zero-mean localization error ([42]), and admits a relaxed and practical definition of mutual localization ([43]).

The diminishing-to-negative-returns effect of increasing the number of robots on performance has been studied in related contexts. In a mathematical model of robot foraging [49], it was shown that adding more robots to the system improved the group performance while decreasing individual robot's performance. Based on that model, an optimal group size was found that maximizes the group performance. Explicit anti-interference strategies are studied in real robots in [50], to increase performance in the transportation task. Congestion control in a dense multi-robot system is studied in [51], where asymmetries that resolve conflicts are introduced by modifying either the environment or the robot controllers.

A related idea using occupancy grids to model multi-robot interaction is described in [52]. There, a global histogram of occupancy is constructed, and areas with high probability of co-location are identified and fed into an (unrelated) interference reduction method.

3.3 Localization-Space Trails (LOST) review

This section briefly reviews the generalized trail-following method formulated in [43].

LOST generates trails between the locations of *Events*. An Event is defined as a task-relevant occurrence that may happen to any member of the team, and is locally but reliably perceived. For example, in our transportation task the relevant Events would be 'pick-up-resource' and 'drop-resource'. A robot must be able to recognize these events in order to switch between resource-seeking behavior and home-seeking behavior. When an Event occurs to a robot, its current pose in localization space is recorded to create an [Event,Pose] tuple called a *Place*. A robot can then express information about the world relative to the Places it has seen. Other robots that have position estimates for the same Events can interpret the coordinates in their own local frame of reference. Thus robots are mutually localized by the shared experience of the common task, rather than conventional global localization in some arbitrary coordinate system.

The purpose of LOST is to guide the robot to a Place currently of interest: the goal. The algorithm provides the robot controller with two pieces of information; (i) the *heading-hint* that is the local direction in which to travel to reach the goal; (ii) the *distance-hint* that is the estimated cost (usually in time) to reach the goal. These hints are extracted by examining a set of waypoints called *Crumbs* which are poses specified relative to a Place. The current set of Crumbs specified relative to a particular Place is a *Trail* to that place. A Crumb is a tuple $C = [P_c, L_c, d_c, t_c]$ containing the name of the Place P_c to which it refers, a localization space pose L_c, an estimate d_c of the distance (in some distance function) from L_c to P_c, and the time t_c when the Crumb was created.

Each robot maintains an initially empty temporary trail. Every S seconds, a robot inserts a new crumb to the temporary trail. The crumb contains the current location of the robot, the name of the most recent Event experienced by that robot, the distance from the last event, and the current time. When another event occurs to the robot (e.g., when a robot drops off its cargo), the temporary trail is broadcast to all robots, including itself, then deleted. A new temporary trail is then created for the recent Event.

Besides the temporary trail, each robot maintains a trail for each different Event it has learned about from the network. When a broadcast trail is received, the crumb poses are transformed into the local frame of reference by the rigid body transform defined by comparing the local and received poses of the trail's Place. The transformed crumbs are added to the local trail for this Place. All trails are periodically scanned and any Crumb with time stamp older than age threshold a seconds is discarded. Thus the trail is updated dynamically, and out-of-date information is expired. The dynamic response of the trail to changing environments is a function of a.

Suppose a robot at pose L_r has Place P_g as its goal, such as $Event(P_g)$ = 'drop-resource-at-home'. The robot searches the set of Crumbs with Place = P_g to find the set of crumbs that lie within its *field of view* (FOV) i.e., within radius d_f of L_r. From this set it finds the crumb

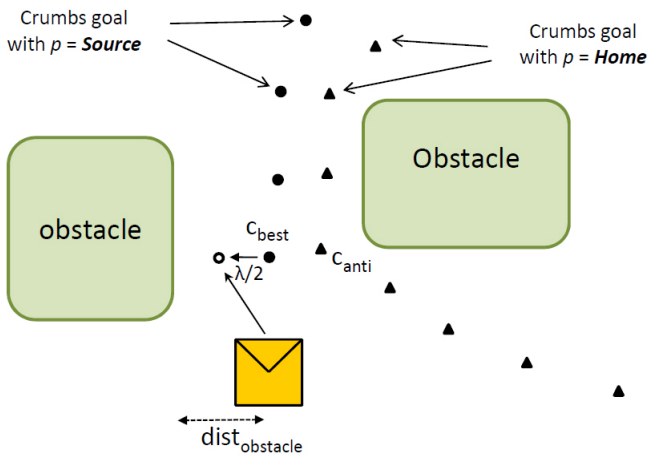


Figure 3.2: Sketch of the new LOST algorithm. While the robot was following the trail (filled circles), it sees a crumb with different goal (triangle) and thus changes its direction to a new point (the empty circle).

C_L with the smallest distance-to-goal d_c. This distance is returned as the distance-hint. The heading-hint is the angle from the robot's pose L_r to $L_c = Pose(C_L)$. If the robot moves in the direction of the heading hint and repeats this process, it will encounter crumbs with decreasing distance to goal values, and eventually arrive at P_g.

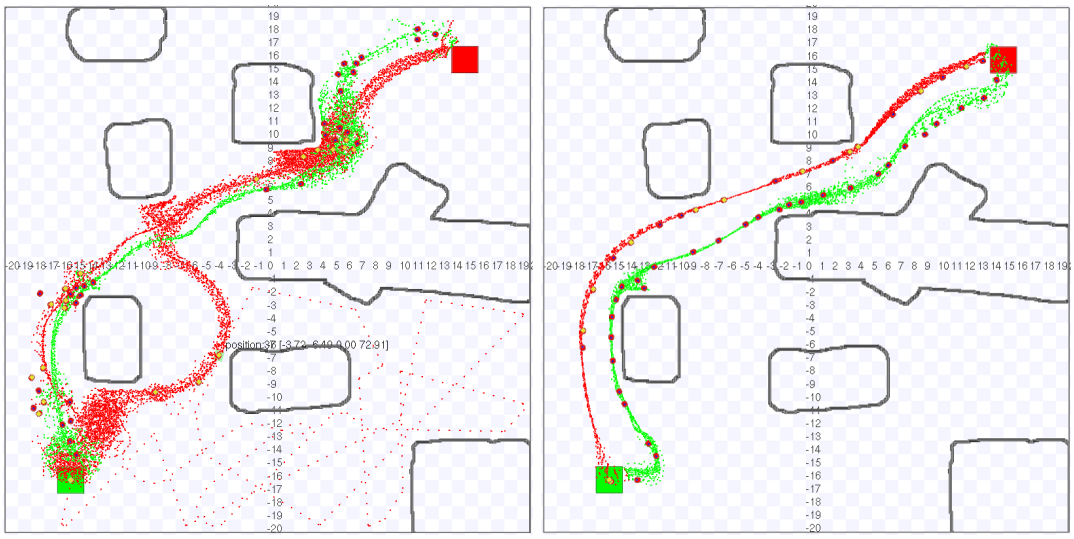
The robot will take the shortest route so far discovered from that location. By following the Crumbs dropped by the whole population, each robot benefits from the others' exploration; robots will find a reasonable route much more quickly than they would alone. The larger the population size, the greater the probability of finding a good route and the more quickly a good route is found.

3.4 Spread-Out LOST

In the LOST algorithm, as the robots move they "lay" crumbs. The goal Place of these crumbs is the place that the robot has most recently visited. This means that in order to reinforce a trail, the robots should travel in the opposite direction that the crumbs are showing and consequently robots following a trail are very likely to interfere with robots laying (reinforcing) it. With few robots, this does not have much effect on performance and the "pick-up-source" and "drop-resource" trails converge to one shortest discovered path. However as the robots' team size increases, these interferences damage the performance of the system.

To address this, we modify the LOST algorithm so that when a crumb is created, the P_c data field will be the goal of the robot rather than the recently visited place. With this modification, the robots have to perform two searches at the beginning; one for finding a path from home to source and another one for a path from source to home. We can avoid the need for the second search by copying the first discovered trail and changing the goal and reversing the distance hint along the trail.

When the environment in which the robots are working is complicated and contains narrow corridors and doorways, or is very crowded, LOST may produce trails with different goals that are either very similar or have many parts in common. Figures 3.1(a),3.3(a),3.4(a) show this phenomenon in our trail-following robot system implemented in the well known simulator Stage ([3]). The trails formed between source and home are often very close to each other, leading



(a) LOST (b) SO-LOST

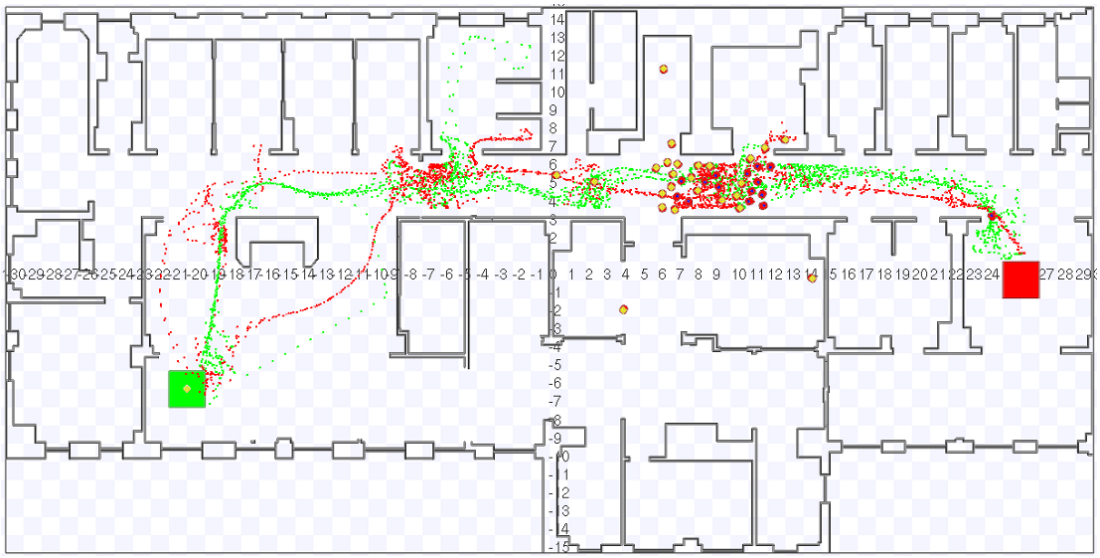
Figure 3.3: Trails formed in the cave environment using the LOST and the new algorithm after 30 mins of simulation.

to problematic interference between robots travelling in opposite directions. Since the crumb trail data structure does not contain any explicit information about the fixed obstacles in the environment, there is no way to directly process the trail data to avoid robot-robot interference without risking directing robots into fixed obstacles. Instead, we use a small modification to the robots' trail following control strategy that results in emergent trail separation.

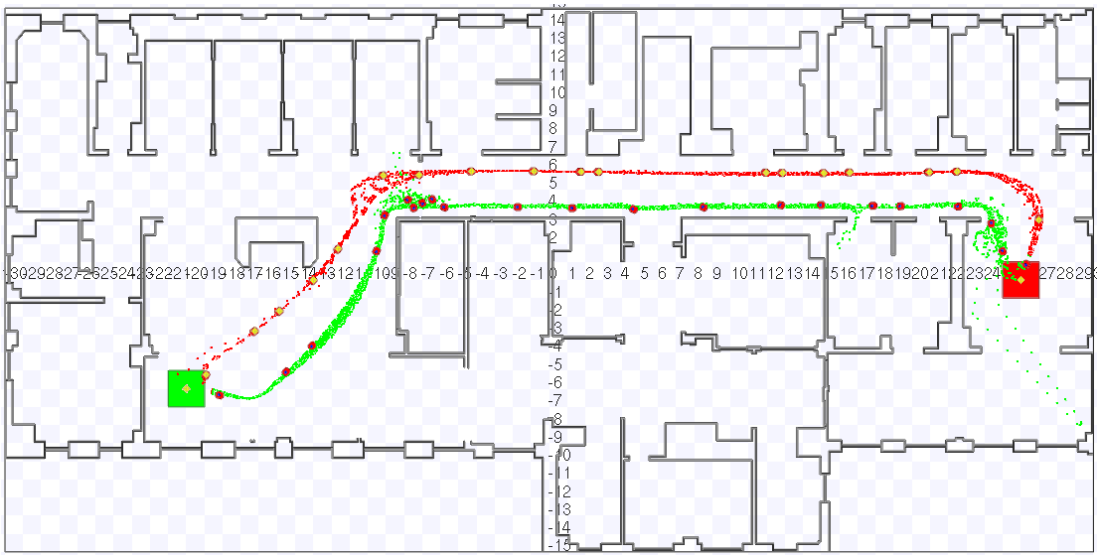
A robot following a trail to get to P_c, can interpret crumbs with goals other than P_c, as proxies for potentially interfering robots. If the robot follows the trail to P_c while slightly avoiding all other nearby crumbs, the new P_c crumbs it lays will tend to be slightly more distant from other crumbs than those just followed. This mechanism is essentially similar to the iterated corner-cutting that drives the ant-algorithm's ability to locally improve trail length. The resulting trails may be slightly longer but may reduce interference significantly, as suggested by the results below.

The new trail-using algorithm is presented in Algorithm 2. It first searches for the crumb c_{best} with minimum distance to goal that is located in the robot's FOV. Then if there exists a crumb c_{anti} with different goal than the robot's goal and it was closer to the robot than a distance threshold (*crumb_avoid*), the direction to which the robot moves will turn to the robot's left. This will change the angular velocity of the robot so that it keeps away from c_{anti}. The shift vector is orthogonal to the $\overrightarrow{(robot, c_{best})}$ vector. Also, the magnitude λ is calculated based on the obstacles near the robot such that the robot's target point does not lie inside an obstacle. Trails with different goals are necessarily very close to each other around source and home. Thus the shift vector is not applied when the robot is near the goals to prevent robot's circular trajectory in these areas.

Figure 3.2 illustrates how the behavior of the robot changes in presence of c_{anti}. The robot is following the small circles. On seeing the triangle crumbs, the robot's target point is changed from c_{best} to another point (the empty circle). This simple mechanism alters the robots movement so that different trails are gradually separated from each other. The divergent movement of trails continues until they are away enough from each other, if possible.



(a) LOST



(b) SO-LOST

Figure 3.4: Trails formed in the hospital environment. using the LOST and the new algorithm.

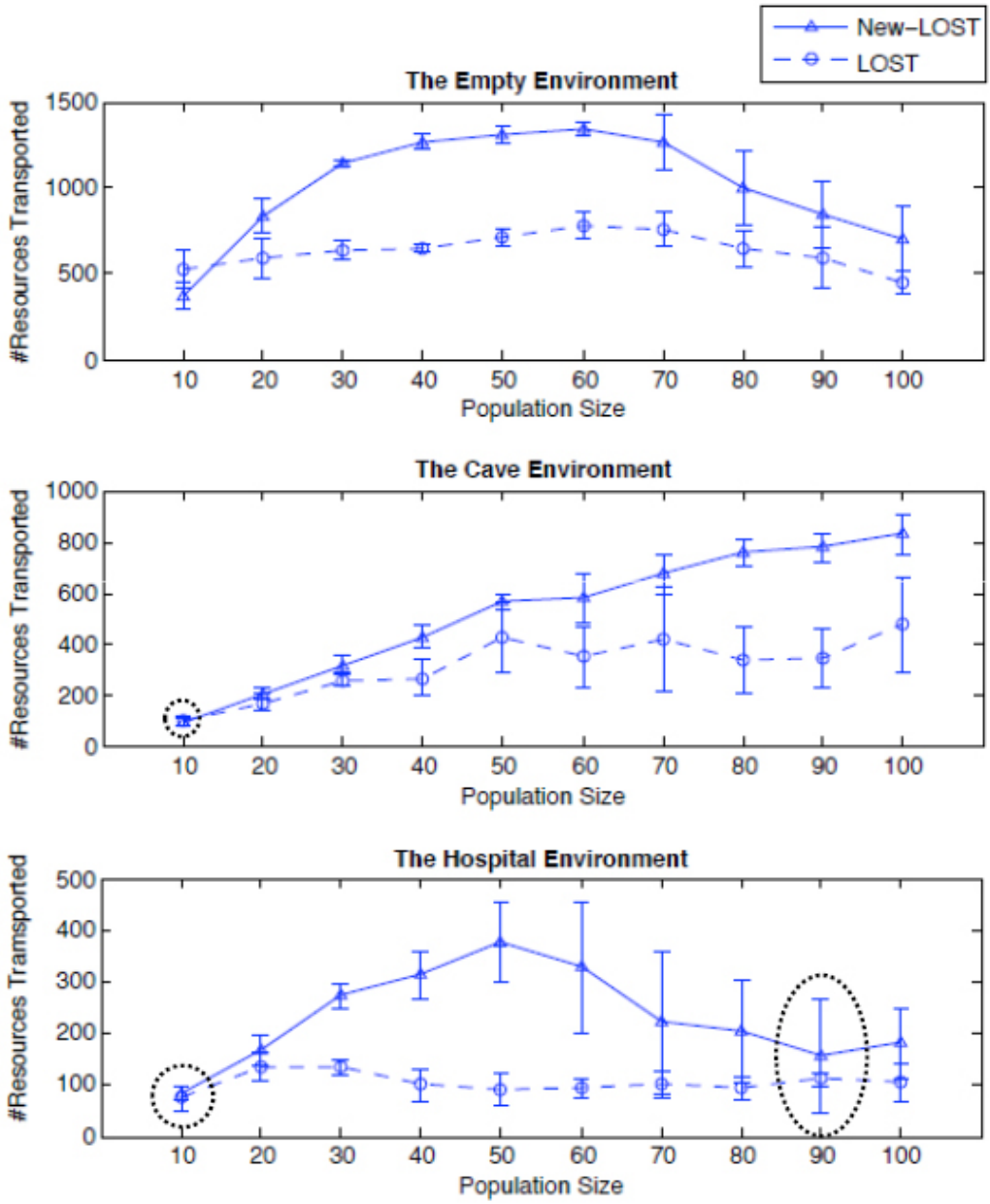


Figure 3.5: The result of the experiments in the 3 environments. The mean performance over 10 trials are shown with errorbars showing the standard deviation for both the original LOST and the new algorithm. The dotted line shows the data point for which the two algorithm do not show significant difference in distribution.

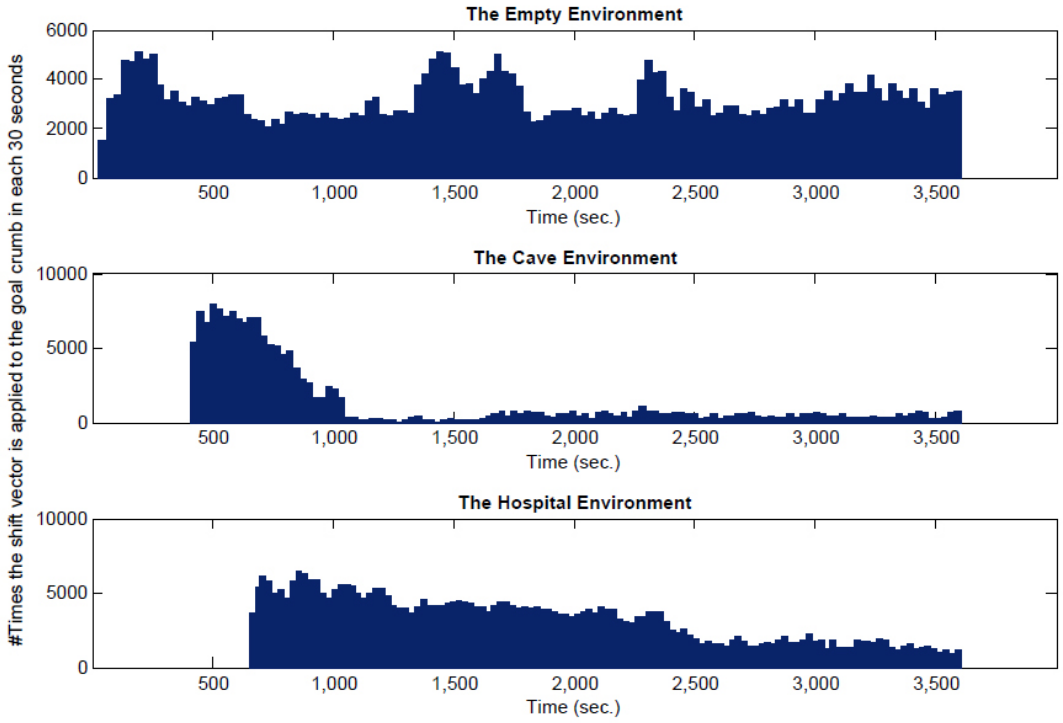


Figure 3.6: The histograms showing the number of times the goal crumb was shifted. The time bin is 30 seconds of simulation time. 30 robots are used in the empty environment and 50 robots are used in the other environments.

Require: The distance $dist_{obstacle}$ from the robot to the nearest non-robot obstacle on the left side of the robot.
return the direction Dir_{robot} to which the robot should move

Θ = all the crumbs in the robot's FOV with positions relative to the robot;
$\Sigma = \{c | c \in \Theta \wedge (c.p_c = robot.goal)\}$;
$\Pi = \{c | c \in \Theta \wedge (c.p_c \neq robot.goal)\}$;

$\lambda = Min(crumb\_avoid, dist_{obstacle})$;

$c_{best} = c$ s.t. $(c \in \Sigma) \wedge (\nexists c' \in \Sigma$ s.t. $c.d_c > c'.d_c)$;

if $(\exists c_{anti} \in \Pi$ s.t. $dist(c_{anti}, robot) < crumb\_avoid) \wedge (c_{best}.d_c \leq 2s)$ **then**
  $Dir_{robot} = \overrightarrow{(robot, c)} + \frac{\lambda}{2} \times \overrightarrow{(-1, 0)}$;
else
  $Dir_{robot} = \overrightarrow{(robot, c)}$;

end if

Algorithm 2: The New Trail-Using Algorithm

3.5 Experiments

3.5.1 Simulation Setup

We ran Stage simulations to evaluate the new algorithm in three different environment settings: *empty* (Figure 3.1), *cave* (Figure 3.3) and *hospital* (Figure 3.4). The size of the *empty*, *cave* and *hospital* environments are 20x20m, 40x40m and 60x30m respectively, with robot length 0.45m. Robots are Stage's Pioneer 3DX and SICK LMS200 laser rangefinder models. The bottom left (green) square is the source; top right (red) square the sink of resources. In the screenshots, robots (red polygons) are shown with yellow diamonds to indicate they are carrying a unit of resource. Robots start every trial at the same randomly-chosen uniformly distributed positions, do not know the initial location of source and sink locations, and must find them by exploration at the start of the trial. Each trial runs for 60 minutes, and the total number of resources delivered at the end of the trial is our performance metric. 10 trials are performed for each population size. LOST is deterministic but the local obstacle avoidance and searching is stochastic (for robustness), hence the need for repeated trials. For all experiments the *crumb_avoid* parameter is set to 2m.

3.5.2 Results

The results of the experiments are summarized in Figure 3.5, showing the mean and standard deviation of performance over 10 repeated trials plotted for each population size. The plot shows a marked improvement in many cases (in some cases 3 times better) in performance with the new algorithm.

As expected, with few robots (20), there is not much difference in performance since the interference among robots is small. In the empty environment with population size of 10, the LOST outperforms the new algorithm. This is because the benefit of interference reduction can not outweigh the penalty of increase in the length of the trails. As the population size increases and the environment becomes more constrained, improvement in performance gets bigger. This can be seen in the plot showing the results of the experiments in the hospital environment; For the smallest populations, the two methods perform about the same; however, since the hospital environment contains corridors and doorways (Figure 3.4(b)), there is a degradation in the

LOST performance with more robots whereas the new algorithm improves the performance in some populations up to 3 times.

To verify that the performance results are significantly different for different algorithms, we performed hypothesis testing using a T-test. The P values for the hypothesis that the performance values for LOST and the new algorithm are from the same distribution are calculated. For all population sizes, the test suggests that the distributions are significantly different ($P << 0.02$), except for the pairs identified in Figure 3.5 with dotted line.

3.6 Discussion

The new algorithm is based on the idea that laying crumbs near other crumbs with different goals increases the probability of co-location among the robots performing different tasks. This is more clear in transportation task in which the trails for 'pick-up-resource' and 'drop-resource' tasks can be formed very close to each other. In the new algorithm robots follow the trails and also try to keep a distance from other crumbs and therefore new trails are laid at a safe distance from each other. Figures 3.1(b), 3.3(b), 3.4(b) show the trails formed with the new algorithm. It is visible that different trails are separated from each other and consequently robots do not approach the unattractive trails. The magnitude of the shift vector (*crumb_avoid*) determines the distance of the trails from each other and should be large enough to keep robots away from each other.

In order to see if the trails converge to a stable state we plotted the number of simulation cycles in which the shift vector was applied in each 30 *sec* of simulation time (Figure 3.6). In the *cave* and *hospital* environments, after the trails are formed they are gradually separated from each other due to the high use of shift vector. After some time, the trails come into a relatively stable state. The shift vector is still applied occasionally since the trails in some narrow parts of the environment (like doorways) are at their maximum distance from each other and can not go farther away. For the *empty* environment since the area is small and there is a short distance between source and sink, the robots tend to be pushed towards other trails which results in the high use of shift vector throughout the experiment.

We do not know of any biological system that uses a similar approach to reduce destructive effects of interference among individuals, but still we believe that these techniques can be used in systems inspired from animals and social insects to improve the efficiency of robots in performing a task.

3.7 Conclusion and Future Work

In this paper we presented SO-LOST, a new navigation strategy to reduce interference in ant-inspired foraging-and-trail-following robot systems. The method makes use of the different trails formed in the environment to prevent robots with different goals from getting in each other's way. It is quantitatively evaluated through simulation experiments and shown to be effective in relatively constrained environments. Qualitatively, the screenshots of simulation experiments show that distinct separate trails with different goals were formed while keeping a distance from each other hence reducing the interference.

In future work we will implement the new algorithm on real robots and run experiments to verify our findings in simulation. Also, we will investigate methods of congestion resolution in trail-following robot systems. The algorithm presented in this paper is used to avoid congestion and conflicts between robots. However, there is plenty of room for improvement in mutual robot-robot avoidance methods, and development here would have an impact in many multi-robot systems.

The LOST and SO-LOST framework allows us to add various kinds of meta-data to the crumb and trail data structures. Here we have allowed all nearby trails to influence the behaviour of a trail-follower. We expect that performance could be further improved by clever use

of other meta-data embedded into crumbs, perhaps by gathering some global statistics. This would be unusual in ant-inspired systems, and perhaps powerful.

For now, we believe SO-LOST may be the most real-world practical trail-following algorithm yet described, since it explicitly manages the spatial interference that plagues real-world robots in any number.

3.8 Code publication

All source code, scripts, etc. used to produce the results reported in this paper are available online:
URI: http://www.cs.sfu.ca/ sas21/personal/abbas_alifeXII.tar.gz
SHA1: 9ea88a34d9971dffce26bc511f6ad2f875b8e44d

Chapter 4

Blinkered LOST: Restricting Sensor Field of View Can Improve Scalability in Emergent Multi-Robot Trail Following

Seyed Abbas Sadat and Richard T. Vaughan [8]

4.1 Abstract

We consider the classical task of transporting resources from source to home by a group of autonomous robots. The robots use ant-like trail following to navigate between home and source. This paper studies the effect on global performance of changing the field of view of each robot's trail-following sensor. It is shown that, under certain conditions, a narrow field of view can improve system performance. We argue that the benefit is obtained by selectively degrading the individuals' trail-following ability so that more work space is exploited in parallel, thus decreasing mutual spatial interference. This "worse-is-better" idea may be applicable to other large-scale multi-robot systems.

4.2 Introduction

We consider the classical *resource transportation* task, in which a team of robots works to transport resources in an initially unmapped environment. Robots start from a home position and search for a supply of resources. On reaching the source, they receive a unit of resource and must return home with it, then return to fetch more resource repeatedly for the length of a trial. Achieving this task reliably with robots will meet a real-world need. It is a canonical multi-robot task since the work is inherently parallelizable. The critical factor limiting scalability is mutual spatial interference between robots.

Our earlier work [42, 43] examined an implementation of ant-inspired trail following that is suitable for imperfectly-localized mobile robots. In our "localization-space trails" (LOST) algorithm, robots generate and share trail data structures composed of waypoints specified by reference to task-level features that are shared by all robots. The trails are continuously refined online, and maintain the ant-algorithm property [44] of converging to near-optimal paths from source to home.

An attractive feature of LOST and other ant-algorithm methods is that it is simple and

natural to use travel-time as the distance function to be optimized, so that the system can discover paths that may be longer in space but shorter in time since they spread robots out to minimize mutual spatial interference between robots. As the population size increases, such interference eventually dominates the travel cost. However, since ant algorithms (including LOST) tend to converge to a single "best" trail, in large populations this trail can become badly congested and performance reduced. To address this, we seek to perturb the ant algorithm such that it does not converge to a single trail when the interference is high, and instead to maintain multiple trails that spread the robots in space and time.

The contribution of this paper is to examine the effect of modulating the field of view (FOV) of the robots' trail-detecting sensor. We show that global throughput is a function of robot FOV, where narrower FOVs perform better in large populations. Experimental evidence suggests that the narrow FOV causes multiple trails to be maintained, so that the system can support larger population sizes before saturating due to interference. This simple means of controlling congestion does not require any change in the original trail following algorithm, or any additional sensing.

4.3 Related Work

Various different robot implementations of ant-like trail following have been presented. Real chemical marks were first used to produce true stigmergic trail-following in [46]. Also recently, Fujisawa et al. [47] carried on a study out of communication in a swarm of robots using pheromone and proposed a behavior algorithm for robots to search for prey and attract other robots. They used ethanol as pheromone in their real robot experiments. The challenge of chemical and sensor engineering makes these methods often impractical. A more parsimonious method was invented by Payton [48] where virtual pheromone trails are implemented by directional infra-red messages transmitted from robot to robot. Robots echo received messages, incrementing a contained hop-count which is used to estimate the distance to the message source. In both chemical and IR-mediated methods, the local "gradient" is sensed directly from the environment. If robots are mutually localized, virtual trails can be created from global waypoints, which are communicated by wireless network. We showed that this scheme can be robust to large zero-mean localization error [42], and admits a relaxed and practical definition of mutual localization [43].

The diminishing-to-negative-returns effect of increasing the number of robots on performance has been studied in related contexts. In a mathematical model of robot foraging [49], it was shown that adding more robots to the system improved the group performance while decreasing individual robot's performance. Based on that model, an optimal group size was found that maximizes the group performance. Explicit anti-interference strategies are studied in real robots in [50], to increase performance in the transportation task. Congestion control in a dense multi-robot system is studied in [51], where asymmetries that resolve conflicts are introduced by modifying either the environment or the robot controllers. In contrast, the method presented here is symmetric and can be considered complementary.

4.4 Localization-Space Trails (LOST) review

This section briefly reviews the generalized trail-following method formulated in [43].

LOST generates trails between the locations of *Events*. An Event is defined as a task-relevant occurrence that may happen to any member of the team, and is locally but reliably perceived. For example, in our transportation task the relevant Events would be 'pick-up-resource' and 'drop-resource'. A robot must be able to recognize these events in order to switch between resource-seeking behavior and home-seeking behavior. When an Event occurs to a robot, its current pose in localization space is recorded to create an [Event,Pose] tuple called a *Place*. A robot can then express information about the world relative to the Places it has seen.

Other robots that have position estimates for the same Events can interpret the coordinates in their own local frame of reference. Thus robots are mutually localized by the shared experience of the common task, rather than conventional global localization in some arbitrary coordinate system.

The purpose of LOST is to guide the robot to a Place currently of interest: the goal. The algorithm provides the robot controller with two pieces of information; (i) the *heading-hint* that is the local direction in which to travel to reach the goal; (ii) the *distance-hint* that is the estimated cost (usually in time) to reach the goal. These hints are extracted by examining a set of waypoints called *Crumbs* which are poses specified relative to a Place. The current set of Crumbs specified relative to a particular Place is a *Trail* to that place. A Crumb is a tuple $C = [P_c, L_c, d_c, t_c]$ containing the name of the Place P_c to which it refers, a localization space pose L_c, an estimate d_c of the distance (in some distance function) from L_c to P_c, and the time t_c when the Crumb was created.

Each robot maintains an initially empty temporary trail. Every S seconds, a robot inserts a new crumb to the temporary trail. The crumb contains the current location of the robot, the name of the most recent Event experienced by that robot, the distance from the last event, and the current time. When another event occurs to the robot (e.g. when a robot drops off its cargo), the temporary trail is broadcast to all robots, including itself, then deleted. A new temporary trail is then created for the recent Event.

Besides the temporary trail, each robot maintains a trail for each different Event it has learned about from the network. When a broadcast trail is received, the crumb poses are transformed into the local frame of reference by the rigid body transform defined by comparing the local and received poses of the trail's Place. The transformed crumbs are added to the local trail for this Place. All trails are periodically scanned and any Crumb with timestamp older than age threshold a seconds is discarded. Thus the trail is updated dynamically, and out-of-date information is expired. The dynamic response of the trail to changing environments is a function of a.

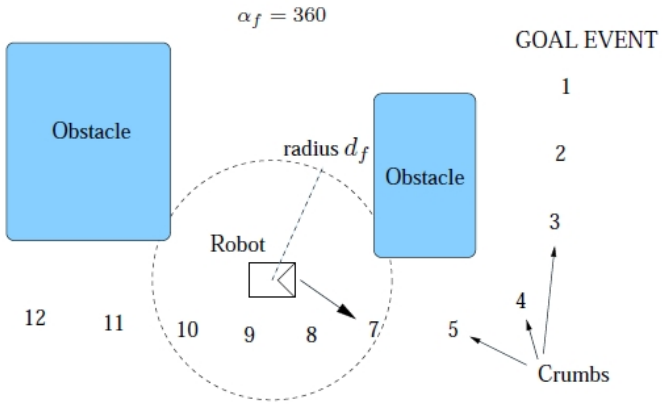
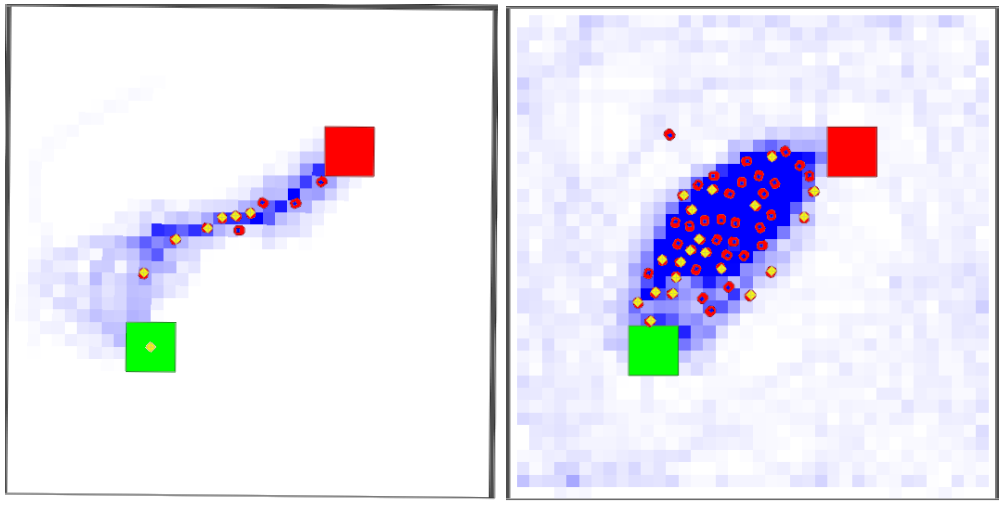


Figure 4.1: Sketch of the LOST algorithm, showing a trail of Crumbs with decreasing distance values leading to a goal Event. The robot moves towards the crumb within radius d_f that has the lowest distance estimate.

Suppose a robot at pose L_r has Place P_g as its goal, such as $Event(P_g)$ = 'drop-resource-at-home'. The robot searches the set of Crumbs with Place = P_g to find the set of crumbs that lie within its *field of view*, i.e. within radius d_f of L_r. From this set it finds the crumb C_L with the smallest distance-to-goal d_c. This distance is returned as the distance-hint. The heading-hint is the angle from the robot's pose L_r to $L_c = Pose(C_L)$. Figure 4.1 shows the robot's field of view which is a circle about the robots current location with radius d_f.

If the robot moves in the direction of the heading hint and repeats this process, it will

encounter crumbs with decreasing distance to goal values, and eventually arrive at P_g.



(a) 10 robots (b) 50 robots

Figure 4.2: Effect of increasing the population in trail-following robots

The robot will take the shortest route so far discovered from that location. By following the Crumbs dropped by the whole population, each robot benefits from the others' exploration; robots will find a reasonable route much more quickly than they would alone. The larger the population size, the greater the probability of finding a good route and the more quickly a good route is found.

4.5 Trail congestion

We seek to increase the throughput of our transportation system by adding more robots. But every added robot increases the probability of spatial interference, which can reduce global performance. Eventually the marginal value of adding another robot goes below zero. Figure 4.2 shows this phenomenon in our trail-following robot system implemented in the well known simulator Stage [3]. In Figure 4.2(a), 10 robots are successfully following a trail between source and sink (large squares). The small/blue squares are a normalized histogram of space occupancy over time: the robots can be seen to be frequenting the same places as they follow a reasonably direct route. Robots are able to navigate past each other using local obstacle avoidance and the throughput is 232 round-trips per hour, which is close to the ideal of ten times the single robot work rate (18 round-trips per hour). In Figure 4.2(b) 50 robots work in the same space. The histogram shows that the robots are still mostly working on a single direct trail. The space is now so crowded that local obstacle avoidance breaks down and robots make little progress. They also lose the trail frequently and explore to recover it. The throughput is 368 round-trips per hour, which is much less than 50 times the single robot workrate. The method described below aims to ameliorate this problem.

4.6 Changing the Field Of View

Using LOST, each robot moves toward the visible crumb with the minimum distance to goal. The more that two robots' fields of view overlap, the higher the probability that they will select the same crumb and thus follow the same trail. Our approach to congestion reduction is to modify the robots' field of view so that trail-following is still achieved, but to increase the probability of different best crumbs being detected. The hypothesis is that this can cause different trails to be followed and thus reinforced and maintained, spreading robots out in the environment to reduce interference while still making progress on the transportation task.

An analogous mechanism may be used in biological systems. For example in the recruitment behavior of honey bee colonies, unemployed foragers will select at random a single bee displaying food source information, while ignoring all the others. Thus individual bees are not fully informed, and may choose to forage sources other than the best available. It has been argued that this has advantages for the colony overall by preventing overconvergence, for example maintaining exploration of the environment as it changes [53].

To selectively hide crumbs from the LOST forager, we simply vary the radius d_f and reception angle α_f of their virtual crumb-detecting sensor. The FOV of the robot always points forward. In all previous work, α_f was effectively 360°.

4.7 Experiment 1

4.7.1 Simulation Setup

To test our hypothesis, we ran Stage simulations with the task environment shown in Figure 4.2. The arena is 20x20m, with robot length 0.45m, and free of obstacles. Robots are Stage's Pioneer 3DX and SICK LMS200 laser rangefinder models. The bottom left (green) square is the source; top right (red) square the sink of resources. In the screenshots, robots (red polygons) are shown with yellow diamonds to indicate they are carrying a unit of resource. Robots start every trial at the same randomly-chosen uniformly distributed positions, do not know the initial location of source and sink locations, and must find them by exploration at the start of the trial. Each trial runs for 30 minutes, and the total number of resources delivered at the end of the trial is our performance metric. 10 trials are performed for each of a range of settings of radius d_f, reception angle α_f, and population size. LOST is deterministic but the local obstacle avoidance and searching is stochastic (for robustness), hence the need for repeated trials.

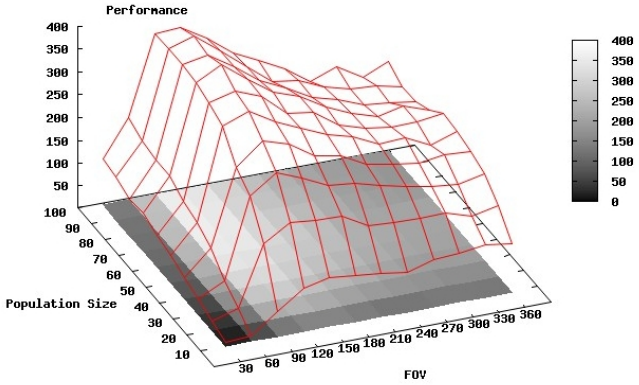
Experiment 1 examined all permutations of $d_f = [1.5, 2.0, 2.5]$ meters, $\alpha_f = [10, 20, \ldots 360]$ degrees, population $P = [10, 20, \ldots 100]$ robots.

4.7.2 Results

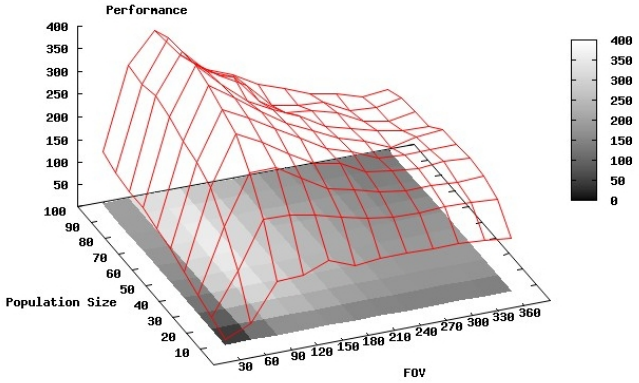
The results of the first experiment are summarized in Figure 4.3, with mean performance over 10 repeated trials plotted for each $[d_f, \alpha_f, P]$ configuration. Error bars are ommitted for clarity: the variance is $< 12\%$ in 80% of experiments.

The FOV range parameter d_f appears to have relatively little effect on the performance, but the FOV angle α_f appears to have an important effect. The results show that, with a constant d_f, a small team of 10 robots has about the same performance for any α_f above 90 degrees. As the population size increases, the performance is better for smaller α_f, until a lower bound is reached. Performance falls off quickly with α_f below 90 degrees in all cases.

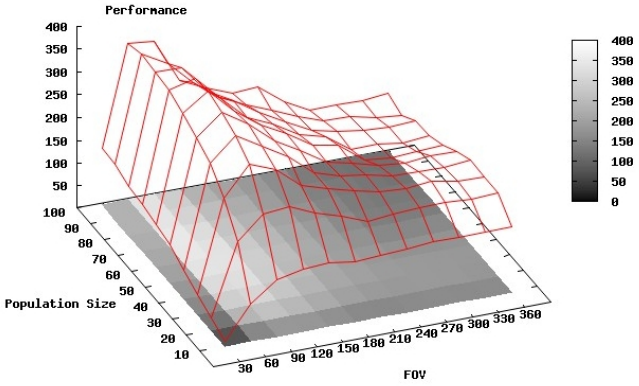
To verify that the performance results are significantly different for different values of α_f, we performed hypothesis testing using a T-test. The P values for the hypothesis that the performance values for α_f=90 and α_f=180 are from the same distribution are given Table 4.1. For all population sizes above 10, the test suggests that the distributions are significantly different, and combined with the higher mean scores for $\alpha_f = 90$, we conclude that $\alpha_f = 90$ performs better than $\alpha_f = 180$ for all populations above 10.



(a) With lookahead distance $d_f = 1.5m$



(b) With lookahead distance $d_f = 2m$



(c) With lookahead distance $d_f = 2.5m$

Figure 4.3: Results of Experiment 1, in a world with no obstacles, showing the mean number of resources transported by robots with different populations and field of view configuration. Variance is $< 10\%$ for each point.

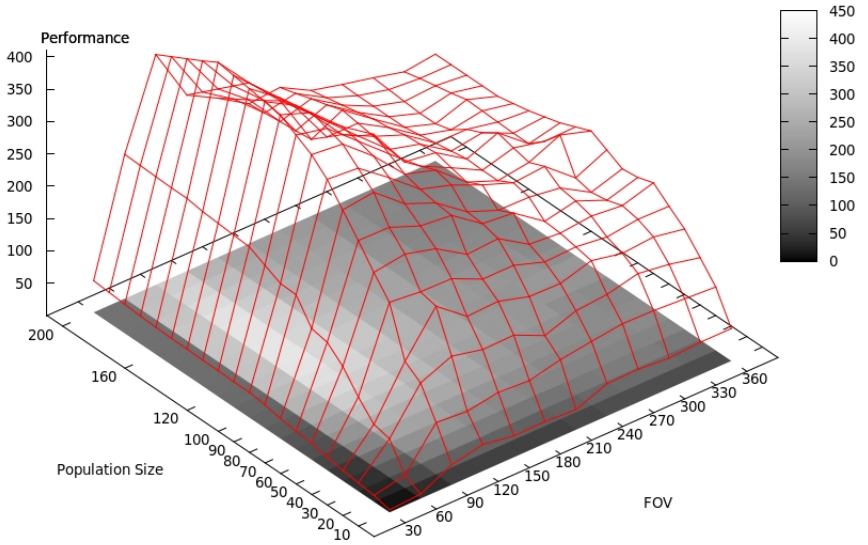


Figure 4.4: Results of second experiment

These results show improved performance at large population sizes, suggesting we have achieved a reduction in interference.

To explain how this is happening, see Figure 4.5, which shows normalized histograms of the last 5 minutes of robot positions at 10, 20 and 30 minutes during one of the trials of Experiment 1 $[d_f = 2, \alpha_f = 90, P = 70]$. Multiple trails between source and sink can be perceived, and the robots spread out in the environment, using these trails. Comparing Figure 4.6, with a wider FOV $[d_f = 2, \alpha_f = 180, P = 70]$, we see fewer, wider trails and robots closer together.

4.8 Experiment 2

To test larger populations and a more challenging search task, we performed a similar experiment in a 4 times larger world (40x40m) containing obstacles. The experimental procedure is identical, testing all permutations of $d_f = 2.0$ meters, $\alpha_f = [10, 20, \ldots 360]$ degrees, population $P = [10, 20, \ldots 100, 120, 160, 200]$ robots.

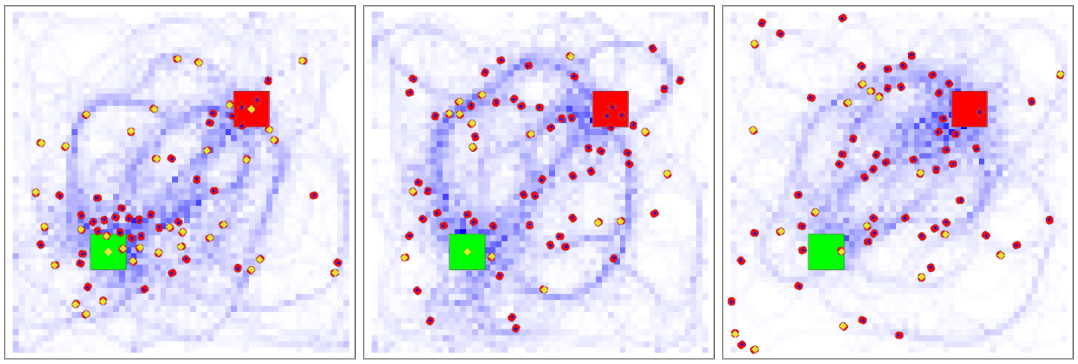
The results are plotted in Figure 4.4, showing a similar trend to Experiment 1. Performance is not improved by employing more than 100 robots, but performance is better for smaller values of $\alpha_f > 60$ once the population rises above 50 robots. Hypothesis testing supports this interpretation (Table 4.1 right side).

Occupancy histograms are shown in Figures 4.7 $[d_f = 2, \alpha_f = 90, P = 120]$ and 4.8 $[d_f = 2, \alpha_f = 180, P = 120]$. The smaller FOV angle produces more trails that are spread around the space. The larger FOV angle produces fewer trails, the shortest of which are heavily used, creating congestion.

A second important feature (we believe) is the lack of congestion at the source and sink locations when the smaller FOV angle is used. Congestion at these areas is very significant since all robots must access them eventually.

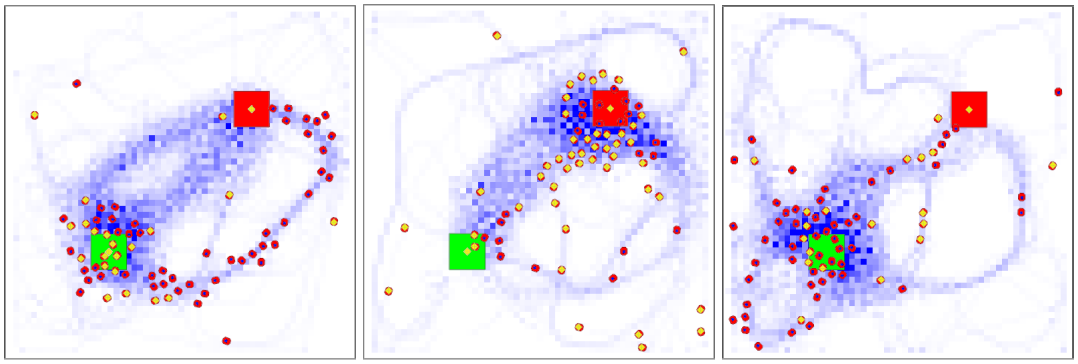
4.9 Discussion

The results above support our hypothesis that LOST robots with a narrow ($60 < \alpha < 120$ degree) field of view perform better than the original 360 degree FOV. The occupancy histograms



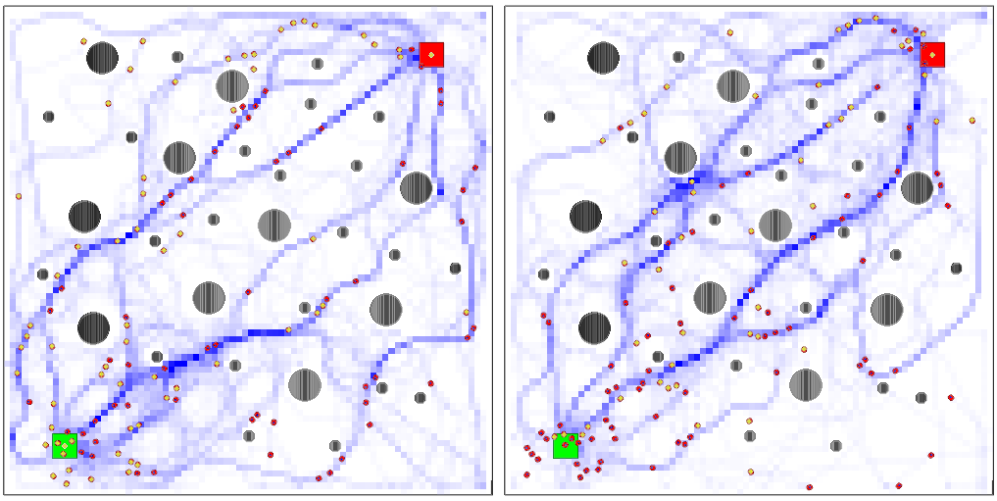
(a) $t_1 = 10min$ (b) $t_2 = 20min$ (c) $t_3 = 30min$

Figure 4.5: Histograms of robots' locations in the last 5 *mins* with $\alpha = 90, d_f = 2m$



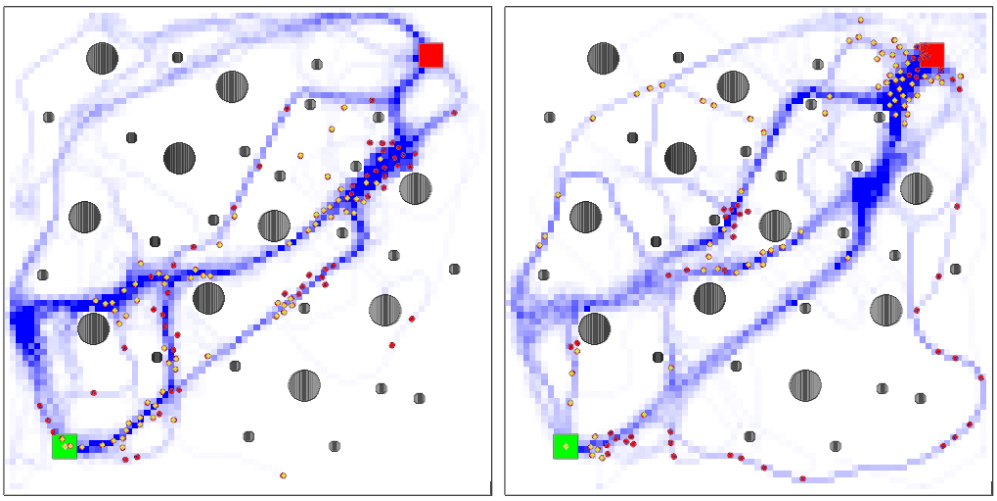
(a) $t_1 = 10min$ (b) $t_2 = 20min$ (c) $t_3 = 30min$

Figure 4.6: Histograms of robots' locations in the last 5 *mins* with $\alpha = 180, d_f = 2m$



(a) $t_2 = 20min$ (b) $t_3 = 30min$

Figure 4.7: Histograms of robots' locations in the last 5 *mins* with $\alpha = 90, d_f = 2m$



(a) $t_2 = 20min$ (b) $t_3 = 30min$

Figure 4.8: Histograms of robots' locations in the last 5 *mins* with $\alpha = 180, d_f = 2m$

Table 4.1: Results of hypothesis testing, showing the result of a t-test between the datasets gathered using $\alpha_1 = 90, \alpha_2 = 180$ for different population sizes in both experiments.

Population Size	Parameter p Experiment 1	Parameter p Experiment 2
10	0.25	0.69
20	< 0.005	0.93
30	< 0.005	0.25
40	< 0.005	0.12
50	< 0.005	< 0.015
60	< 0.005	< 0.03
70	< 0.005	< 0.024
80	< 0.005	< 0.001
90	< 0.005	< 0.0001
100	< 0.005	< 0.0001
120	-	< 0.0001
160	-	< 0.0001
200	-	< 0.0001

show that when the FOV is restricted, robots form a number of less-direct paths, each with a reduced probability of interference. But when the robots have a larger field of view, they converge to a smaller number of more-direct paths, with increased probability of interference.

The dispersal of narrow FOV robots over multiple trails can be seen as a spontaneous load balancing on the routes. Consequently, the robots enter/exit home and source from different sides which leads to another mechanism to distribute robots into trails. This also reduces the congestion near home/source.

With small population sizes where interference is not significant, the narrow field of view performs no differently than the original 360 degrees.

4.10 Conclusion and Future Work

In this paper we showed how changing the field of view of a sensor can influence the overall system performance of an ant-inspired foraging-and-trail-following robot system. It was shown through simulation experiments that if the receptive angle of the trail-marker sensor is narrow, but not too narrow, (about 90 degrees) the overall performance of our swarm was maximized. To the best of our knowledge, this is the first study that shows hiding some information from the agents can reduce the congestion and improve the overall performance. The "worse-is-better" data-hiding idea used here may be applicable to other large-scale multi-robot systems.

In future work we will investigate navigation strategies for trail-following multi-robot systems. To date, our LOST robots use a very simple controller for trail-following and local obstacle avoidance. We expect that performance can be improved by adding more data into crumbs, and exploiting the trails more intelligently. Also, local coordination strategies such as flocking or formations could increase navigation efficiency and possibly throughput while maintaining the attractive features of the LOST method.

Code publication

All source code, scripts, etc. used to produce the results reported in this paper are available online:

URI: http://autonomy.cs.sfu.ca/source/abbas_blinkered.tar.gz
SHA1: b0f49743f408701a4ce0411f2cff296fbb8b215e

Chapter 5

Fall in! Sorting a group of robots with a continuous controller

Yaroslav Litus and Richard Vaughan [6]

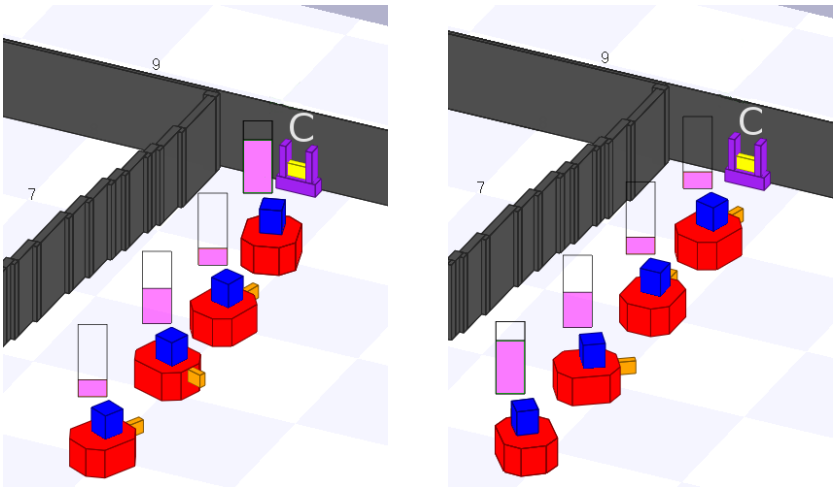
5.1 Introduction

Ordering robots may be a necessary part of many robotic tasks. For example, a team of robots may need to board a transport vehicle in a certain order to use cargo space more efficiently. Likewise, a certain order may be preferred when deploying robots from that vehicle. Maintaining a priority queue of robots may be required when robots line up for service or refueling (see Fig. 5.1) or when robots perform a convoying task. Finally, ordering robots may improve performance of certain spatial interference resolution algorithms [50].

Sorting is usually solved by algorithms running on discrete state machines. To our knowledge this is the first paper to explicitly consider performing such a discrete computation by a continuous dynamics of a multi-robot system. Other multi-robot systems either use continuous dynamics to solve continuous problems [54, 55] or use discrete dynamics to solve discrete problems [48]. We describe a decentralized multi-robot controller that sorts robots by coupling them according to the Brockett double bracket flow system. This controller is a novel robotic application of the well-known Brockett system.

Formally, let n robots on a plane be initially positioned on a line $y = y_0$ at coordinates $(x_i(0), y_0)),\ i = 1, \ldots, n$ in a Cartesian coordinate system. Assuming, without loss of generality, that the sought ordering coincides with the robot numbering, the goal is to drive the system to a state $(x(t), y(t))$ where $x_1(t) < x_2(t) < \ldots < x_n(t)$ and $y_1 = y_2 = \ldots y_n = y_0$. That is, we want the robots to sort themselves along the horizontal axis and return to the line where they started. The sorting computation should be performed solely by the dynamics of interacting and moving robots.

The next section presents related work, which is followed by a description of the Brockett sorter, a dynamical system capable of performing sorting by means of smooth dynamics. After an informal theoretical argument about issues that may arise in using the Brockett sorter in a real robotic system we describe a controller based on this sorter. The controller is tested in a short demonstration followed by discussion of the observed behavior and properties of the system. The paper concludes by suggesting several directions for future work.



(a) Robots lined up in arbitrary order. The last robot in the queue has a high chance of running out of charge before reaching the station.

(b) Robots are sorted by the remaining charge. The chance of starvation is minimized.

Figure 5.1: Robots queuing for recharging at the station "C" located near the wall. The share of remaining charge is shown for every robot. Sorting robots increases the probability that all robots can charge before running out of energy.

5.2 Related work

The idea of viewing robotic systems as computational devices is related to the Brooks' advice against internalizing the world [56]. As long as the computation problem faced by a robotic system is set in terms of the variables that the system can modify through its behavior, required computation can potentially be performed by that behavior. The computation is hence externalized and happens outside of the internal information processing unit. From this point of view a robotic system can perform computations not only by using its conventional dedicated processing units, but also by its sensors and actuators.

These computational capabilities have some recognition in the robotics community. In the domain of single robot systems Paul [57] and Pfeifer [58] show that robot morphology can assume a computational role. In multi-robot systems Payton et.al. [48] introduce a concept of "world embedded computation" and describe a robot swarm system that runs an analogue of Dijkstra's shortest path algorithm in a world embedded manner. Robots spread themselves in the environment and then find the shortest path between two points by exchanging local messages. Hamann and Wörn [55] discuss the idea of embodied computation and present a swarm system that computes an approximation to the solution of geometric Steiner tree problem. Litus and Vaughan [54] argue that embodiment and spatial embeddedness can serve as a surrogate for computational resources for developing decentralized distributed gradient descent optimization algorithms for teams of embodied agents.

Since robotic systems evolve in continuous state space, the relation between discrete computation and continuous systems is very important for treating robot systems as computers. Brockett [59] shows that double-bracket matrix flow dynamical systems can serve as an analog computer solving a variety of combinatorial problems including sorting. These systems bear similarity to finite aperiodic Toda lattices, a recent thorough review of which is given by Kodama and Shipman [60]. Saxena and Clark [61] describe an electronic implementation of

Brockett double bracket flow built from analogue integrated circuits. Zavlanos and Pappas describe a dynamical system inspired by the double bracket flows that approximates the solution to the combinatorial weighted graph matching problem [62]. Bloch and Crouch [63] argue that from control theoretical point of view some combinatorial problems can be seen as minimization of a function over a set of configurations which can be contrasted with minimization over a class of curves in classical optimal control. The problem of sorting robots could be related to formation control (see Bahceci et.al. [64] for a recent review).

5.3 Brockett smooth sorter

In a seminal paper [59] Brockett analyzes dynamical systems

$$\dot{H} = [H, [H, N]] \tag{5.1}$$

where H and N are symmetric matrices and $[A, B] = AB - BA$. He shows that these so called "double bracket flow" systems define an isospectral gradient flow, so as H evolves its eigenvalues do not change. The author proves that these systems can serve as a versatile analog computer solving linear programming problems, certain combinatorial optimization problems and diagonalizing symmetric matrices. This last ability is of a particular interest for robot sorting since it provides means to sort a list of numbers by a smooth dynamical system.

Brockett proves that if N is a diagonal matrix with distinct elements, the equilibrium matrix $H(\infty)$ will be a diagonal matrix with its elements arranged in the same order as elements of N. In particular, if $N = \mathrm{diag}(1, 2, \ldots, n)$ then for almost all Θ and for

$$H(0) = \Theta^T(\mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n))\Theta$$

the equilibrium $\dot{H} = [H, [H, N]]$ will approach

$$H(\infty) = \mathrm{diag}(\lambda_{\pi(1)}, \lambda_{\pi(2)}, \ldots, \lambda_{\pi(n)})$$

where permutation π sorts the final list by its size. Our choice of $H(0)$ is dictated by a need to decrease the number of state variables in order to simplify the controller. Diagonal $H(0)$ has the fewest number of variables, but produces no dynamics. However, symmetric tridiagonal matrix $H(0)$ will produce the desired dynamics and result in $H(t)$ being symmetric tridiagonal for any time t. Setting

$$H(0) = \begin{pmatrix} b_1 & a_1 & & \\ a_1 & \ddots & \ddots & \\ & \ddots & \ddots & a_{n-1} \\ & & a_{n-1} & b_n \end{pmatrix} \tag{5.2}$$

where $b_i, i = 1, 2, \ldots, n$ are the values to be sorted and $a_i, i = 1, 2, \ldots, n-1$ are small non-zero values will make $H(\infty)$ a diagonal matrix containing entries that approach the sorted list of eigenvalues that are close to b_i. The smaller the values of a_i, the closer matrix $H(0)$ is to being diagonal and having b_i as its eigenvalues, and the closer the diagonal of $H(\infty)$ is to the list of sorted components of b_i. Decreasing a_i, though, comes at the cost of increasing convergence time.

System (5.1) with $N = \mathrm{diag}(1, 2, \ldots, n)$ and $H(0)$ as in (5.2) is equivalent to symmetric tridiagonal Toda equations [65]. These equations describe the behavior of the system of n unit mass particles arranged along a line with adjacent particles interacting with a magnitude that exponentially depends on the distance between them [66]. Component-wise in terms of a and b dynamics of this system can be written as

$$\dot{a}_k = a_k(b_{k+1} - b_k),\ k = 1, 2, \ldots, n-1 \tag{5.3}$$

$$\dot{b}_k = 2(a_k^2 - a_{k-1}^2),\ k = 1, 2, \ldots, n \tag{5.4}$$

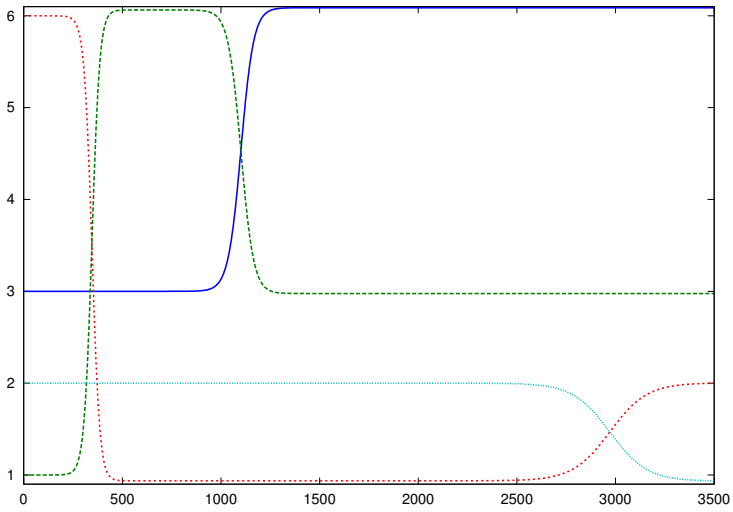


Figure 5.2: Dynamics of sorting system (5.3-5.4) initialized with $b(0) = (3, 1, 6, 2)$, $a(0) = (.001, .001, .001, .001)$. Four components of vector b (vertical axis) are plotted against time (horizontal axis). In 3 500‘ simulation steps the system converges to $b = (6.088, 2.976, 1.999, 0.937)$.

with initial conditions $a_0 = 0$.

Fig. 5.2 illustrates the evolution of the system (5.3-5.4) with initial values $b(0) = (3, 1, 6, 2)$, $a(0) = (.001, .001, .001, .001)$. The system was simulated in discrete time as $b(t+1) = b(t)+\delta\dot{b}(t)$, $a(t + 1) = a(t) + \delta\dot{a}(t)$ with discretization parameter $\delta = 0.005$. Four lines on the figure show the values of four components of vector b plotted against simulation time. After 3500 steps the value of the vector b converges to $b(3500) = (6.088, 2.976, 1.999, 0.937) \approx (6, 3, 2, 1)$: the desired ordering.

5.4 Application to robot sorting

5.4.1 Theoretical considerations

The dynamical system (5.3-5.4) (hereinafter “the sorting system”) can serve as a basis for a multi-robot system controller that sorts robots. To do this we need to find a way to embed the sorting system into the state space of a multi-robot system. This way the movement of the robots can be dictated by the evolution of sorting system and result in robots sorting themselves.

Regardless of the means of embedding the sorting system into the state space of a multi-robot system we need to acknowledge the limitations a realistic robot system imposes on the possible trajectories in a state space. For example, robots have limited speed and acceleration and may be non-holonomic. Also, a real system will have noise present in sensor readings and responses to control inputs. Finally, collision avoidance should be employed by a real robot system introducing additional constraints on trajectories. This raises an important question: is the sorting system able to withstand some perturbation of the state variables as the system evolves and still converge to the same equilibrium? A brief analysis shows that this is not the case. Assume that sorting system evolves from tridiagonal matrix $H(0)$ to diagonal matrix $H(\infty) = \mathrm{diag}(\lambda_{\pi(1)}, \lambda_{\pi(2)}, \ldots, \lambda_{\pi(n)})$ and at time t one of the state variables was perturbed resulting

in matrix $H'(t) \neq H(t)$. As long as the spectrum of $H'(t)$ differs from the spectrum of $H(t)$ the system will now converge to a different equilibrium $H'(\infty) = \mathrm{diag}(\lambda'_{\pi(1)}, \lambda'_{\pi'()}, \ldots, \lambda'_{\pi(n)})$. There is no feedback in the system to correct this deviation from the original trajectory and restore original spectrum of H. In this sense the sorting system is fragile and using it as a base for a robot sorting controller seems to be difficult.

However, the sorting system should not be discarded because of its fragility. Despite the sensitivity of the equilibrium to the perturbation of state variables, all equilibriums are in fact diagonal matrices with sorted eigenvalues. Thus, if some state variable s_i of the i-th robot in the sought ordering behaves as the i-th diagonal entry of H, the equilibrium values $s_i(\infty)$ will follow the sought ordering irrelevant of the changes in the spectrum of H brought by the deviations from the perfect trajectory. In other words, though the actual values of $s_i(\infty)$ will differ from what they could have been in the absence of perturbations, as long as the system is allowed to converge the values of robots state variables $s_i(\infty)$ will be sorted. In this sense the sorting system is reliable and we can attempt to use it to control an appropriately constructed robot system.

5.4.2 Implementation

We will embed the sorting system into the robot system as follows. x_i will correspond to the diagonal entries of H while $y'_i = y_i - y_0 + \epsilon$ where ϵ is a small non-zero value will correspond to non-diagonal entries of H. In these variables, sorting system (5.3-5.4) can be rewritten as a first order controller

$$\dot{x}_1 = 2y_1^2, \tag{5.5}$$

$$\dot{x}_i = 2(y_i^{'2} - y_{i-1}^{'2}),\ i = 2, \ldots, n-1 \tag{5.6}$$

$$\dot{y}_i = \dot{y}'_i = -(x_i - x_{i+1})y'_i,\ i = 1, \ldots, n-1 \tag{5.7}$$

$$\dot{x}_n = -2y_{n-1}^{'2}, \tag{5.8}$$

$$\dot{y}_n = y'_n = 0; \tag{5.9}$$

This controller requires every robot i to use the following information:

1. y'_i, the vertical distance from the original line $y = y_0$
2. y'_{i-1}, the vertical distance of the previous robot (if there is one) in ordering from the original line $y = y_0$
3. $(x_i - x_{i+1})$, horizontal distance to the next robot (if there is one) in the ordering

Note, that y'_{i-1} can be computed from y'_i if robot i knows the vertical distance to robot $i-1$. Therefore, the controller requires the robot to be partially localized (know an estimate of its y coordinate) and be able to estimate horizontal distance to one robot and vertical distance to another robot. These requirements can be met by various sensory/communication solutions. To avoid any sort of communication, in the demonstration robots are equipped with fiducial detectors that can determine the relative bearing and distance to the previous and next robot in the ordering. Robots are also localized, though we use only orientation and the y coordinate. Horizontal or vertical distance to a team member is calculated from the position of self, and estimates of distance and bearing to the team member. Thus, all information required by the controller is available in this robot system.

There are several issues that emerge when using this controller in a realistic system (or realistic simulation). First, fiducial finders require clear line of sight and hence occlusion can prevent the robot from getting information about distances to other robots. Second, as all sensors, fiducial finders are noisy, therefore information about bearing and distance to the teammates is imprecise. Third, all robots have bounded magnitude of their speed vector, and

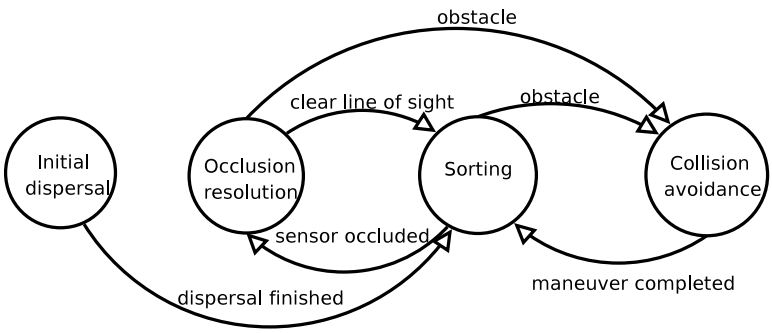


Figure 5.3: State diagram of the robot sorting controller

non-holonomic robots have also bounds on the direction of this vector. Finally, robots can not drive through each other so collision avoidance should be used. All these issues will force the robots to deviate from the trajectory prescribed by the sorting system, however as we argued above we still expect the robots to converge to the sorted order. We will address aforementioned issues in the following paragraphs. The state machine of the resulting controller is shown on Fig. 5.3.

Occlusions Robots have no knowledge of the team size or their absolute position in the ordering. They are capable of sensing positions of the previous and the next robot in the ordering if the line of sight is not occluded by other robots. Initially every robot assumes that it is the only robot in the team. Once robot i senses the previous robot $i-1$ in the ordering, without sensing robot $i+1$ before, robot i assumes that i is the last robot. It will behave accordingly and use the relative position of robot $i-1$ in its future calculations of speed vector. If robot i senses the next robot $i+1$ in the ordering without previously sensing $i-1$, robot i assumes that i is the first robot. If robot i senses $i-1$ and $i+1$ simultaneously or in any order, it knows that i is not the first and not the last robot and needs relative positions of both $i+1$ and $i-1$ to calculate its speed vector. Therefore, the robot discovers its position in the ordering (last,first, in between) as the dynamics unfolds.

Regardless of the currently assumed position of robot i if its fiducial sensor is occluded and robot i can not see one or both of the robots it needs to calculate the speed vector, it reduces speed to a certain predefined constant without changing its direction. Once the line of sight is restored, the robot selects the desired speed and direction following Eq. (5.5). Since initially all robots are located along a line and almost all fiducial finders are occluded, the robots disperse themselves by moving with random speeds along the vertical axis for a fixed predefined time. With high probability this allows most of the robots to observe the required teammates and follow Eq. (5.5-5.9).

Noise We will not use any noise reduction techniques and instead treat all noisy sensor readings as true values.

Bounds on a speed vector If the magnitude of the desired speed vector $\vec{v} = (\dot{x}, \dot{y})$ exceeds the maximum speed $V_{\max}$ of the robot, the robot tries to set its speed vector to $\vec{v}_{\text{clamped}} = \frac{\vec{v}}{\|\vec{v}\|} V_{\max}$ thus attempting to go in the desired direction with the maximum speed. In the demonstration below we use non-holonomic steering robots which are driven by a simple

Maximum speed	1.2 m/s
Collision avoidance speed	0.1 m/s
Collision avoidance turning speed	0.5 rad/s
Collision avoidance initiation distance	1 m
Minimum front stopping distance	0.5 m
Collision avoidance duration interval	[1,2]s
Speed during occlusion	0.2 m/s
Initial dispersing duration	1.5s
Fiducial finder bearing accuracy	±3 degrees
Fiducial finder range accuracy	± 15 mm

Table 5.1: Parameters used in Stage simulation

negative feedback controller

$$\dot{\theta} = \theta - \angle\vec{v} \tag{5.10}$$

$$s = \|\vec{v}\| \cos(|\dot{\theta}|), \tag{5.11}$$

where θ is the robot bearing, $\angle\vec{v}$ is the direction of the desired speed vector in the same coordinate system, s is the driving speed.

Collision avoidance We employ a simple collision avoidance algorithm that uses laser range-finder sensor readings. If there is an obstacle closer than a certain distance d_{stop}, the robot stops. If there is an obstacle which is at closer than a certain distance $d_{\text{avoid}} > d_{\text{stop}}$ then the direction that gave the smallest distance reading is found. If smallest reading came from the direction to the right of the robot bearing, a collision avoidance maneuver with a duration randomly selected in a certain interval is performed. The robot starts to turn left with a fixed turning speed and driving speed. Otherwise, the robot performs a right turn maneuver. If smallest reading came from the left, a right turn maneuver is performed. Once the collision avoidance maneuver is over, the robot continues to set the speed as prescribed by Eq. (5.5-5.9).

5.4.3 Demonstration

For the demonstration we use a team of simulated robots. Robots are placed along a horizontal line and the sorting controller is executed simultaneously on every robot. We perform two kinds of simulations. In the first, "idealistic", simulation the robots are holonomic, with no speed restrictions, no sensor occlusions and no collisions. Therefore, the dynamics of this system are described by Eq. (5.3-5.4). In the second, "lifelike" simulation Pioneer robots are simulated in the well-known Stage robot simulator [67]. Simulated Pioneer robots can collide, so they need to use collision avoidance, they have non-holonomic driving, top speed restriction and their fiducial sensors can be occluded by other robots. Sensor noise is simulated by adding uniformly distributed random errors to the true distance and bearing reading of fiducial finder before they are used by a controller. Robots in the Stage simulation use the controller described in Section 7.3. Parameters of the Stage simulation are given in Table 5.1. All simulations run until the speeds of all robots converge to a near-zero value.

Figure **??** shows the trajectories produced by the robots for three different initial conditions. For every initial condition two trajectories are shown. The first trajectory is produced by an idealistic simulation, the second by a lifelike Stage simulation. Note that Stage simulations are non-deterministic because of the sensor noise and initial random dispersal and infinitely many different trajectories are possible of which we show only one. The idealistic simulation trajectory, on the other hand, is repeatable and fixed up to the rounding and discretization errors during simulation. Each trajectory is described by two graphs. The first graph plots

values of x coordinates of robots against simulation time, the second shows the joint (x, y) trajectories of the robots.

In the idealistic simulations most of the robots initially start moving with small speed with vertical component of the speed vector dominating horizontal component. As robots move away from the start horizontal line their speed grows, the horizontal component of the speed vector increases and the vertical component decreases to the point where the vertical component becomes negative and robots begin to return to the start horizontal line but with different horizontal coordinate. Some of the robots move along the start horizontal line for the part of their trajectory while the last, n-th robot never leaves the start line moving only horizontally (see Eq. (5.9)). Depending on the initial conditions robots may depart from and return to the start line several times before the robots converge to the sorted order (see, e.g., robots starting at positions $x = 3$ and $x = 5$ on Fig. **??**,**??**). Also, robots may switch their vertical direction before reaching the start line (see, e.g., robot starting at positions $x = 13$ on Fig. **??**, **??** and robot starting at position $x = 8$ on Fig. **??**, **??**). Once the sorted state is reached, robots do not depart from it any more. The final configuration has robots rearranged in the sorted order along the start horizontal line with the set of final horizontal positions having values very close to the set of original horizontal positions. Therefore, the robots not only end up in the sorted order, but they also jointly occupy same horizontal positions that where occupied by the team initially.

In the Stage simulations robots initially disperse themselves by moving for a fixed time with a random speed in a vertical direction. This eliminates some of the occlusion and allows some robots to move in the direction prescribed by the sorting system. After dispersal the robots start moving with increasing speeds which is limited by the top speed of the robot. Occlusions that were not resolved by the initial dispersal are eventually resolved as occluded robots move, slowly creating new lines of sight (see Section 5.4.2). Some occlusions are resolved by the collision avoidance behavior. Robots move away from the horizontal line with horizontal component of their speed vector increasing and vertical component decreasing until the vertical component changes the sign and robots return to the start line at a different horizontal coordinate. Part of the robot trajectory may include a horizontal segment when robot moves along the start line without departing from it. As in the idealistic case, a robot may return to and depart from the start line several times (see robot starting at position $x = 5$ on Fig. **??**, **??**). Robots can also switch their vertical direction before reaching the start line (see the robot starting at position $x = 8$ on Fig. **??**, **??**). If the robots meet, they initiate collision avoidance which may be repeated several times if their desired trajectories lie close to each other. The occlusion resolution described in Section 5.4.2 ensures that the robots keep moving even if they can not observe one or both of their neighbors, thus eventually restoring the line of sight. Once the robots reach the sorted order, there are no occlusions and robots can finish convergence by bringing their speeds to zero. The final configuration has robots rearranged in the sorted order along or close to horizontal start line. However, they jointly occupy horizontal positions that differ from those occupied by the team initially.

5.5 Discussion

Both idealistic and lifelike simulations result in successful sorting of the robots. However, in a lifelike simulation robots end up converging to a set of positions that differs from the original set. That agrees with theoretical considerations stated in Section 7.3 as departures from the perfect trajectory caused by speed limitations, occlusions and collision avoidance change the eigenvalues of matrix $H(t)$ in Eq. (5.1) and, thus, the set of final horizontal positions. Therefore, the limitations of real robots used in simulations break down the position set preservation property of the ideal sorting system while still reaching the goal of sorting robots.

Another difference between the idealistic and life-like simulation is the convergence time. Life-like simulations take more time to converge due to the speed limitations, time spent on

collision avoidance, and time spent moving during the occlusions when trajectory may stray away from the convergence path. For example, it takes approximately 10 seconds for idealistic system to sort robots under conditions shown on Fig. **??**, while life-like Stage simulation takes about 90 seconds. With more robots (see Fig. **??**) the idealistic system still takes 10s to sort robots while the life-like simulation takes approximately 300 seconds to converge. A similar difference is observed under different initial conditions (see Fig. **??**) where the idealistic system converges in 7 seconds, while the life-like system sorts robots in 250 seconds. Fiducial sensor range imposes another limit on the practicality of this system as an increase in the team size will lead to an increase in the distance between robots which will eventually exceed the fiducial sensor range.

Formal analysis of the behavior of this system, including convergence properties is desirable but presents difficulties. While including non-holonomic driving and noisy sensors and controls into the formal model of the system seems tractable, sensor occlusions and collision avoidance present a major obstacle. Sensor occlusions result in a non-smooth changes in the robot controls and require analysis that is significantly more complicated than the analysis of the double flow system itself. Likewise, the simple threshold based obstacle avoidance algorithm we employ introduces non-smooth transitions into the system which are further complicated by a random selection of the collision avoidance duration. Even if more tractable deterministic repulsive potentials are used for collision avoidance, convergence analysis of the flow based system is prohibitively challenging[68].

Due to the slow convergence observed in simulations the described system should be viewed as a proof of concept rather than a suggested practical solution. While this system can definitely be used in situations where restrictions on the robot communications and sensing preclude using other sorting methods, it is desirable to find ways to accelerate the convergence and make extensive experimental evaluation of the modified system before it could be recommended as an engineering recipe.

5.6 Conclusion

We described a multi-robot system sorting controller based on a smooth Brockett double-bracket flow equation. This controller provides a novel demonstration of computational capabilities of multi-robot systems solving a combinatorial problem by means of continuous dynamics. The controller integrates the Brockett system with non-holonomic steering, simple collision avoidance and sensor occlusion resolution. This strength of this approach is in the fact every robot needs to identify only two or one other robots in the team. No conventional pairwise robot-robot comparisons that standard sorting algorithms suggest are necessary. The robots are reactive agents with information between robots exchanged by means of relative position sensing. Robots have no global knowledge of the system state and very limited memory capacity (memory is used only for dispersal and collision avoidance timers and queue position status). These modest information processing requirements for robots come at a cost of slow convergence and a potential need for long distance sensing which is the weakness of this approach.

Future work includes three main directions. The first direction is performance improvement of the robot sorting controller. This includes devising faster and more reliable collision avoidance strategies, trying to include some feedback mechanisms to correct changes in eigenvalues and thus preserve the set of original horizontal positions, looking for ways to bring down the system convergence time and eliminate the need for long-distance sensing. The second direction is looking for other ways to embed the Brockett sorter into a multi-robot system. While this controller uses the vertical coordinate of the robot as a non-diagonal entry in the H matrix, other modalities may be used. For example, robots can display values of non-diagonal entries by emitting sound, or changing the color or intensity of a display light. Also, more state variables may be involved by considering non-tridiagonal matrices H. Finally, as double-bracket flow systems' computational capabilities include more than sorting, these capabilities should be

evaluated in the context of multi-robot systems.

Simulation code

In accordance with the Autonomy Lab's policy on code publication, the source code is made available online at
`http://www.sfu.ca/~ylitus/robotSortingSources.zip`
MD5SUM: cb0eb487bf06a0e62b75350f377dcd3f.

Chapter 6

Adaptive Mobile Charging Stations

Alex Couture-Beil and Richard Vaughan [10]

6.1 Abstract

We consider systems of mobile robots that execute a transportation task and periodically recharge from a docking station. The location of the docking station has a considerable effect on task performance. In nonstationary tasks the optimal dock location may vary over the length of the task. In multiple-robot systems, spatial interference between charging and working robots can make it difficult to find an optimal dock location, even in static tasks. We propose a new approach whereby the dock is itself an autonomous robot that attempts to incrementally improve its location. We show simulation results from a simple local controller that adapts to nonstationary tasks and spatial interference, and thus improves overall task performance compared to a static dock.

6.2 INTRODUCTION

If a mobile robot is to expend more energy in work than it can store in an initial charge, it must have a means of obtaining more energy during runtime. The most common strategies for powering long-lived autonomous robots are (i) capture ambient energy directly from the environment, e.g. with solar panels, or (ii) transfer energy from a deliberately provided source, such as a charger connected to an electrical outlet. Mechanically coupling with a charging station provides a reasonable recharging rate and straightforward electromechanical design, and so is widely used.

Consider a prototypical autonomous mobile robot task, where robots must collect resources at one location and drop them off at another. To survive for long periods, they must drive to a charging station (a *dock* hereafter) before their stored energy is exhausted, recharge, then return to collect or drop off resources. Time spent driving to the dock, charging, and returning is pure overhead and should be minimized. The physical location of the dock can greatly effect the performance of such a system. Placing the dock too far from the worksite will increase travel time; time that could have been spent working. In extreme cases, the robot may run out of energy before even reaching the worksite, or may not have enough energy to work then return to charge.

We might choose therefore to place the dock at a worksite - either at a pick-up or drop-off point, or somewhere along the path between them. However, placing the dock too close

to the worksite (i.e. anywhere along the trajectory of a working robot) will increase spatial interference among robots, decreasing the amount of work done. Consider that the work/charge time ratio is often around 2/1, 1/1 or even 1/2, so robots spend considerable time charging. A stationary, charging robot is a large obstacle for the remaining robots. If the dock is shared, and multiple robots are queued up waiting to charge, the interference is exacerbated. It is likely that the optimum dock placement will be close to the current worksite, but not so close as to cause significant interference. The fact that interference is often a complex dynamic feedback process could make identifying the best location in advance infeasible.

Consider the location of your most frequented lunch spot. It is likely to be no more than a few minutes from your laboratory, so that the travel time is not significant, yet far enough that the lunchtime line-up does not interfere with access to your lab space. The lunch spot is shared with your colleagues on campus, so it is placed where it can meet these criteria for most of the local population.

Some animals manipulate the dynamic availability of their food sources by caching food. Animals are thought to cache for many different reasons; however, the male MacGregor's Bowerbird caches fruit near his bower during the mating season in order to maximize time spent on courtship (his work) [69].

This paper investigates the effect of placement of charging stations, demonstrating the effects of interference and non-stationary tasks. We propose a new approach whereby the dock is itself an autonomous robot that attempts to incrementally improve its location whenever useful information is available. We show simulation results from a simple dock controller that uses only local information to adapt to nonstationary tasks and spatial interference, and thus improves overall task performance compared to a static dock. The controller needs no information about the worker robot's task, number or locations.

This concept is related to our earlier proposal of a 'tanker' robot that visited workers at their static worksites [70]; however in this work the aim of the dock is to find a good place to remain still, rather than visiting each worker repeatedly. The long term motivation of both methods is to increase the overall energy efficiency of the system - though this is not addressed directly in this paper - and we suggest that these two alternative strategies may be suitable in different scenarios.

To coordinate multiple robots recharging at a single dock, a basic but effective queueing algorithm requiring minimal sensing between robots and no communication with the dock is also presented.

6.3 RELATED WORK

Charging stations are common among mobile robots [71] [72] [73]; however, they are constrained by the availability of a wall outlet. When a robot's energy falls below a given threshold, the robot visits the charging station to recharge before doing any work. There are different approaches to determining the threshold. A naive and easy to implement approach is to use a static threshold. Another approach is to calculate if enough energy is available to travel to the worksite, work, and then travel to the charging station, otherwise recharge first and then do some work. [74]

The stationary solar powered charging stations proposed in [75] uses sound to communicate with a team of robots. The characteristics of sound and the environment are used as an analog computer to select the most appropriate charging station. The static locations of the charging stations bound the size of the worksite which robots can travel in.

In [76] robots are autonomously reconfigured with portable tools or 'effectors' with help from a standardized mounting system. The robot is capable of transporting these tools to different worksites. A charging station is also presented with the same mounting system; however, it is assumed to be stationary as it is requires a wall outlet.

A docking station capable of transporting, deploying, and coordinating a team of heterogeneous robots is presented in [77]. To facilitate long-term tasks, the docking station is capable of recharging the worker robots. The paper presents an energy efficient way of recharging robots by minimizing a cost function related to the Euclidean distance between the docking station and all worker robots.

A mobile 'tanker' robot is described in [70], which is used to actively locate and recharge worker robots. Further work in [78] shows that the problem of finding the most energy efficient path for a tanker robot to rendezvous with a team of heterogeneous robots is NP-hard.

Spatial interference is used as a performance benchmark in [79]. The measurement of interference is used as an evaluative tool while designing multi-agent controller code. Several puck foraging techniques are designed and benchmarked. Further work in [80] has studied the correlation between spatial density and interference. Territorial division is presented as a means to evenly distribute the spatial density of robots, thus minimizing interference. However, territorial division may not be appropriate when all robots must share a common charging station.

The issue of interference around a charging station is discussed in [81]. When a robot fails to dock with a charging station which is already in use, rather than wait, the robot enters into a random wander mode for a short period before attempting to charge again. This costly behavior reduces robot density commonly found at a charging station.

6.4 System Structure

Consider a team of one or more mobile worker robots with a finite energy supply. To prolong operations, worker robots dock with a charging station to recharge once their energy level has dropped below a static threshold.

6.4.1 Dock Placement

A single worker robot is placed in a 16 x 16 meter arena containing no obstacles. The robot must transport 200 pucks, one at a time, from a source, located in the bottom-left corner at $(-7, -7)$ to a sink located in the top-right corner at $(7, 7)$. The dock is flush with the ground and does not cause any interference by itself when not in use.

Where should the dock be placed in order to minimize the time taken to deliver the pucks? One possible approach is to minimize the Euclidean distance between the worksite and the dock. Given a path function $f(t)$ which returns the location of a robot's path at a given time t, and the position of a dock p, we need to minimize $M(p)$ in order to find p^*, the optimal location for a dock.

$$M(p) = \int_0^\infty \|f(t) - p\|\, dx.$$

$$p^* = argmin(M(p))$$

For this particular task, the robot's path function, $f(t)$ is the shortest path from the source to the sink: $y = x$ from $(-7, -7)$ to $(7, 7)$. $M(p)$ can be rewritten as:

$$M(p) = \int_{-7}^{7} \sqrt{(x - p_x)^2 + (x - p_y)^2}\, dx. \tag{6.1}$$

Equation (6.1), which is plotted in Fig. 6.1, is minimized when the position of the dock, p^*, is set to $(0, 0)$ - directly between the source and the sink.

To evaluate the model, 225 (15×15) experiments are simulated in the well known Stage robot simulator. [3] where a dock is placed at each position (x, y) for all integer values of x and y where $-7 \leq x \leq 7$ and $-7 \leq y \leq 7$. The time taken to deliver 200 pucks is plotted

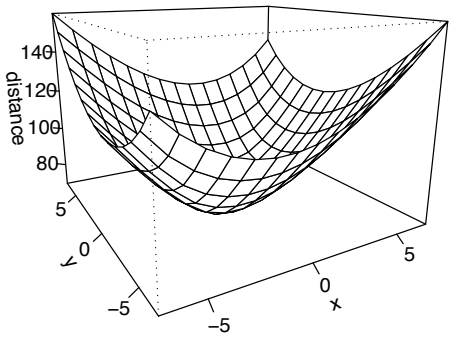


Figure 6.1: Euclidean distance can be used to model the optimal location for a fixed dock.

in Fig. 6.2(a). This is a very different picture than predicted by the model. There are two possible explanations for this. (i) Instead of using a fixed energy level as the charging threshold, the amount of energy required to deliver a puck and return to the dock is used as a dynamic threshold to prevent the robot from traveling to the sink only to have to backtrack before completing the work. Thus placing the dock anywhere along the line $y = x$ will optimize performance. However, the experiments in this paper do not use a dynamic threshold, so a different explanation is required. (ii) In this experiment, robots will always leave the dock with the same amount of energy, and energy is used at a constant rate regardless of terrain. Therefore, each robot will always drop below the static energy threshold after traveling some constant distance from the dock. Equation (6.1) assumes the probability of a robot running out of energy anywhere on the $y = x$ line is uniformly distributed; however, this is not the case as the robot always leaves the dock fully charged, and will always run out after traveling x meters.

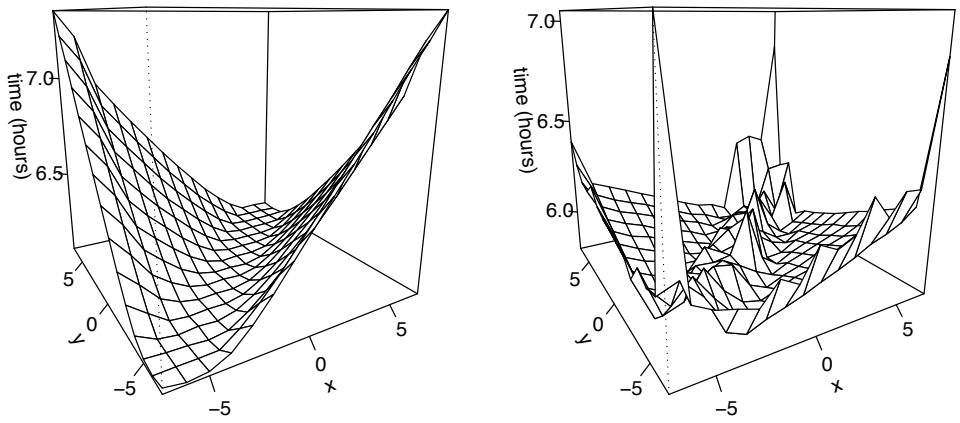
Adding a second worker robot to the system will introduce spatial interference. Fig. 6.2(b) shows that $y = x$ is no longer an optimal solution. The reason is due to an increase in interference caused by robots recharging along the shortest path. The optimal position for the dock is parallel to the $y = x$ line. This is analogous to placing gas stations next to a highway rather than in the middle of the highway.

Since minimizing (6.1) is not an appropriate solution, and creating a new model incorporating interference for any given task is difficult, if not impossible, an autonomously controlled adaptive dock is presented.

6.4.2 An Adaptive Dock

Selecting a dock that is not directly attached to a power grid, allows it to be repositioned anywhere. The dock might be powered by solar panels, a gas generator, or a large battery. In the two later cases, as robots recharge, the dock's energy supply is diminished and must be resupplied either by a human, or a specialized tanker robot. The research in this paper assumes the dock has a sufficiently large energy reserve to last throughout the simulated experiments without needing to be resupplied.

The dock may or may not have actuators capable of moving the dock throughout the environment. If it is the case that it can move, then it can be treated similar to a tanker robot. In the case where it cannot be moved by itself, then it could be moved by a service robot.



(a) The time for a single robot to deliver 200 pucks from a source at $(-7, -7)$ to a sink at $(7, 7)$ is plotted against possible locations for a fixed dock. Placing the dock between the source and the sink achieves the best performance.

(b) When two robots are used, placing the dock along the shortest path increases spatial interference, and results in less performance.

Figure 6.2: The position of a fixed dock effects the time to deliver 200 pucks from a source at $(-7, -7)$ to a sink at $(7, 7)$

The dock requires short range sensors capable of detecting approaching robots, and the angle which they are coming from. It must also be able to detect if an approaching robot requires recharging, or just experiencing interference. The sensors need not sense the distance of the approaching robot, but merely the existence. No additional information is shared - no global map, no global coordinate system.

When a robot is first detected at angle α, the value is stored in the 'rechargers' set R[1]. Once the size of R is sufficiently large, a vector is summed:

$$\vec{v} = \sum_{\alpha \in R} < \cos(\alpha), \sin(\alpha) > \tag{6.2}$$

The size of $\vec{v}$ provides an indication of how well positioned the dock is. A small size indicates robots are approaching the dock equally from all sides. A larger size indicates the majority of the robots are coming from the direction pointed to by $\vec{v}$; however, no notion of distance is captured by the vector. Once the sample size of recharging robots has exceeded a threshold, as determined by the controller code in algorithm 3, then the dock is moved in the specified direction.

6.4.3 Minimizing Interference in the Worksite

As the dock moves towards the worksite of the robots, interference is likely to increase. At some point there is a balance between minimizing travel time to and from the dock, and minimizing the amount of interference created. Whenever the dock senses a robot, at angle α, which is not actively seeking the dock, then this robot is considered as interference, and α is recorded in the interferers set I.

[1] In the experiments, the array used would only store the most recent 10 values.

$K_1 \leftarrow 5$ //sample threshold
$K_2 \leftarrow \frac{\pi}{2K_1}$
$\vec{last\_v} \leftarrow 0$
loop
  calculate $\vec{v}$ from (6.2)
  if $|R| \geq K_1$ **then**
    move dock to $\vec{v}$
    clear R
    $\vec{last\_v} \leftarrow \vec{v}$
  else
    if $|R| > 0$ and $\vec{last\_v} \neq 0$) **then**
      $\alpha \leftarrow \arccos(\vec{last\_v} \cdot \vec{v})$ //angle between vectors
      if $\alpha \leq K_2 * |R|$ **then**
        move dock to $\vec{v}$
        clear R
        $\vec{last\_v} \leftarrow \vec{v}$
      end if
    end if
  end if
end loop

Algorithm 3: Adaptive Dock Controller

Algorithm 3 is modified to use a new vector $\vec{v\prime}$ as input.

$$\vec{v\prime} = \vec{v} - k \sum_{\alpha \in I} w < \cos(\alpha), \sin(\alpha) > \tag{6.3}$$

Where k is a constant used to control the balance of distance minimized versus acceptable interference allowed.

6.4.4 Charging Station Sensors

To facilitate sensing between robots and a dock, one approach could be to add an array of photodetectors around the dock. An LED on the robot is used to indicate the charging state of the robot: solid for recharging required, or blinking for normal mode. By adjusting the frequency of the blinking light, it is possible to indicate the level of interference experienced.

6.4.5 Dock Queues

A queue mechanism is used to control access to the dock. The algorithm presented does not use any communication, but simply relies on a photodetector mounted on the front of each robot to sense other robots recharging state.

When a robot enters the recharge-needed state, the LED is set to a solid light, and the robot seeks the dock. If along the way, it senses another robot that also has a solid LED (e.g. needs to recharge), it must stop and maintain a fixed distance from that robot. For example, if a robot is currently recharging at the dock, and another robot approaches, it will sense a solid LED and must wait at some fixed distance. Once the robot has finished recharging, the recharging LED is turned off (or set to blink), and the next closest robot will be able to charge. In our simulations, a photodetector sensitivity of 2 meters, and sensing angle of 45 degrees was emulated. Setting the sensitivity to a large distance, limits the robot density, thus reducing traffic jams. There are however, a few special cases which must be addressed:

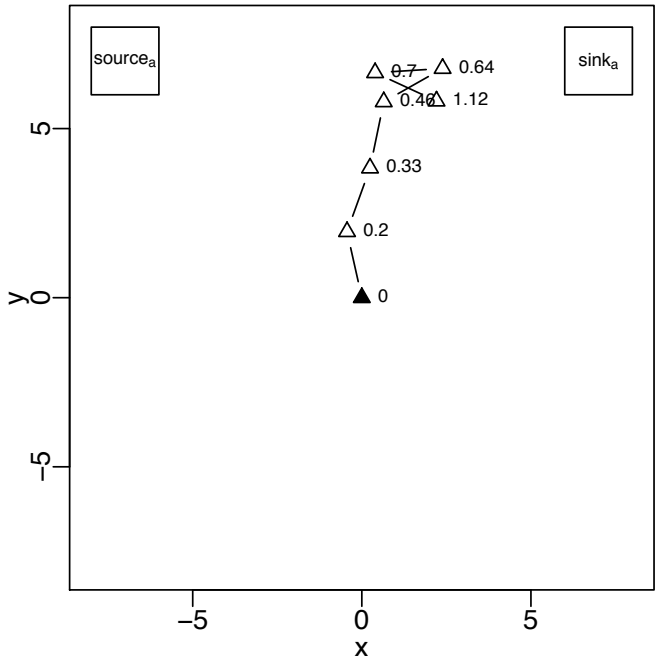


Figure 6.3: Trajectory of an adaptive dock during a trial. Each triangle represents the location of the charging station at the given time in hours. The black triangle represents the initial placement.

Simultaneously approaching from opposite sides

If two robots approach the dock simultaneously from opposite sides, then a deadlock will occur since both robots will see each other. To address this issue, an exception must be created to allow a robot to approach the dock if it is closer than the detected robot. In cases where two robots approach the dock at the exact time, a tie breaker must be implemented.

Cyclical deadlock among robots

Given enough robots, it would be possible to construct a cyclical placement of worker robots, where $robot_a$ sees $robot_b$ see $robot_c$ sees ... sees $robot_a$. In this case adding a stochastic feature to the queueing algorithm will break the cycle over time. However, by limiting the number of worker robots, and the sensing width of the photodetector, this issue was never encountered in the experiments.

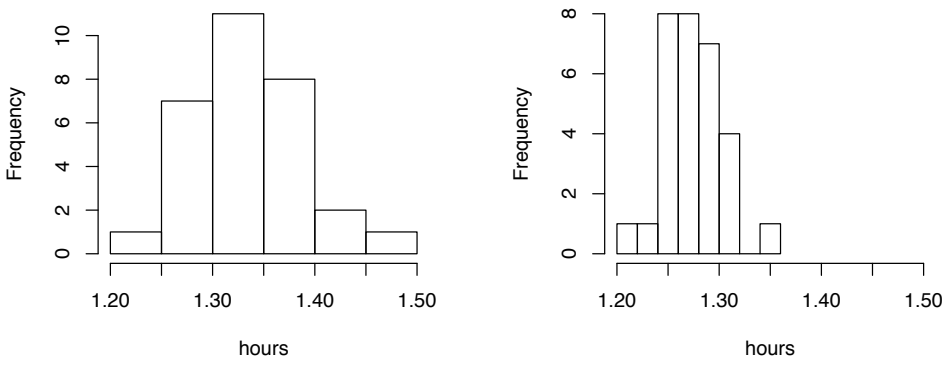
6.5 SIMULATION AND RESULTS

To evaluate the adaptive dock controller, an experiment is simulated in which a single robot transports 50 pucks from a source (located at $(-7, 7)$) to a sink (located at $(7, 7)$). The optimal placement for a fixed dock is anywhere between the source and the sink along the line $y = 7$. Three experiments are run: one with a fixed dock at (1,7), another in the middle of the arena at (1,0), and an adaptive dock initially placed at (0,0). Fig. 6.3 shows the trajectory of the adaptive dock during the experiment.

The performance of the adaptive dock should fall between the performance of the two fixed docks. Table 6.1 shows the numerical results for the simulation of 30 trials. The values show that the adaptive dock moves to a better location.

Dock Position	Mean Hours	Standard Deviation
(1,0)	2.22	< 0.01
(1,7)	1.96	< 0.01
adaptive dock	2.02	0.01

Table 6.1: time to deliver 50 pucks from $source_a$ to $sink_a$



(a) interference-awareness disabled

(b) interference-awareness enabled

Figure 6.4: Histograms of task completion time where interference-aware adaptive behavior is either enabled or disabled.

6.5.1 Dealing with interference

Repeating the experiment with a team of two robots introduces interference to the system. To evaluate the controller's ability to adapt to interference, 60 experiments are simulated, 30 with interference adaptiveness enabled, and 30 with it disabled. Fig. 6.4 indicates a slight improvement in performance in certain trials; however, more research is required to correctly deal with interference.

6.5.2 Dynamic Tasks

The worksite of tasks can change over time. Consider an extension to the previous experiment, where a second source and sink are added. Once the robot has delivered 50 pucks from $source_a$ to $sink_a$, a second set of 50 pucks must be delivered from the new source, $source_b$, to the new sink, $sink_b$ as illustrated in Fig. 6.5(a). The optimal location for a fixed dock, as determined by exhaustive search on a 1m grid, is $(1, 0)$. Since this task is really composed of two different sub-tasks, a smarter strategy is to locate a dock between $source_a$ and $sink_a$ for the first set of pucks, and then then to move the dock anywhere between $source_b$ and $sink_b$ for the second set of pucks.[2] These optimal locations for a fixed dock are indicated in Fig. 6.5.

Since this experiment is an extension to the original experiment, the results for the first set of pucks are the same. The second sub-task is much like the first sub-task: the fixed dock

[2]If a task is decomposed into an infinite number of sub-tasks, then the ideal solution is to physically attach the dock station to the robot, but this is not very interesting.

placed along the line $y = -7$ performs best, followed by the adaptive dock, followed by the fixed dock in the center of the arena. However, when these two sub-tasks are combined, the adaptive dock outperforms the fixed dock positioned in the center. Fig. 6.6 verifies that the adaptive dock adapts to the different worksites as was hypothesized. Numerical results are recorded in table 6.2.

Dock Position	Mean Hours	Standard Deviation
(1,0)	4.44	< 0.01
(1,7)	5.66	< 0.01
(1,-7)	5.63	< 0.01
adaptive dock	4.23	0.03

Table 6.2: Time to deliver 100 pucks

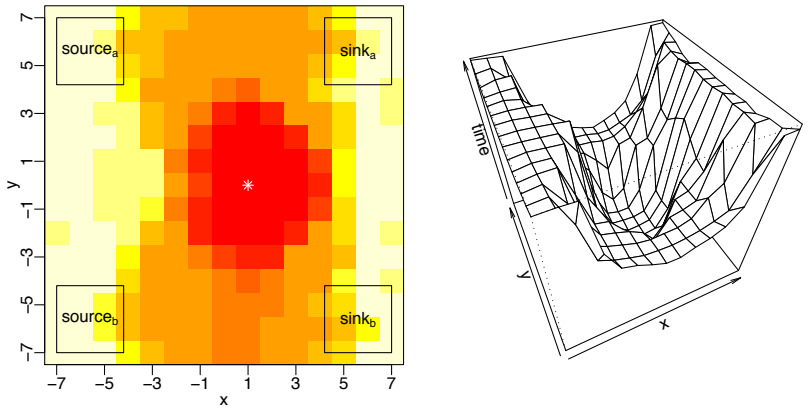
6.5.3 Oscillation Problem

The adaptive dock's controller is susceptible to oscillation, as seen in Fig. 6.3 and 6.6. Adding thresholds to the size of the summed vector from (6.2) managed to stop some oscillations; however, in most cases, it simply slowed down the frequency. One reason for the oscillation is due to the structure of the experiment; using a static recharge level, coupled with the fact that every trip between the source and sink is a static distance, the robot will always enter the seek-dock-to-recharge state at the same location. As the dock moves closer to the robot, it will simply shift the location where the robot enters the seek-dock-to-recharge state. However, that state will always be a constant distance relative to the dock's location.

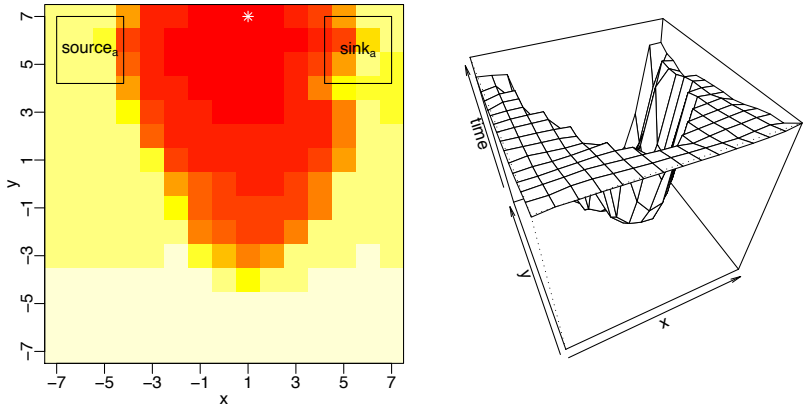
6.6 CONCLUSIONS AND FUTURE WORK

Tests so far indicate that an adaptive dock may be useful in cases where tasks can be decomposed into distinct sub-tasks. The dock would also be appropriate for nomadic tasks where exploring terrain is limited by the placement of a fixed docks. Future work will determine the feasibility of using such an adaptive dock in a real world environment with rooms, corridors and obstacles. In this case the adaptive dock must be capable of positioning itself in a sufficiently large open area in order to minimize spatial interference.

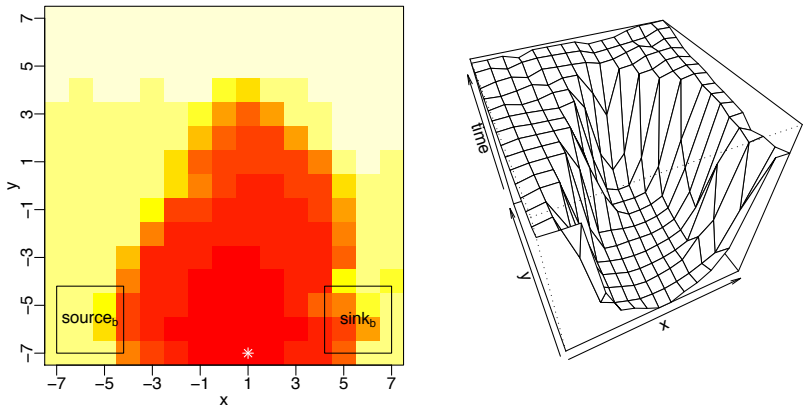
Future work includes simulating a service robot capable of resupplying and managing a collection of adaptive docks. In addition to optimizing the task completion time, a more accurate energy model must first be developed before we can minimize energy consumption incurred by the movement of a mobile dock.



(a) Time to deliver 100 pucks



(b) Time to deliver the first set of 50 pucks from top left to top right



(c) Time to deliver the second set of 50 pucks from bottom left to bottom right

Figure 6.5: Time for a single robot to deliver two sets of pucks. Positioning a fixed dock at a good central location along the most travelled path delivers pucks quicker. The asterisk indicates the optimal positions for the fixed docks.

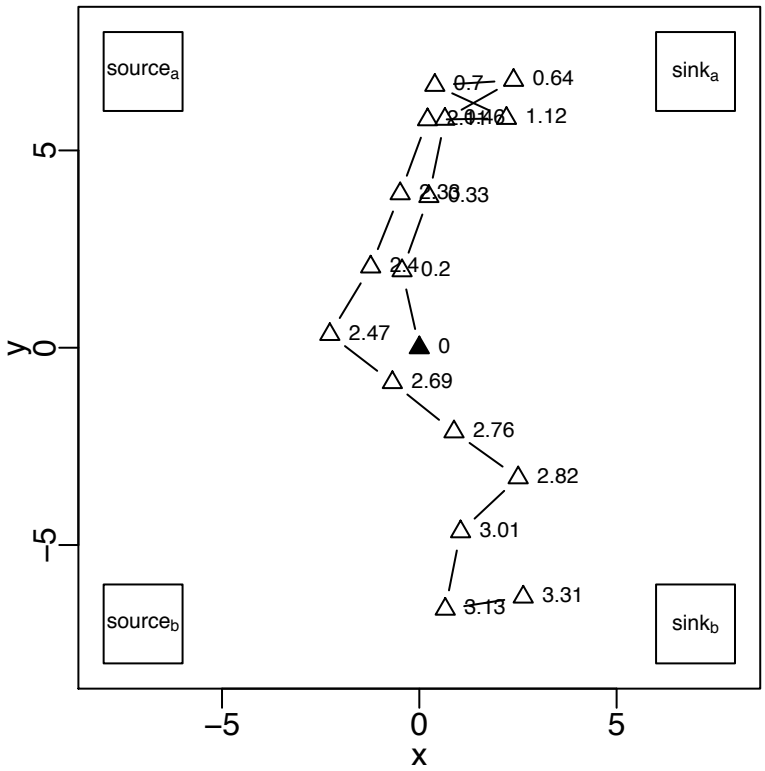


Figure 6.6: Trajectory or an adaptive dock during the delivery of 100 pucks. Each triangle represents the location of the charging station at the given time in hours. The black triangle represents the initial placement.

Chapter 7

Robot Task Switching under Diminishing Returns

Jens Wawerla and Richard T. Vaughan [9]

7.1 Abstract

We investigate the problem of a robot maximizing its long-term average rate of return on work. We present a means to obtain an estimate of the instantaneous rate of return when work is rewarded in discrete atoms, and a method that uses this to recursively maximize the long-term average return when work is available in localized patches, each with locally diminishing returns. We examine a puck-foraging scenario, and test our method in simulation under a variety of conditions. However, the analysis and approach applies to the general case.

7.2 Introduction

The purpose of a robot is to perform work [82] and thus earn some reward that justifies its human owner's investment. Usually more work gives more reward, but the instantaneous rate of reward may either be independent of the amount of work done previously, e.g. I sell apples at $1 each, no matter how long my shop has been open; or it may be dependent, e.g. picking the last few apples from a tree is more time-consuming than the first few apples. This latter 'diminishing returns' is characteristic of many robot foraging tasks, such as mining, de-mining, and collecting fungus or trash.

In a previous paper [40] we investigated optimal task switching between heterogeneous tasks with constant reward gain rate. In this paper we analyze robot behaviour for homogeneous tasks that exhibit diminishing returns. We use foraging as an example task, but our analysis and approach applies to any tasks subject to local diminishing returns.

Foraging is well-studied in behavioural ecology [83, 84] and robotics. Liu [85] and Ulam [86] examine dynamic allocation of the optimal number of workers to a given foraging task. Østergaard [87], Shell [88] and Lein [89] explore methods to reduce interference between foraging robots by separating them in space in order to improve the system's performance.

Almost all the previous work examines role allocation and interference reduction in collaborative multi-robot systems. It examines *central place foraging* [84], where foraged items are delivered to a single privileged location, e.g. a nest. Very little work has been done on high-performance solitary foraging in robots, though this is well-studied in animals.

An exception, and the most similar previous work, is by Andrews et al. [90] which explores the use the Marginal-Value Theorem (see Section 8.2.1) for task switching. The experiments

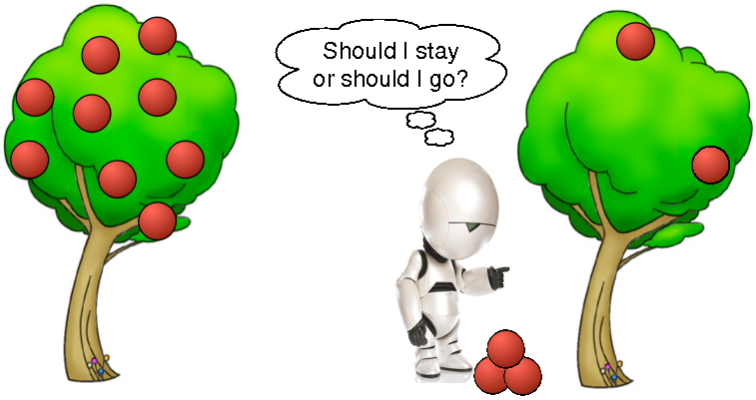


Figure 7.1: The robot must maximize global collection rate in an environment of multiple work sites, each with locally diminishing returns.

in this paper are grounded in a complete low-level robot controller in a sensorimotor simulation rather than the mathematical models of [90]. Another important technical difference is mentioned below.

In this paper we consider a single robot foraging for atomic units of resources that are consumed (and reward obtained) as they are encountered, instead of delivered centrally. This models self-feeding, for example. Crucially, units of resource are not distributed uniformly, but exist in regions of locally high density known in the biology literature as *patches*. Danchin [91] defines a patch as "an homogeneous resource containing area (or part of habitat) separated from others by areas containing little or no resources". The advantage of patches is two-fold: (1) patches give the forager *a priori* information about the likelihood of finding resources. For example we expect to find valuable apples only on apple trees, and not on the ground; (2) patches reduce the complexity of the decision process: instead of making decisions about each apple, it is sufficient to only make decisions about each patch. As the number of patches is usually significantly smaller than the number of resource units, we can expect reduced complexity of decision-making.

However, considering patches instead of atomic units of resource introduces the issue of the reward rate per patch, usually called *patch quality*. This may not be known *a priori*, and may require the forager to forage in the patch for some time to determine a good estimate of the patch quality. This sampling cost inevitably harms overall performance.

The task illustrated in Fig. 7.1 has the properties of interest. A robot collects apples in an orchard, and is rewarded as it collects each apple. As a tree (patch) is depleted of apples, the marginal cost of picking the next apple increases. How does the robot decide to abandon the current tree and move to the next, such that the overall rate of apple collection is maximized?

7.2.1 Marginal-Value Theorem

Charnov [92, 93] proposed the Marginal-Value Theorem (MVT) to model the decision faced by an animal foraging in a patchy environment. The MVT says a rate-maximizing forager should leave the current patch once the marginal gain rate equals the long-term average rate of the habitat. Charnov's mathematical derivation gives two results: (1) the marginal rate of leaving must be the same for all patches in a habitat; and (2) as the cost to switch patches increases the forager should exploit each patch longer.

The simplicity of the rule makes it very appealing, but the theorem and its validity has also been widely and controversially discussed, for example by [94, 95, 83]. It has been argued that the rule is circular in that the long-term average needs to be known in order to select the best

leaving threshold. But the choice of the leaving threshold influences the long-term average. We think the rule is clearly circular, in fact we will exploit this circular dependency between leaving threshold and long-term average gain rate in our robot controller. Another important issue often pointed out is that the MVT uses the instantaneous gain rate. But how does the forager measure the instantaneous rate when resources are found in discrete lumps? We will show one possible solution below.

The MVT is not the only patch-leaving rule proposed by behavioural ecologists. A number of similar rules have been discussed by [96], differing in the parameter used to make the patch-leaving decision. Candidates are (1) the total patch residence time; (2) the time since the last item was encountered; and (3) the number of collected items. We favour the instantaneous gain rate because it has some important practical advantages over the other proposed parameters. For example, if the forager encounters an exceptionally low quality (e.g. empty) patch, a patch-leaving policy based on instantaneous gain rate would leave the patch quickly (the desired behaviour), where a constant time policy would stay for the whole length of the residence interval and a policy based on the number of collected items would cause the forager to never leave the patch.

Below we present the robot control policy and demonstrate its effectiveness in series of simulation experiments.

7.3 Robot Controller

We use a generic mobile robot model in the well known simulator Stage[97]. It is equipped with a short-range colour blob tracker to sense pucks, our unit of resource. The robot knows (or equivalently can detect) the boundaries of puck patches. Patches are 620 times the size of the robot, and contain 10, 30 or 50 uniform randomly placed pucks. A minimum distance between pucks is enforced to avoid overlap. To exploit a patch, the foraging robot can use one of two foraging policies:

1. Under the **random foraging policy** the robot drives straight until it comes to the patch boundary, where it chooses a new heading that brings it back into the patch, at random. When pucks are detected, the robot servos towards the closest puck and collects it.

2. Under the **systematic foraging policy** the robot employs a regular square-wave-like search pattern from one side of the patch to the other side. The distance between legs in the pattern is small enough to guarantee that the whole patch is searched.

These policies exhibit different reward dynamics. To illustrate this we tested each policy on 100 patches of 50 uniform random placed pucks each. Fig. 7.2 shows the average time required to collect a certain number of pucks under the two policies. The random foraging policy suffers from diminishing returns, since over time it is increasingly more likely to re-visit previously cleared, now unproductive, parts of the patch. The systematic forager has an approximately constant collection rate. While we might prefer the systematic forager because of this property, it may be much more costly to implement than the random forager. Real-world low-cost robots like the iRobot Roomba use a randomized method that suffers from diminishing returns.

The point here is not to choose the best policy, but to illustrate that low-level robot control policy can fundamentally influence the nature of the overall optimization task, and to show that single-robot random foraging is subject to diminishing returns and thus provides a suitable model for all tasks of this domain.

Given this randomized foraging policy, the robot needs to determine when to abandon the current patch and move on. As suggested by Charnov's MVT [92], our robot abandons a patch once the instantaneous gain rate drops below a threshold. To do this we must (1) determine the instantaneous gain rate of atomic items (pucks) encountered at every time-step and (2) select a leaving threshold that maximizes the long-term average gain rate.

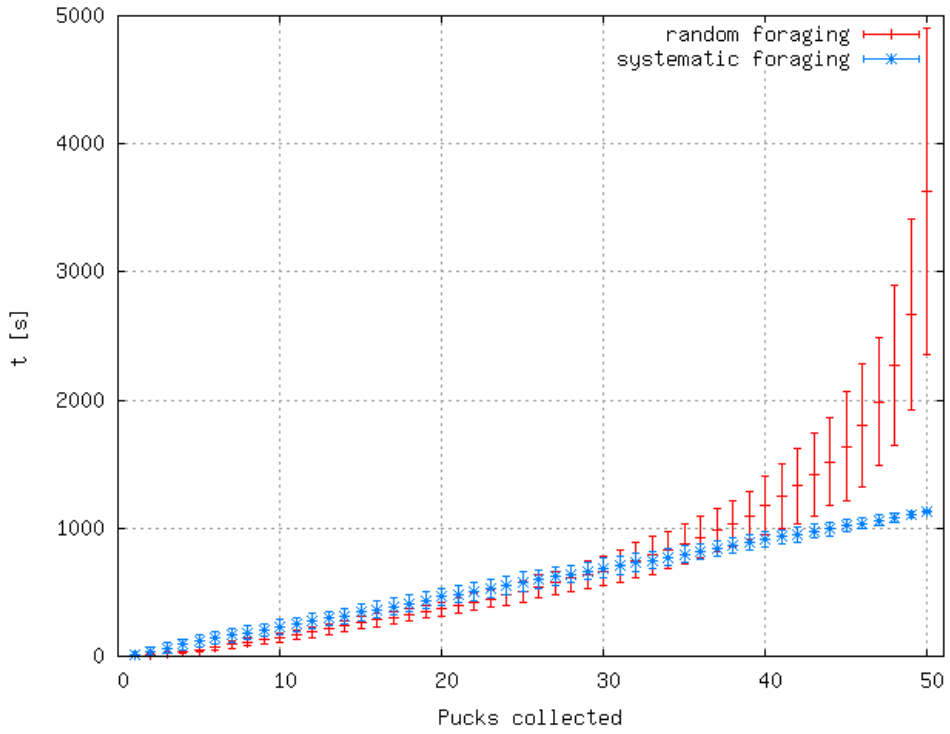


Figure 7.2: Time required to collect all 50 pucks in a patch using random search and systematic search.

7.3.1 Instantaneous Gain Rate

Measuring the instantaneous rate of atomic events is impossible. [90] suggest calculating the slope using the current and the previous two gain function values. This approach may work for continuous gain function, but fails in situations where the gain function is non-continuous and stochastic such as in unit-based foraging in uniform random patches. So we resort to an alternative representation. We use the expected value of a beta distribution over time-steps in which the robot found a puck and those in which it did not. Equation 7.1 gives this expected value of the beta distribution which we use as a proxy for the instantaneous rate $\hat{\lambda}_i(t)$.

$$\hat{\lambda}_i(t) = \frac{p_i(t)}{p_i(t) + q_i(t)} \tag{7.1}$$

Where i refers to the i-th patch, t is the amount of time spent in a patch, $p_i(t)$ is the number of time-steps in which the robot collected a puck and $q_i(t)$ is the number of time-steps during which the robot did not collect any pucks. This method has an interesting problem: until the first puck is found, the number of time-steps in which the robot collected pucks is zero ($p_i = 0$), and thus our estimate of the rate is zero ($\lambda_i = 0$). Hence the robot will immediately leave the patch. We avoid this by initializing $p_i(0)$ and $q_i(0)$ with a prior ϕ_p and ϕ_q respectively. These priors represent the robot's expectation of the initial gain rate of a patch. They can either be set to an environmental constant (if known) or they can be determined at run time by experience from the previous patch. The ratio of the priors represents the expected initial gain rate, the magnitude of each is an indication of the forager's confidence in these priors. The higher the magnitude the more experience is required to modify λ, i.e. change the robot's belief about the prevailing rate.

Algorithm:forage(θ)
init ϕ_p, ϕ_q
$t = 0$
$p(0) = \phi_p$
$q(0) = \phi_q$
$\hat{\lambda}_{filt}(0) = \frac{p(0)}{p(0)+q(0)}$
repeat
    $t = t + 1$
    randomly forage for one time-step
    if *puck found* **then**
        $p(t) = p(t-1) + 1$
    else
        $q(t) = q(t-1) + 1$
    end
    $\hat{\lambda}(t) = \frac{p(t)}{p(t)+q(t)}$
    $\hat{\lambda}_{filt}(t) = \hat{\lambda}_{filt}(t-1) + k_1(\hat{\lambda}(t) - \hat{\lambda}_{filt}(t-1))$
until $\hat{\lambda}_{filt}(t) < \theta$
return (t, p)

Algorithm 4: Forage in the current patch until a proxy for the instantaneous rate $\lambda(t)$ drops below the threshold θ

Because puck encounter is a random and infrequent process, the value of $\hat{\lambda}_i(t)$ is very variable in practise, particularly for small values of $p_i(t)$ and $q_i(t)$. We apply a low-pass filter to smooth this slightly. These steps are combined into Algorithm 4.

7.3.2 Patch-Leaving Threshold

Recall that the MVT predicts that a forager should leave a patch once λ drops to the environment's global average. In the general case it is impossible for the robot to know this true global average rate. All the robot can measure directly is the long-term average gain rate it experiences: a value which depends on the robot's past behaviour. The experienced average gain rate $\mu(\theta)$ for foraging in n patches is given by

$$\mu(\theta) = \frac{\sum_{i=1}^{n} g_i(\theta)}{\sum_{i=1}^{n}(t_i + \tau_i)} \tag{7.2}$$

where $g_i(\theta)$ is the gain function that gives the total number of pucks collected in patch i when the patch is abandoned according to Algorithm 4, t_i is the time spent in the i-th patch, and τ_i is the patch switching duration. The objective of the forager is to maximize μ by selecting θ.

$$\theta^* = \text{argmax}(\mu(\theta)) \tag{7.3}$$

Unfortunately $g_i(\theta)$ is unknown and difficult to obtain. The function depends precisely on the robot's sensorimotor interaction with the environment, the patch quality and the distribution of pucks in the patch. Hence a closed–form solution of eq. 7.3 is not obtainable. Fig. 7.3 shows the long-term average gain rates over a range of values of θ for different environmental conditions. From these graphs we can see that the observed average gain rate varies widely under different circumstances. The variance in average gain rate observed is high, but error bars are omitted for clarity.

Since we only have an approximation of the instantaneous gain rate, we cannot, as the MVT requires, leave a patch once this rate drops to the long-term average. Instead we resort

1 **Algorithm:**adaptive()
2 **init** θ_{min},θ_{max}, N, k_1, k_2, k_3
3 $\delta = \frac{\theta_{max}-\theta_{min}}{N}$
4 **for** *i=1 to* ∞ **do**
5 **if** $i \leq N$ **then**
6 $a = i$
7 $\theta = i \cdot \delta$
8 **else**
9 draw a with probability $\propto \frac{e^{w(a)/k_2}}{\sum_{j=1}^{N} e^{w(j)/k_2}}$
10 draw θ uniform randomly from $[\Theta(a) - k_3\delta, \Theta(a) + k_3\delta]$
11 **end**
12 $(t, p) =$ forage(θ)
13 $\mu = \frac{p}{t+\tau}$
14 **if** $i \leq N$ **then**
15 $\Theta(a) = \theta$
16 $w(a) = \mu$
17 **else**
18 $\Theta(a) = \frac{w(a)\cdot\Theta(a)+\mu\cdot\theta}{w(a)+\mu(a)}$
19 $w(a) = w(a) + k_1(\mu - w(a))$
20 **end**
21 **end**

Algorithm 5: Adaptive leave rate selection

to maximizing the long-term gain rate by recursively improving the leave threshold θ based on previous experience.

In practise we found that the observations have such high variance (evident in the errorbars of Fig. 7.2), that local gradients are not very useful and thus simple gradient descent methods resulted in poor performance. A more sophisticated heuristic approach is required in our scenario.

Action-Value methods have been shown to be effective for n-armed bandit problems; maximizing the return of a discrete set of unknown process. Sutton and Barto [98] give a overview of various action-value methods and Vermorel et al. [99] give an empirical comparison. We can adapt this framework to our continuous action-space by finding a discrete set of values of θ that approximate the input-output mapping of the true gain function. We continuously refine this set by exploring θ space, while frequently using the current best estimate of θ^* to obtain good performance as we go.

Since no generally optimal method is known, we use here a combination of *softmax* and *ε-first* adapted for continuous action-spaces. A comparative analysis of the various possible algorithms is beyond the scope of this paper. The complete method is shown as Algorithm box 5. The parameters θ_{min} and θ_{max} are the minimum and maximum patch leaving thresholds. These are determined by the environment and the robot specification, e.g. the maximum travel speed. N denotes the number of bins into which the action-space is discretized. More bins potentially allow a better approximation, at the expense of longer start-up time. Constant k_1 is used to smooth the weight update, k_2 is the *temperature* for *softmax* action selection and k_3 determines how much we are willing to modify the leave rate threshold between adjacent patches.

To initialize our discrete approximation of the gain function we must fill the N bins. They can be initialized with prior expected values if available, but here we obtain them empirically with a start-up phase. For the first N patches visited we use the discretized leave rate threshold (lines 5-7 in *ε-first*). Here $1 \leq a \leq N$ denotes the bin number. $\Theta(a)$ describes the centre of bin

a and $w(a)$ its weight. For the remaining patches we select a bin using *softmax* (line 9). The leave rate threshold θ is uniform randomly chosen from an interval centred around the centre of the bin (line 10).

The robot now forages with the new leave rate threshold θ using Algorithm 4. This results in some patch residence time t and in p pucks collected. Therefore we can calculate the average gain rate for the patch μ as the number of pucks collected divided by the sum of patch residence time and patch switching time τ (line 13). Next we update the weights and the centre of the bin. For the first N patches the weight is simply the average gain rate μ. The centre of the bin is the leave rate threshold used (lines 15-16). For the remaining patches the centre of the weight is the moving average of the average gain rate and the centre of the bin is the weighted sum of the previous centre and the just explored leave rate threshold (lines 18-19), where the previous centre is weighted by the weight of the bin and the latest leave rate threshold is weighted with the average gain rate that resulted from the use of this threshold.

Summarizing the action of Algorithm 5: *Softmax* selects the best bin in a greedy fashion while it keeps exploring the other bins with probability proportional to the weight of each bin. Since the weight is an estimate of the expected yield rate, it explores higher paying areas of the action-space more frequently then lower paying areas. The discretization step might choose bin centres that are suboptimal; to over come this problem we perform a local random walk of the bin centres. Over time this will move the bin centres towards better values.

7.4 Experiments

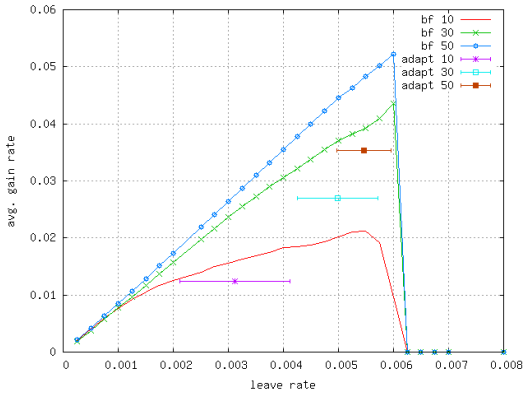
To investigate the effectiveness of our approach, we conducted a series of simulation experiments consisting of two phases (1) generate foraging data and (2) test our adaptive task (patch) switching policy on the generated data.

To generate the foraging data we used the method described in Section 7.3. For each of the three puck qualities (10, 30, 50 pucks per patch) we generated 100 samples and recorded for each time-step whether the robot collected a puck or not. Note that in this phase no patch-leaving decisions are made, we just recorded data while the robot simply collected pucks until each patch was exhausted.

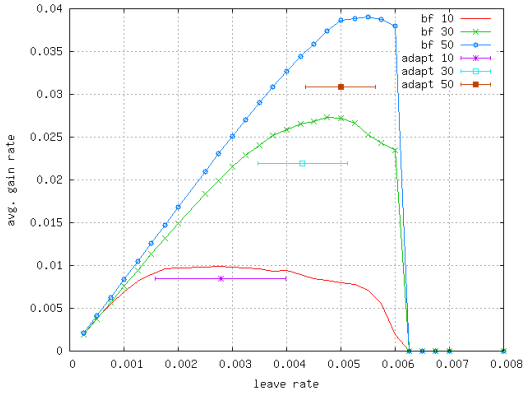
In the second phase we ran our adaptive patch switching policy on the recorded data and compared it against the best solution obtained by exhaustive search over the same data set. Decoupling the data generation from the analysis of the control policy allowed us to cancel any noise in the performance analysis caused by the random generation of the patches. It also made brute-force search for approximately optimal solutions feasible.

To obtain a benchmark with which to compare our online adaptive method, we choose a finite discrete set of rate thresholds over a range of θ between our preset maximum and minimum. For each threshold we had the robot forage in each patch until the instantaneous rate dropped to that selected leave rate threshold (Alg. 4). On leaving the patch, the robot takes some time in which no pucks are collected before arriving at the next patch: the patch–switching cost. This way we found the leave rate that maximizes the long-term average gain rate for the recorded data. Fig. 7.3 shows the long-term average gain rate over the leave rate threshold for patches of different quality and different switching costs. The prediction of the MVT that the patch residence time should increase with increasing switching cost is clearly visible. The graphs also give an idea of the search space for the adaptive algorithm. Especially low switching costs (fig. 7.3(a)) exhibit a sharp performance peak which is difficult to adapt to, in this situation it is desirable to stay on the side with the smaller slope. Patch quality is one factor that determines the environmental gain rate. The other is the average switching time between patches. To investigate the influence of the switching time τ we analyzed the system with 4 different switching times, $\tau = \{10, 100, 500, 1000\}$ seconds.[1].

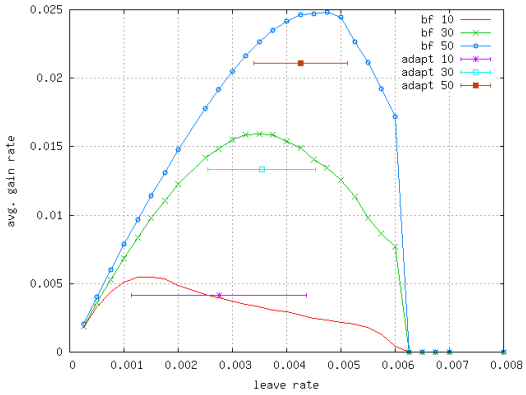
[1]The graph for switching cost 1000 sec. in Fig. 7.3 was omitted due to space constraints, but it is qualitatively similar to that with cost 500



(a) Average gain rate with patch switching time 10 sec.



(b) Average gain rate with patch switching time 100 sec.



(c) Average gain rate with patch switching time 500 sec.

Figure 7.3: Long-term average gain rate observed versus leave rate threshold for different patch switching cost and patch qualities (errorbars in y-axis omitted to improve readability). The sharp drop-off at $\tau = 0.006$ occurs because the leave threshold is too high for the robot-environment interaction, the robot leaves a patch before it encounters the first puck

Table 7.1: Adaptive patch switching and random foraging

	τ [s]							
	10		100		500		1000	
pucks	μ[%]	σ	μ[%]	σ	μ[%]	σ	μ[%]	σ
10	60.2	3.72	84.6	2.68	76.7	3.40	66.6	3.89
30	62.0	0.76	79.9	2.20	82.8	2.75	81.4	2.03
50	68.3	0.64	79.3	1.11	85.4	1.60	86.3	1.38

Table 7.2: Adaptive patch switching and random foraging under varying patch switching times with mean 100 sec

	switching cost variance							
	0		30		50		80	
pucks	μ[%]	σ	μ[%]	σ	μ[%]	σ	μ[%]	σ
10	84.6	2.68	82.6	3.30	79.5	4.79	80.0	2.97
30	79.9	2.20	80.0	1.48	79.5	2.55	80.8	3.01
50	79.3	1.11	79.5	1.16	79.9	0.90	80.9	1.18

The performance of the adaptive method is presented as mean and standard deviation of observed gain of 20 trials of 100 patches each, compared to the estimated optimum obtained by exhaustive search. All algorithm parameters required were set manually and kept constant without attempting to optimize them, except in one case. The confidence of the priors for the instantaneous gain rate were lowered for patches with 10 initial pucks, so that the estimate of the gain rate would change faster. The original values gave poor performance in this challenging scenario.

Table 7.1 shows the results of the first experiment in which we analyzed the adaptive patch switching policy for 3 different patch qualities and 4 different switching costs. The results are given in percentage of the bruteforce long-term gain rate. The small standard deviation of about 2.5% indicates that the system performs fairly consistently despite the high noise levels in the data. Compared to the brute-force method we expect a lower performance since the system, like any ϵ-greedy method, has to trade off between exploration and exploitation. Our method performs a fixed number ($N = 10$) of initial exploration trials and subsequently performs continuous exploration with a small probability. Especially during the initial exploration we expect some trials with very poor performance because the whole action-space is sampled. This does harm the overall performance.

The actual performance appears quite good: between 75 and 85% of the brute-force benchmark solution as long as the patch switching cost is not too small. The performance drops with the very small patch switching cost of 10 seconds. As Fig. 7.3(a) indicates, this is a challenging situation since the gain rate function has a very sharp peak. Another reason why this is a difficult situation is that the small switching cost requires the forager to leave the patch very early. For example the best threshold found by brute-force causes the robot to stay for only an average of 35 seconds in the 10 puck patch. Any error the adaptive method makes is relatively large compared to the small switching cost.

Nevertheless as Fig. 7.3 shows, the mean of the leaving threshold selected by the adaptive system is usually near the optimal threshold and always on the side of caution by being on the side of the smaller slope.

In a second experiment we investigated the performance of the system when the patch switching time varies from one patch to the next. For this experiment we drew patch switching

Table 7.3: Adaptive patch switching for random foraging under changing patch quality and systematic foraging

	τ [s]							
	10		100		500		1000	
pucks	μ[%]	σ	μ[%]	σ	μ[%]	σ	μ[%]	σ
30→50	82.0	2.87	92.8	3.02	90.9	2.92	90.8	1.67
50→30	80.0	2.44	90.7	2.44	91.3	1.50	89.3	1.50
10	55.2	4.54	81.2	6.32	84.4	6.50	83.4	7.03
30	80.7	3.46	80.5	3.76	79.1	3.89	79.2	2.05
50	76.5	1.80	80.1	1.49	84.9	2.64	83.7	2.79

times from a normal distribution with a mean of 100 seconds and variance of 30, 50 and 80 seconds respectively. As in the previous experiment we obtain a near-optimal threshold by brute-force search as a benchmark and compare it with the average of 20 instances of the adaptive system. Table 7.2 shows the results. The performance is around 80% for all situations with low deviation, indicating that the system is not sensitive to variance in the switching cost.

To analyze the system under changing patch quality conditions we set up two simulations in which patch quality changes from 30 pucks per patch for the first 100 patches, to 50 pucks per patch for another 100 patches and vice versa. We ran these simulations with a patch switching cost of 10, 100, 500 and 1000 seconds respectively. We compare the performance of our method with that observed when switching between the best fixed thresholds for 30 and 50 pucks obtained by brute force search. The results are shown in the top two rows of table 7.3. The data suggest that the system handles the patch quality switching well. The performance seems slightly better than in the stationary situation in table 7.1. This is probably due to a relatively longer exploitation duration: the cost of the initial 10 patch exploration is amortized over 190 patches rather than 90. Despite that fact that the bruteforce method is given the advantage of actually knowing when the patch quality switch occurs, something our method has to detect and adapt to, the system performs very well.

In a last experiment we investigated the adaptive method's performance in situation in which the forager searches each patch systematically. Recall that under this foraging policy the instantaneous gain rate is constant, that is until the patch is empty and the gain rate drops to zero. In this situation the robot has not to decide which leave rate maximizes the overall reward but only to determine when the patch is empty. To see how our system copes with this situation we compared the brute-force and the adaptive method when robots forage patches using the systematic foraging policy. As the last three rows of table 7.3 shows the system can also handle situations in with the instantaneous gain rate is given by a step-function.

7.5 Conclusion

In this paper we investigated the problem of a robot maximizing its long-term average rate of return on work. We presented a means to obtain an estimate of the instantaneous rate of return when work is rewarded in discrete atoms, and suggested one way to use this to recursively maximize the long-term average return when work is available in localized patches, each with locally diminishing returns.

We examined a puck-foraging scenario, and tested our method in simulation under a variety of conditions. However, the analysis and approach applies to the general case.

The validity and applicability of the Marginal-Value Theorem to animal behaviour is widely and controversially discussed in the behavioural ecology literature. Here we have provided evidence that the underlying idea of making task switching decisions based on thresholding the

instantaneous gain rate is a valid approach, at least for artificial systems. Whether biological systems make decisions on this principle is an open question.

Chapter 8

Online Robot Task Switching Under Diminishing Returns

Jens Wawerla and Richard T. Vaughan [100]

8.1 abstract

We investigate the task switching problem of a robot maximizing its long-term average rate of return on work performed. We propose an online method to maximize the average gain rate based on only past experience. For that we alter the formulation from optimal foraging theory and recursively include estimates of global task qualities. We demonstrate and analyze our method on a puck-foraging example. In simulation experiments under a variety of conditions we show that our method performs well compared to results obtained by brute force method using post-processed foraging data.

8.2 Introduction

Many robot applications require a robot to make task switching decisions in order to maximize its reward. Often this reward is a diminishing function of the time spent performing the task. These diminishing returns can either be caused by (i) exhausting a given task, e.g. having delivered all mail in a given building or by (ii) increasing difficulty to perform the task, e.g. it will be more and more difficult for a vacuum cleaning robot[1] to remove dirt as it cleans the floor. In fact it will be virtually impossible for a vacuum cleaning robot to remove all dirt particles and thus this task has no well defined intrinsic end point.

In both situations the robot has to decide when it is profitable to terminate the current task, pay a switching cost, and start a new task that yields higher rewards. The switching cost can come in form of an opportunity cost or an actual cost such as energy expenditure, transit toll or task acquisition cost. In other words the robot has to decide when to switch tasks in order maximize its long-term average reward rate. This decision depends on a number of factors: how good is the current task, how high is the switching cost and what is the average payoff function for tasks in the robot's environment?

In an earlier paper [101] we proposed a task switching policy based on the Marginal-Value Theorem (MVT) (see Sec. Marginal-Value Theorem). This policy required the robot to perform exploration steps in order to evaluate the average quality of the available tasks. We showed that the performance of the proposed policy was about 80% of that obtained by a near optimal policy discovered by brute force search.

[1] We assume the robot gets rewarded for the amount of dirt collected and not for time spent vacuuming.

of a brute force near-optimal method. In this paper we propose a recursive task switching policy based on locally available information only, hence no explicit exploration phase and thus no exploration/exploitation trade-off is required.

The policy is applicable to other task switching situations that exhibit diminishing returns. We choose foraging as an example task, since it is a canonical task in autonomous robotics [102]. Robot foraging often means multi-agent *central place foraging* [84], where foraged items are delivered to single privileged location. In contrast in this paper and our previous work [101] we use solitary, instant-consumption foraging in a patchy environment: a single robot immediately consumes items once they are encountered obtaining a reward without the need to deliver them to a centralized location. Items to be foraged are not distributed uniformly, but in patches defined for Behavioural Ecology as "*an homogeneous resource containing area separated from others by areas containing little or not resources*" [91].

8.2.1 Marginal-Value Theorem

In behavioural ecology the task switching problem is often discussed in terms of optimal foraging theory [83] as a patch leaving decision. In this context patches are subject to diminishing returns and thus require the forager to make decisions about changing patches. In this case the task switching cost the inter-patch travel cost. An important result of optimal foraging theory is the Marginal-Value Theorem (MVT). [103, 92] proposed the MVT to model foraging decisions made by animals. His key result is the following patch leaving rule: "*when the intake rate in any patch drops to the average rate for the habitat, the animal should move on to another patch*" [103]. As a consequence an optimal forager should exploit patches for a longer time as the inter-patch travel time increases and for a shorter time as the entire environment becomes more profitable. The simplicity of this rule makes it very appealing as a task-switching rule for robots, but the theorem and its validity has been widely and controversially discussed, for example by [94, 95, 83]. Some of these issues make an implementation of the MVT as a robot task switching policy impossible. The main problems are:

- How to measure the marginal gain rate (the derivative of the gain rate) if the reward comes in discrete lumps. [90] suggest calculating the slope of the gain function between the last gain function change and the one two changes prior. In our tests (not shown) this method proved ineffective due to the stochastic nature of puck encounter during random foraging in patches with randomly placed pucks. In previous work [101] we used the expected value of a beta distribution over time-steps in which the robot found a puck and those in which it did not, as a proxy for the instantaneous rate. While we were able to build a task switching policy around this estimated gain rate, it is not the instantaneous gain rate. Thus leaving a patch once this estimated gain rate equals the long-term average rate does not maximize the long-term gain rate.

- The true long-term average gain rate for a given environment is usually unknown to the forager: all it can know is the average gain rate it experiences. This experience is a result of the foragers behaviour, yet the MVT requires the forager to base it's patch leaving decision on the obtainable long-term average gain rate. This circular dependency necessitates that the forager explores the action space in order to find the maximum long-term average gain rate. Previously [101] we used this circular dependency and turned the foraging task into a multi-armed bandit problem and applied standard ϵ-greedy methods [98] to tackle the exploration-exploitation trade-off.

[83] summarize these problems as "*The MVT survives not as a rule for foragers to implement, but as a technique that finds the rate-maximizing rule from a known set of rules*". Since the MVT does not provide an implemetable policy, behavioural ecologists proposed other patch-leaving rules. (1) **number rule**, "leave after catching n items" [104]; (2) **fixed residence time rule** "leave after being in a patch for t time" [105]; (3) **give up time rule**

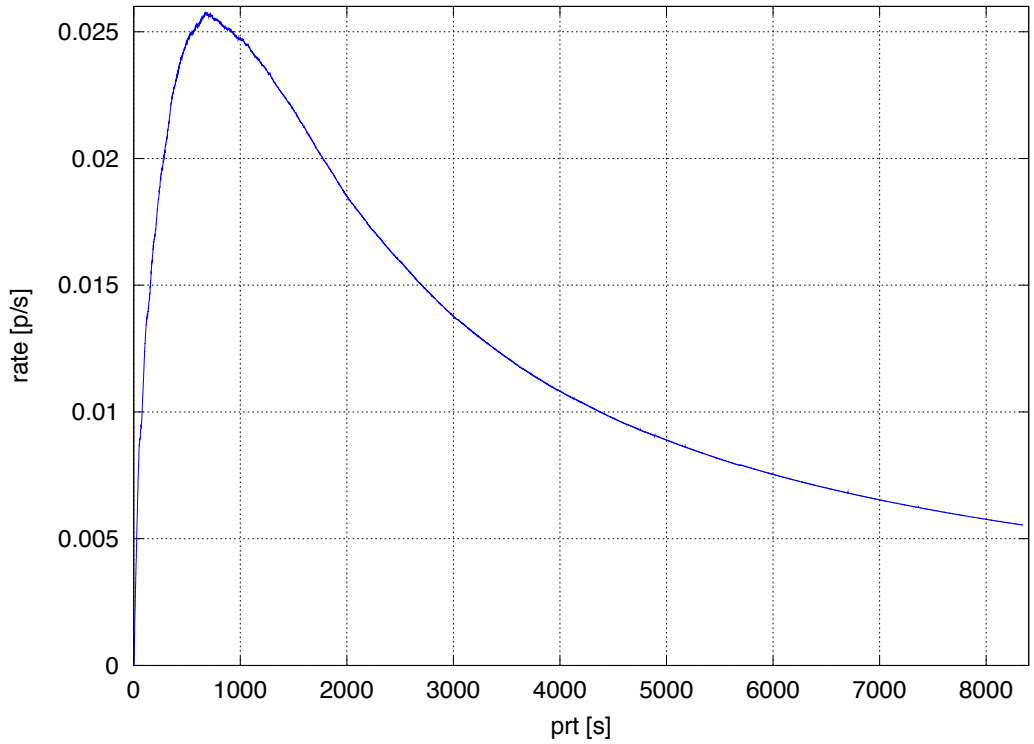


Figure 8.1: Average gain rate for a fixed patch residence time. Series of 100 patches with initially 50 pucks and a patch switching time of 500 seconds.

"leave after t time has elapsed since the last encounter" [106]; (4) **rate rule** "leave when the instantaneous intake rate drops to a critical value r" [95]. Rules 1-3 have the advantage that the decision is based on values that are easily measurable by the forager. The rate rule is an extension of the MVT in that it copes with variance in patch sub-types, but it does not address the two issues mentioned above. None of these rules address the question of how to obtain the magic number on which the decision is based.

To illustrate the difficulty of this task-switching problem we conducted a brief simulation experiment. For this experiment we generated 100 constant size patches, each with initially 50 pucks. Next we had the robot forage in each patch until it was completely exhausted. For each time step we recorded the number of pucks gained from the current patch. From the recorded data we then calculated the average long-term gain rate as a function of patch residence time. In other words we forced the robot to leave each patch in a 100 patch series after a fixed time. By sweeping over patch residence times from 10 to 8000 seconds we obtained Fig. 8.1. This graph shows the long-term gain rate for a given patch residence time for this particular patch configuration and switching cost. The curve is interesting because it shows how large an error (ie. reduction on average reward gain rate) a task-switching robot can make if switching too early or too late. It is worth pointing out that a robot is not actually able to measure this curve and exploit a patch optimally at the same time. Fortunately the robot only needs to find the maximum of the long-term gain rate and not determine the function per se.

Having described the optimization problem, in the following we present a new online adaptive solution that is grounded in the robot's perception and achieves foraging results comparable to an idealized forager that bases its decisions on global, unknowable environmental averages.

8.3 Marginal Gain Rate Task Switching

To derive the MVT [92] argued that an optimal forager should maximize

$$R = \frac{\sum \lambda_j \cdot g_j(t_j) - \tau \cdot E}{\tau + \sum \lambda_j \cdot T_j} \tag{8.1}$$

where λ_j is the proportion of visited patches that are of type j, $g_j(t_j)$ is the net gain function for a patch of type j, τ is the average inter-patch travel time, E the rate of energy expended while switching patches and t_j is the time spent in a patch of type j. The objective of a forager is to select all patch residence times t_j such that R is maximized.

Without loss of generality we ignore the energetic cost of travel $\tau \cdot E$, since it is independent of the decision variables, so Eq. 8.1 reduces to

$$R = \frac{\sum \lambda_j \cdot g_j(t_j)}{\tau + \sum \lambda_i \cdot T_j} \tag{8.2}$$

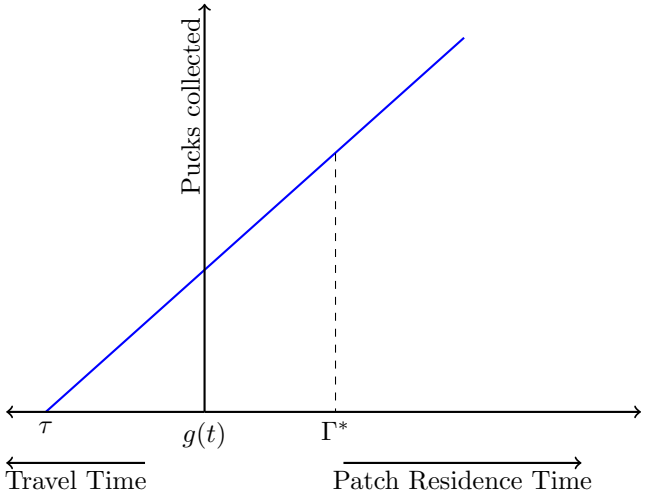


Figure 8.2: Typical MVT plot with two quantities on the abscissa: travel time increasing to the left, and patch residence time increasing to the right. The optimal patch residence time Γ^* is found by constructing a tangent to the gain function $g(t)$ that begins at the patch switching time τ on the travel time axis.

Charnov showed that R is maximized if $\frac{\partial g_j(t_j)}{\partial t_j} = R$. Graphically this is easy to do. As Fig. 8.2 shows, the optimal patch residence time T_j is found by constructing a tangent to the gain function that begins at the patch switching time τ on the travel time axis (see [83] for details).

The gain function $g(t)$ depends on (i) the actual patch quality, which varies from patch type to patch type but can also be variable within a patch type, e.g. if the pucks are placed randomly and (ii) on the robot environment interaction, e.g. sensor range, search strategy, motor control etc. Thus foraging in two equally sized patches, initially containing the same number of pucks, that is patches with the same puck density, may result in two totally different gain functions and there is no way a forager can predict the gain function of a particular patch before entering the patch. Fig. 8.3 shows two exemplar gain functions and the average gain function over 100 patches (each patch with initially 50 pucks). Thus as [95] argues, the sub-patch type variance has to be considered. This immediately raises the question how does the forager determine the type of patch in which she is currently foraging ? In some scenarios the patch type might be

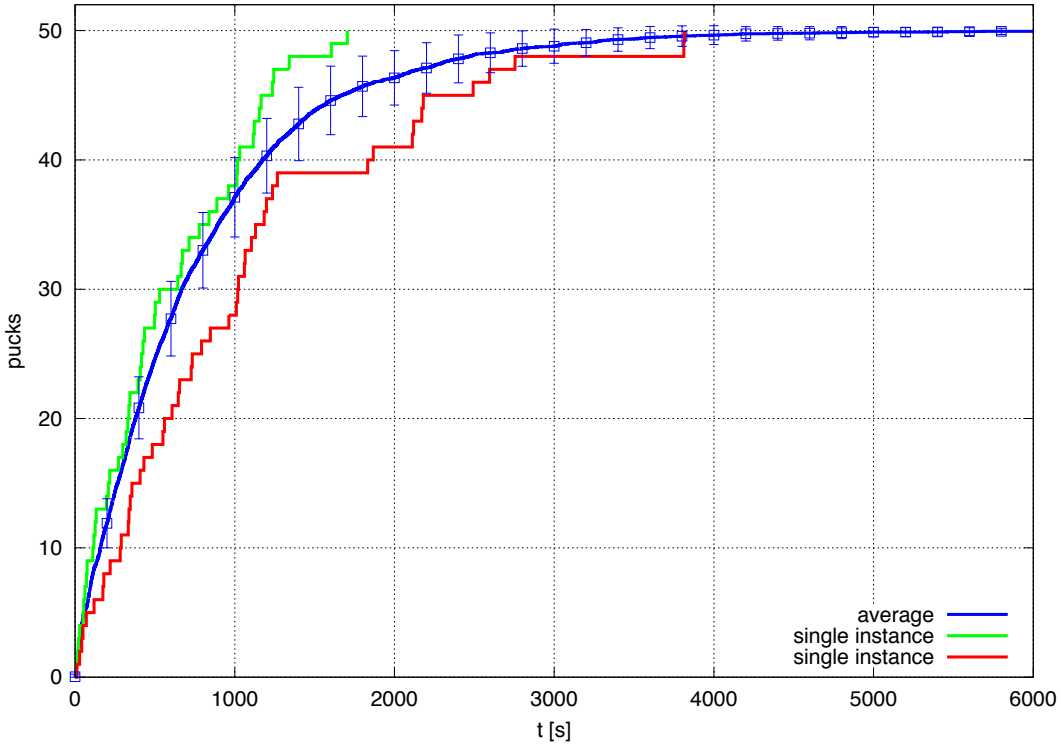


Figure 8.3: Average gain function (thin line) for random foraging in a 50 puck patch, error bars depict the standard deviation. Two instance of the gain function (thick lines) for patches with the same initial number of pucks.

detectable by an external cue, but in general it is not and the forager is required to forage in the patch in order to obtain information about the patch. This adds a patch discrimination problem to the decision process.

To overcome these issues, we suggest dropping the notion of patch types and treating each patch as its own type. (In the following we still use the phrase "patch type" to mean patches with the same initial number of pucks (same puck density), but we do not perform any form of rate maximization based on the notion of patch types.) For unique patches the long-term average gain rate is

$$R = \frac{\frac{1}{n}\sum_i^n g_i(t_i)}{\tau + \frac{1}{n}\sum_i^n t_i} \tag{8.3}$$

We replaced the patch type index j with index i referring to unique patches. The advantage of not having to distinguish patch types and not having to deal with patch subtype variance comes at the disadvantage of having a possibly very large planning horizon of n timesteps. In fact the planning horizon is the lifetime of the robot. Since the robot cannot predict the future, we avoid the large planning horizon by recursively maximizing Eq. 8.3 based on only past experiences and ignoring possible future changes. Then our approximation of the long-term average gain rate while foraging in patch i, based on observations from previously encountered patches $0..i-1$ is

$$\tilde{R}_i = \frac{g_i(t_i) + G_i}{t_i + \tau + T_i} \tag{8.4}$$

Where G_i is the sum of collected pucks and T_i the total time (patch residence plus travel time) from all previous patches $0..i-1$. G_0 and T_0 can be used as a prior that provides the robot

```
1  Algorithm:patchMax
2  init G_0, T_0, τ̃, k_1, k_2, k_3, k_4
3  i = 1
4  forall patches do
5      enter patch i
6      t = 0
7      g(0) = 0
8      repeat
9          t = t + 1
10         randomly forage for one time-step
11         if puck collected then
12             g(t) = g(t − 1) + 1
13         else
14             g(t) = g(t − 1)
15         end
16         r(t) = (g(t) + G_i) / (t + τ̃ + T_i)
17         if t == 1 then
18             r_filt(t) = r(t)
19         else
20             r_filt(t) = (1 − k_3) r_filt(t − 1) + k_3 r(t)
21         end
22         if r_filt(t) − r_filt(t − 1) ≤ 0 then
23             c = c + 1
24         else
25             c = 0
26         end
27     until c > k_1 and t > k_2
28     move to next patch in τ_i time
29     G_{i+1} = G_i + g(t)
30     T_{i+1} = T_i + t + τ_i
31     τ̃ = τ̃ + k_4 (τ_i − τ̃)
32     i = i + 1
33 end
```

Algorithm 6: Task switching algorithm

with an initial estimate of the average patch quality. Both G_i and T_i are a simple model of the average patch quality of the environment. This information (except the prior) is gained by the forager during exploitation. Hence a forager encountering only one patch type will actually maximize Eq. 8.2. But a forager first encountering a series of only low quality patches and then a series of high quality patches will maximize a very different average gain rate function than an omniscient forager. But an uninformed forager maximizing Eq. 8.4, will do as well as possible given the limited available information.

8.3.1 Robot Controller

The core of our task switching method is to maximize Eq. 8.4. This is done by numerically estimating the derivative of R_i at every time step and leaving the patch once the the derivative becomes zero. Since the gain function is assumed to be negatively accelerated, a maximum is found this way.

Algorithm 6 summarizes our task switching method. The robot forages for one time-step, if

it collected a puck the local gain function $g(t)$ is incremented (line 10-15). Next we calculate an approximation of the long-term gain rate based on the experience from previous patches (G_i, T_i), an estimate of the travel time $\tilde{\tau}$ and the value of local gain function at the current time. Because of the stochastic and noisy nature of the gain function the estimate of the long-term gain rate has to be smoothed. In our implementation we use a low-pass filter (line 17-21). Other methods maybe substituted, however it performs well enough for our purpose. As mentioned earlier the patch leaving decision is based on checking if the derivative of the long-term gain rate is equal to zero. Again because of the stochasticity of the gain function we might experience a local region of zero or negative gradient, which could be interpreted as a local maximum. A simple counting step helps to overcome those undesired local maxima (line 22-27). As with the low-pass filter, any suitable method may substituted. The actual patch leaving decision is made in line 27. A patch is left once a maximum is found and a minimum amount of time has been spent in the patch. This minimum patch residence time is helpful during the initial time in a patch, since until the first puck is found $g(t) = 0$ would cause the robot to leave the patch immediately.

Once the robot leaves the patch it travels to the next patch. This travel takes τ_i time. Before starting to forage in the new patch the estimates for the environment quality G and T and the estimate of the switching time $\tilde{\tau}$ are updated (line 29-30).

8.4 Experiments

To investigate the effectiveness of our approach, we conducted a series of simulation experiments consisting of two phases (i) generate foraging data and (ii) test our task (patch) switching policy on the generated data (see Sec. 8.6). To generate the foraging data we used a generic mobile robot model in the well known simulator Stage [97]. The robot is equipped with a short-range colour blob tracker to sense 'pucks', our unit of resources, in its vicinity. The robot knows (or equivalently can detect) the boundaries of a puck patch. Patches are 620 times the size of the robot, and contain initially 10, 30, 50, 100, 200 or 300 pucks placed uniformly at random. A minimum distance between pucks is enforced to avoid overlap. To exploit a patch, the robot randomly forages for pucks, by driving straight until it comes to the patch boundary, where it chooses a new heading that brings it back into the patch, at random. When a puck is detected, the robot servos towards the closest puck and collects it. Collecting a puck takes one simulation time step, so there is virtually no handling time. At each simulation time step we record how many pucks the robot has collected so far in the current patch: this is the gain function.

As mentioned earlier the gain function is not only dependent on the initial number of pucks per patch but also on the robot/environment interaction. To get a good sample of the distribution of gain functions, we randomly generate 100 patches of each of the six patch types and record the gain functions from the robot foraging in those patches. Note that at this point in the experiment no patch leaving decisions are made. The robot simply forages until the patch is exhausted and the simulation is terminated. Testing our approach on this recorded data set rather than during the robot simulation allows us to compare approaches on exactly the same data and it makes it feasible to determine a near-optimal solution by brute force solution search.

As a baseline for comparison we need to find a t_i for each patch such that the long-term gain rate is maximized. No closed form solution is known to this problem, and the gain functions are available as data points only. So we employ a brute force search. Since each patch is unique this technically requires us to solve Eq. 8.3 for all possible combinations of patch residence times. Because this is computationally prohibitive we resort to calculating the average gain function over all 100 instances of a patch type. Then we find the best patch residence time by solving Eq. 8.3 for all possible t ($0 \leq t \leq T_{patch\_exhaused}$) and selecting the t that maximizes the average gain rate. In case of multiple patch types we calculate the long-term gain rate for each combination of residence times on the average gain function. This is only feasible since

	initial pucks per patch					
	10	30	50	100	200	300
μ [s]	1858	2909	3631	4556	5171	5475
σ	825	1184	1271	1337	1206	1208

Table 8.1: Mean and standard deviation of the time required to exhaustively forage patches

the number of patch types considered is small.

In all of the following experiments we used the obtained long-term average gain rate as a metric for comparison. All algorithm parameters required were set manually and kept constant without any attempt to optimize them. The priors G_0 and T_0 were set to zero. To investigate our task switching method under a wide range of conditions we altered the task (patch) switching time τ from very short 10 seconds to very long 5×10^6 seconds ($\approx$6 days). To put this in perspective we report the mean and standard deviation of observed times required to exhaustively forage patches in Table 8.1. The spectrum reaches from almost no switching cost to a switching cost about 200 times the average time required to exhaust a patch.

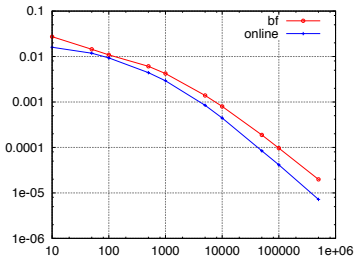
8.4.1 Single Patch Type

In a first experiment we had the robot forage in a series of 100 patches with the same initial number of pucks. Figure 9.3(a)-9.3(f) shows the achieved long-term average gain rate for each patch type over a variety of switching times compared to the brute force solution. From the graphs we can draw three conclusions. (i) If the task switching times are short (ie. much lower than the patch residence times) the performance of our method is in general lower than that of the near-optimal brute force method. The MVT predicts short patch residence times in situations where patch switching is cheap. But because of the various filters (filter parameters kept constant for all experiments over all conditions) our method's responsiveness is too slow in these short residence time situations. We say the performance is lower, but it is still above 78% (except in the 10 puck patches, where the performance drops to 50%). (ii) Under low patch quality situations (10 pucks, 30 pucks) our method performs less well than the brute force method. Again the reason is in the choice of parameters. The filters are too slow for the optimal, short patch residence time. (iii) The method described in this paper achieves similar long-term rates as the brute force method in all other cases examined. Recall that it uses only locally obtained information, in contrast to the omniscient brute force method.

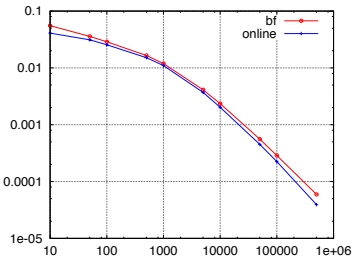
8.4.2 Multiple Patch Types

A more challenging problem is the case where patches of very different quality are encountered. As the MVT predicts the patch leaving decision is not only dependent on the quality of a given patch but on the global quality. To illustrate the difficulty of this decision we give a brief example. Let t_h be the optimal patch residence time if a forager only encounters patches of a fixed, high quality. If the same forager now encounters a mixture of high and low quality patches, t_h is no longer the optimal patch residence time for the high quality patches. The reason is that the cost of lost opportunity has increased due to the patches of low quality. As a consequence the forager should increase t_h under these circumstances.

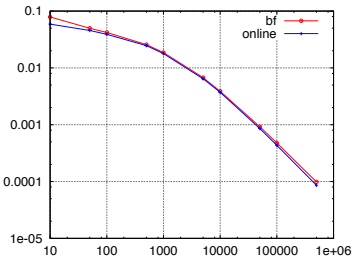
To investigate our system under these conditions we conducted a series of experiments. In a first experiment we had the robot encounter 100 patches of type A and 100 patches of type B in a random order. Figure 9.3(g) and 9.3(h) show the averaged results over 20 trials for patch configurations 50:100 pucks and 50:300 pucks respectively. Errorbars were omitted because of the small standard deviation. As in the single patch type experiments and for the same reasons, the performance is somewhat lower under short switching time conditions, but in the general the graphs show that our method copes well with randomly encountered patches of different qualities.



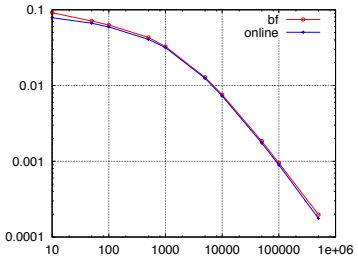
(a) Patch type 10 pucks



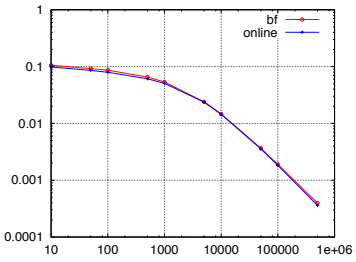
(b) Patch type 30 pucks



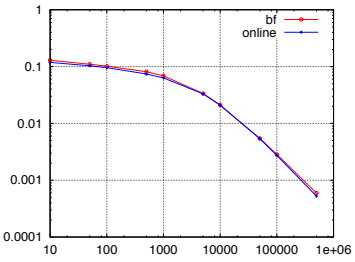
(c) Patch type 50 pucks



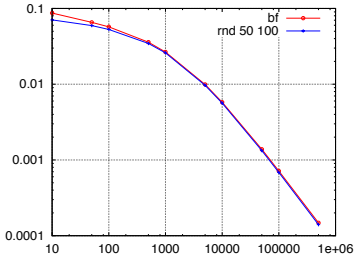
(d) Patch type 100 pucks



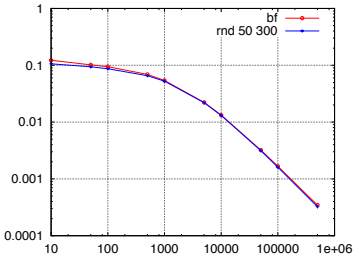
(e) Patch type 200 pucks



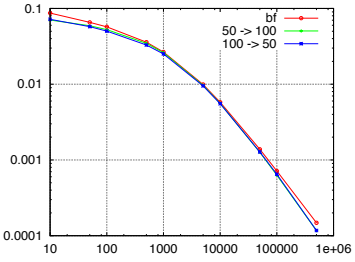
(f) Patch type 300 pucks



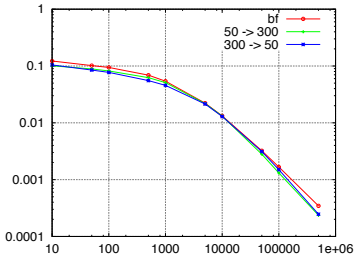
(g) Random sequence of patches with 50 and 100 pucks



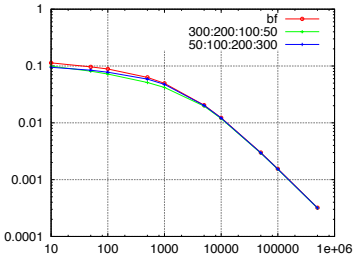
(h) Random sequence of patches with 50 and 300 pucks



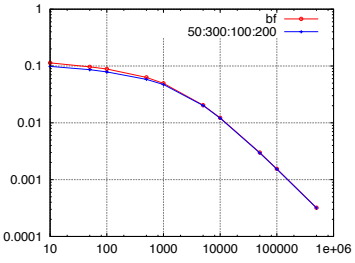
(i) Stepwise sequence of patches with 50 and 100 pucks



(j) Stepwise sequence of patches with 50 and 300 pucks



(k) Stepwise sequence of patches with 50, 100, 200 and 300 pucks



(l) Stepwise sequence of patches with 50, 300, 100 and 200 pucks

Figure 8.4: Long-term average gain rates achived by the bruteforce method (red line with circle) and our online method (green line with cross, blue with asterix). Inter patch travel time τ in seconds on the x-axis and long-term gain rate in pucks per seconds on the y-axis. More details in the text.

	initial pucks per patch					
σ	10	30	50	100	200	300
100	74.0	92.2	96.0	96.4	95.3	92.2
500	76.3	90.2	93.9	94.5	89.7	92.3
700	67.8	89.5	96.9	92.6	88.7	90.1

Table 8.2: Percent performance for variable patch switching time with mean 1000 sec. and standard deviation $\sigma = \{100, 500, 700\}$

An even harder problem is to encounter a longer series of patches of type A followed by a series of patches of type B, where the forager does not know anything about type B patches while it forages in type A patches. On encountering type B patches, the robot has built a strong prior expecting type A patches. In this experiment the robot was faced with a series of 100 patch of one type followed by 100 patch of a different type. The results for 50:100 and 50:300 patches with a stepwise change in both directions is shown in Fig. 9.3(i) and 9.3(l) respectively. Here the brute force method is at a significant advantage because the patch leaving decisions are derived with full knowledge of the future patch change. Our method does not/can not anticipate the patch quality change and thus for the first 100 patches acts under the "assumption" of a constant environment. The error resulting from this "assumption" grows with the difference in patch qualities. That is why the performance difference in the 50:300 scenario (Fig. 9.3(l)) is larger than in the 50:100 case (Fig. 9.3(i)).

Figure 8.4(k) shows the results for a stepwise sequence of 50:100:200:300 puck patches and the reverse ordering. The results are qualitatively very similar to those discussed previously. In one last experiment of this type we choose step wise patch encounter with larger step sizes. The ordering chosen was 50:300:100:200. Results are shown in Fig. 8.4(l). The performance results are again qualitatively similar, suggesting the our method handles this type of variance well.

8.4.3 Variable Switching Cost

So far we tested different switching costs but kept them constant in the single patch type as well as multi patch type experiments. To investigate varying inter-patch travel time, we conducted an experiment in which the travel time between patches was drawn from a normal distribution with mean 1000 seconds and standard deviation 100, 500 and 700 seconds respectively. Table 8.2 shows the results in percent compared to the long-term gain rate of the brute force solutions. Because of the computational complexity the brute force solution was only calculated using the mean and not the actual randomly drawn travel times. As in the previous experiments we see generally good performance and the usual drop in situations with low patch quality.

8.5 Discussion

Task switching under diminishing returns is daily routine for many animals and important for many conceivable autonomous robots. Maximizing the long-term average gain or reward rate under these conditions requires the robot to have knowledge of future gain functions. This is not achievable by a robot relying solely on information obtained by its own actions. To the best of our knowledge no solution to this problem is known. In this paper we have argued that the MVT is not implementable because an instantaneous gain rate is meaningless in the case of rewards obtained in chunks. It also requires a continuous exploration phase in order to find the global maximum rate, but the MVT itself does not explore the action space.

Instead we proposed a task switching method that bases its decision only on previously obtained information, well aware that we therefore maximize a different function. Thus we

may make suboptimal task switching decisions, but these decisions are as good as possible given no information about the future.

An important issue to discuss is how large the time window of past experiences should be, that are considered in the task-switching decision. In this paper we simply included all past foraging experiences when modelling the global patch quality. This is reasonable as long as the past is a good predictor for the future. On the other hand in situations where the future strongly deviates from the past, forgetting or a short memory can be beneficial. The memory size is also interesting from a behavioural ecology point of view, because it might explain why animals often appear to maximize the short-term and not the long-term intake rate [107]. In future it would be interesting to investigate what influence the memory size has on the rate maximization of a robot and what the optimal size is.

We draw a lot of insight from behavioural ecology, but we make no claims about mechanisms employed by animals.

8.6 Experimental Data

In accordance with the Autonomy Lab's policy on code publication, the foraging data and the implementation of the experiments are made available online at `git://github.com/jwawerla/tsw_experiment.git`. The exact data that led to the presented results can be accessed via the commit hash `4f84a82d09f2c181df57ab5d7faa2e53cc3348f3`.

Chapter 9

A Fast and Frugal Method for Team-Task Allocation in a Multi-robot Transportation System

Jens Wawerla and Richard T. Vaughan [7]

9.1 Abstract

In this paper we present two task-allocation strategies for a multi-robot transportation system. The first strategy is based on a centralized planner that uses domain knowledge to solve the assignment problem in linear time. In contrast in the second strategy, individual robots make rule-based allocation decisions using only locally obtainable information and single value communication. Both methods are tested and analysed in simulation experiments. We show that the rule-based method performs well but the lack of information has to be paid for with increased energy consumption.

9.2 Introduction

Mass transportation of goods is an essential feature of the modern global economy. The port of Vancouver alone handles 76.5 million metric tons annually [108]. Transportation is a natural task for autonomous mobile robots, and future robots will perform more of these tasks: moving goods around in warehouses [109], parts around factories, mail around campuses, and shipping containers around the world. To achive this we need systems that assign transportation tasks to robots. Several versions of the problem can be stated, e.g. with either fixed or time-varying item source and sink locations, production rates, storage capacities and delivery rewards.

In this paper we consider a multi-robot transportation system with fixed, known source and sink locations and constant reward for delivered items. Sources produce items at variable rates, but cannot store more than one item at a time. Fig. 9.1 shows an example of this case. The objective of the robots is to transport pucks (the conventional abstract unit of robot-transportable resource) from a source to a corresponding unique sink. We desire that tasks are allocated to robots such that the value of our system is maximized. We use the amount of energy expended by the robots as our cost metric and pucks transported as our work metric. To reduce the total running cost, the task-allocation system can deallocate a robot temporarily

by sending it to a depot, marked D in Fig.9.1. While at the depot the robot is on stand-by and consumes negligible energy.

We propose and compare two practical task-allocation methods: (1) based on a central, omniscient planner; and (2) governed by a simple heuristic, using only locally sensed information and minimal and infrequent communication between the robots. A challenge for both methods lies in correctly dealing with interference, inherent to all multi-mobile-robot systems. Below we show that interference can create an unintended feedback loop in a naive solution, leading to a drop in performance. We then suggest a simple yet effective solution to this problem.

9.2.1 Related Work

Robot task allocation is a widely studied field. It can be broadly classified into two classes: centralized planner based systems and systems that rely on individual robots making individual task allocation decisions. Planners are often based on auction mechanisms in which robots bid for tasks, e.g. Gerkey's MURDOCH [110]. As we will show, the problem we study does not require the sophistication of this class of planners.

Local decision mechanisms are attractive because they are redundant and by definition do not rely on a centralized agency. Labella et al. [111] use a minimalistic rule originating from modelling ant behaviour. Each robot's probability to leave the nest is increased by a small δ if it found food on the last foraging trip and decreased by the same δ if the robot returned empty handed. Liu et al. [85] present a similar central place foraging system but with the explicit goal to minimize energy expenditure. The decision each robot makes is based on two thresholds. One for the amount of time spent resting and one for the time spent foraging. The thresholds are adapted by certain cues. Colliding with other robots makes a robot more likely to rest and successfully retrieving food makes the robot more likely to keep foraging. In an experiment with up to 12 Khepera robots Krieger et al. [112] use a similar threshold based task-allocation mechanism to keep the swarm's energy reserve at a safe level. That paper also provides an extensive list of references to studies about task allocation in social insects. All three methods have been shown to arbitrate between work and rest periods but it is not clear how to select between two work tasks and resting.

Jones [113] proposes a probabilistic rule that is shown to work for at least two tasks. The robots calculate the probability of switching tasks based on a local estimate of the number of robots performing the same task and on an estimate of the available work load. This requires the robots to communicate what task they are currently engaged in. Jones explicitly prohibits an idle task. McFarlands's Cue×Deficit-rule [82] allows robots to select between two types of resource depending on the robot's deficit and encounter rate of a particular resource. McFarland models situations in which the robot is in need of both resources, e.g. needing water and food. In the transportation problem considered in this paper a deficit of a certain task is not meaningful McFarland's method each agent to encounter both resources occasionally in order to update the cue estimates. This cannot be guaranteed in a transportation problem.

Some work uses machine learning techniques to estimate the utility of tasks. A recent example is Dahl's vacancy chain scheduling [114]. Reinforcement learning and ϵ-greedy action selection is used to have robots choose between two transportation tasks. The scenario is similar to ours but less challenging due to the absence of fixed obstacles in the world. Also energy is not considered in the performance metric. The method we propose is significately less complex then that proposed by Dahl and does not suffer from any explicit trade-off between exploration and exploitation as Q-learning does.

9.3 Robot system

The robots used in this work are generic models of a differential drive robot in the well-known simulator Stage [97]. Each robot is equipped with a 360° field of view laser range finder

providing 64 samples with a maximum range of 5 meters. Each robot can localize itself with an abstract localization device, modeling a GPS module or SLAM implementation. To transport pucks each robot is capable of sensing pucks, picking them up and dropping them off. The cargo capacity is limited to one puck per robot. In order to communicate with other robots or with the centralized planner the robots are equipped with a generic communication device. We assume reliable communication, but as we will show communication used by our task allocation policies is (informally) minimal.

As energy expended by the robots is one of our performance measures the energy model of the robots is important. Two types of energy expenditure occur, (1) energy used for computation and sensing is expended at a constant rate and (2) energy used for locomotion is expended at a rate proportional to the speed of the robot. The first energy model is fairly accurate. The second model does not take acceleration into account and is thus not fully realistic. Acceleration and deceleration occur frequently in high interference situations, therefore our energy model does not penalize interference as much as one would expect. The actual energy rates are modelled after a typical Pioneer 3DX robot. Robots in the depot are in a stand-by mode and do not use any energy.

The robot control software is split in two parts, (1) a common part that handles navigation, obstacle avoidance, picking up and delivering pucks etc. and (2) a task allocation part that decides which task to perform next. An overview is shown in Fig. 9.2.

Path-planning is done using a wavefront planner (WP). This widely used technique discretizes the world into a grid and builds a gradient of shortest distances starting at the goal location. The shortest path is then given by gradient decent from the robot's current position. (see [115] for details). In our implementation we augment the map data with sensor data from the past n time steps. This enables the planner to incorporate knowledge about temporary changes in the world such as congestion. Fig. 9.1 illustrates this feature, one robot has planned a seemingly longer path around the south side of the obstacle at (0,10). While four other robots planned the shorter path along the often more congested north side of the same obstacle. Emphasis was put on reliability of the planner and not so much on interference reduction. For example the introduction of simple traffic rules such as "always stay on the right side of corridors" may reduce interference noticeably. But as highways in any major city during rush-hour evidence, interference cannot be avoided as the number of agents increases. In fact to investigate our task-allocation methods we want to explore the configuration space under low and high interference conditions.

The planned trajectory is used by a sensor-based obstacle avoidance routine to navigate the robot along the trajectory to the goal. We use Minguez's [116] Nearness Diagram (ND), an extension of the Vector Field Histogram approach. Minguez's implementation is available on his homepage[1].

The key contributions of this paper are the task-allocation methods. We investigate and compare two methods, both using the same navigation and control routines described above.

9.3.1 Allocation Re-Planner

The allocation re-planner is a centralized planner that has full knowledge of the world, the production rate of the sources and the current robot task assignments. An optimal assignment is achieved if the puck production rate of a task equals the sum of the transport rate of all assigned robots. Only under this condition is there no unused transport capacity nor are pucks waiting at the source. The optimal number of robots is thus given by

$$N_i = \frac{\dot{p}_i T_i}{c} \tag{9.1}$$

where the index i refers to the i-th task, $\dot{p}_i$ is the production rate of the task, T_i the round trip time the robot needs to drive from the source to the sink and back, and c is the puck

[1] `http://webdiis.unizar.es/~jminguez/code/sources_ND_english.zip`

carrying capacity of the robot ($c = 1$ in our case). It is fair to assume that a central planner has accurate information about the robot's carrying capacity and the production rate for each task. But knowledge of the round trip time is difficult to obtain. The round trip time can be estimated from the distance between source and sink and the speed of the robot. The distance between source and sink maybe accurately obtained from a path-planner e.g. the wavefront planner. The actual speed of the robot is a function of the robots hardware, the control software, as well as the interference between the robot and other robots or fixed obstacles in the environment. Modelling this interference and the resulting change in speed is difficult. For example, in the environment in Fig. 9.1 task A is subject to a higher degree of interference then task B because of the narrow corridor between the obstacles at (0,10) and (-8,14). Modelling interference is generally a difficult endeavour. A simple model is proposed by Seth [117]. Given an interference factor Seth's model uses this factor and the number of agents to calculate the change in performance. This approach has two limitations, (1) we do not know the interference factor and (2) since this model calculates the effect of interference on the average performance and not on a sensori-motor level it is too high level for our purpose. Instead of modelling interference we take the embodied approach. A naive solution is to average over the actual round trip time, as experienced by the robots, and base the planners allocation on this time estimate. Unfortunately this method has an undesired feedback loop. Randomly occurring interference will increase the estimate of the round trip time, causing the planner to allocate more robots to the task and thus eventually increasing the interference. This in turn results in even more allocated robots. Pointing out this effect is a novel contribution of this paper.

What is needed, is a way to distinguish between the "true" round trip time and delays caused by interference due to randomly occurring congestion. As a simple measure for interference we use the speed of the robot. We sum up the time steps during which the robot's speed is below a certain threshold. The robots then report the experienced round trip time corrected for the time wasted due to interference and the time spent waiting at the source and sink $\hat{T}_i$.

Once a robot completes a task the planner may assign it a new task using Eq. 9.1 and a low-pass filtered estimate of the interference-corrected experienced round trip time. Alternatively the robot might be sent to the depot if it is no longer needed. The complexity of this planning step is $O(k)$ where k is the number of tasks. In the depot, robots are waiting in a queue and only the head of the queue queries the planner for a new assignment.

9.3.2 Allocation Heuristic

The allocation heuristic has no *a priori* access to any information other than the location of sources and sinks. Since the production rates are unknown, robots initially select a task at random with equal probabilities. Every time a robot returns from delivering a puck it faces one of three possible situations:

1. robots are already waiting in a queue to pick up a puck
2. no robots are waiting and a puck is available for pick up
3. no robots are waiting and no puck is available yet

Situation (1) can be evidence that the number of robots assigned to the task is too high. It can also be a result of an unfortunate spatial clustering of robots. Situation (2) might indicate that the number of robots assigned to the task is too low since the production rate appears to be higher then the transportation rate. But again this could also be a consequence of a brief congestion somewhere along the transportation route. Situation (3) is, at least from this particular robots point of view, the ideal case. On one hand the absence of a robot waiting queue indicates that there are not too many robots assigned to the task. On the other hand pucks not already waiting indicates that the task has recently been serviced by another robot.

The allocation heuristic is entirely based on these three situation, which are easily identifiable by the robots. A robot in situation (1) randomly chooses a maximum time it is willing to

wait in the queue according to

$$t_{wait} = max(t_{minwait}, \hat{T}_i \cdot r) \tag{9.2}$$

where $t_{minwait}$ is a minimum waiting time, $\hat{T}_i$ the corrected round trip time and r is a uniform random number in the interval $[0, 1]$. If the robot is able to enter the loading bay before the waiting time is up, it will continue with the current task. Otherwise it stops performing any task and returns to the depot.

A robot in situation (2) broadcasts a worker recruitment message for the current task with a probability p_b. The first robot in the depot will respond to the allocation call by starting to perform the task in need of an additional worker. In the case that there are no robots in the depot, the allocation message remains unanswered. Situation (3) does not require the robot to alter the allocation, thus it simply continues to perform the task. Because production rates may drop to zero, a robot in situation (3) will only wait for a puck to be produced for a maximum duration. We choose the last experienced round trip time to be the maximum waiting time before returning to the depot.

Note at no point are the robots obtaining any information about the tasks production rates, the number of assigned robots per task or the number of unassigned robots. The group allocation is entirely based on locally observed information (apart from information about the work site location) and a single value broadcast message, used infrequently.

9.4 Experiments

All experiments reported in this paper were conducted with 18 simulated robots in Stage [97], a sensori-motor level multi-robot simulator. The robot controller was the same during all experiments, just the task selection component was varied. The simulation environment is shown in Fig. 9.1. Initial starting positions of the robots, the location of source and sink as well as the total sum of production rates were fixed. We just altered the ratio of the production rate between the two tasks. Performance was measured in two dimensions, total number of pucks transported in a fixed amount of time and total energy expenditure during this time. These values are not directly convertible into a single performance criterion. The result of any form of linear combination is dependent on the weighting and is thus application dependent and not suitable for a general comparison. Due to the cost of interference, ratios like pucks transported per unit energy, generally favour allocations where only one robot is performing the task while the remaining robots are in stand-by. So we report the results of our experiments in a energy-work graph, similar to a precision-recall graph in machine learning. Results closer to the upper left corner of a energy-work graph should be considered better. Less energy is then expended and more work performed. All experiments were run for 2 simulated hours.

9.4.1 Constant Production Ratios

In a first set of experiments we kept the production rate constant at 1:1, 2:1 and 10:1 respectively. Each configuration was tested 20 times. Fig. 9.3 shows the results in energy-work graphs for the re-planner and the heuristic allocation for a selection of broadcast probabilities. We tested probabilities with 0.1 increments, but due to space constraints only show the most interesting ones.

In order to not only compare our two task-allocation methods with each other and to allow the reader to see where in the possible energy/work space our solutions lie, we also characterize the possible energy/work space. To do so we assigned a fixed number of robots to each task and kept this assignment constant during the course of a 2 hour simulation run. We repeated this process for all possible ways to assign up to 18 robots to two tasks, where the unassigned robots return to the depot, just like in the case of the other two policies (a total of 189 experiments).

Table 9.1: Average number of robots assigned between two tasks with stepwise change of production rates from 3:1 to 1:3

Ratio	re-planner	heuristic with broadcast probability			
		0.0	0.3	0.7	1.0
3:1	8.0 : 3.0	7.4 : 3.6	9.1 : 4.1	10.1 : 4.1	10.6 : 4.4
1:3	3.1 : 8.0	3.4 : 3.6	3.7 : 8.1	3.8 : 10.4	3.8 : 11.7

As mentioned earlier it is application dependent whether we wish to minimize energy expenditure or maximize puck transportation or find some middle ground. But in general we wish to minimize wasting energy. Therefore control policies that achieve the same number of pucks transported while spending less energy have to be considered better policies. From the fixed assignment we can see the line of good performance, it is the left hand side of the fixed assignment energy-work tuples. From Fig. 9.3 we conclude that the re-planner performs very well, since its results are on the line of good performance for all configurations tested.

The results from the allocation heuristic are subject to a higher degree of variance. This is to be expected due to the stochastic nature of the heuristic. Further we observe that a low broadcast probability yields results closer to the line of good performance. The reason is that a higher probability causes a higher frequency of re-assignments and these cost energy as the newly allocated robot travels to the worksite. The key to the heuristic proposed is to trade-off re-assignments and adaptation to the task configuration. From the graphs it is apparent that this adaptation is more difficult in the 2:1 case. The 1:1 case is easier, because of the initial random task selection, the ratios of robots already closely matches the production rates and "only" a reduction of workers has to be achieved by the system. The cues used by the heuristic are more pronounced in the 10:1 case, because the ratios are so much more different.

Summarizing, the allocation heuristic performs remarkably well considering that no information about the production rates or the allocation of other robots is known to the heuristic.

9.4.2 Changing Production Ratios

Next we investigate how the two policies perform in situations of changing production rates. Therefore we conducted an experiment in which we set the production ratio to 3:1 for the first hour and then reversed it to 1:3 for the second hour. Unlike in the constant rate situation, a fixed assignment obviously does not show us the possible energy-work space. So we restricted our analysis to comparing the re-planner with the heuristic. The results are summarized in energy-work graphs (Fig. 9.4) and the average robot assignment numbers in Tab. 9.1. The assignment numbers are the average number of robots assigned to a task during a production rate period. As to be expected, with zero broadcast probability the heuristic just de-allocates robots once the production rate of the first task drops. With broadcasting probability $P_b = 0.3$ we achieve enough recruitment so that the ratio of assigned robots matches that of the re-planner. The number of assigned robots is generally higher, because of the overhead of constant re-assignment. As the probability increases we observe an increase in overhead and thus higher assignment numbers but the ratio is similar to that of the re-planner. Disabled recruitment ($P_b = 0.0$) means fewer workers after the ratio change and that results in fewer pucks transported (Fig. 9.4(a)) and less energy spent. Using a broadcasting probability of $P_b = 1.0$ does not yield more pucks transported but results in more energy wasted on frequent re-allocation (Fig. 9.4(c)), this coincides with our analysis of the assignment numbers. More interesting is Fig. 9.4(b), which shows that the work rate achieved by both policies is about the same (broadcast probability of 0.3), but the heuristic requires more energy. As in the first experiments, lack of information has to be paid for with increased energy expenditure.

9.5 Discussion

For energy to be used in a performance analysis of simulation experiments an accurate model of the robots energy consumption is required. Mainly this model must be able to capture possible differences in energy requirements of the sensori-motor activity caused by different control policies. To test our methods under more realistic circumstances we are currently working on real robot experiments. An issue not addressed in this work and left for the future is refuelling. Our current implementation can handle refuelling in principle but more and especially longer experiments are required for a thorough investigation.

The sources of the transportation task used throughout this paper were only able to store one puck at a time. In the future we plan to investigate systems in which pucks accumulate at the source if the robots do not pick them up in time. Another question left for the future is how does task priority, e.g. expressed in terms of task-dependent rewards, influence the allocation policies?

The paper introduced two task-allocation strategies for a multi-robot transportation system. The computational complexity of both strategies is small. In simulation experiments we compared the strategies in form of energy-work graphs. It is not possible to generally say which method performs better because we lack a unifying metric. But we can conclude that the re-planner's performance is on the line of good performance based on a comparison with exhaustive search. The performance of the heuristic allocation is often slightly below the line of good performance. Yet the heuristic still performs well, considering that it uses no information about the production rates nor about the task allocation of other robots. This simple rule-based allocation policy yields remarkable results entirely relying on local information and minimalistic broadcasting.

9.6 Experimental Data

In accordance with the Autonomy Lab's policy on code publication, the source code and analysis scripts of the experiments are made available online at `git://github.com/rtv/autolab-fasr.git`. The exact data that led to the presented results can be accessed via the commit hash `ed47f9212cf3b72f6a083b48bc2d39a1b79006af`.

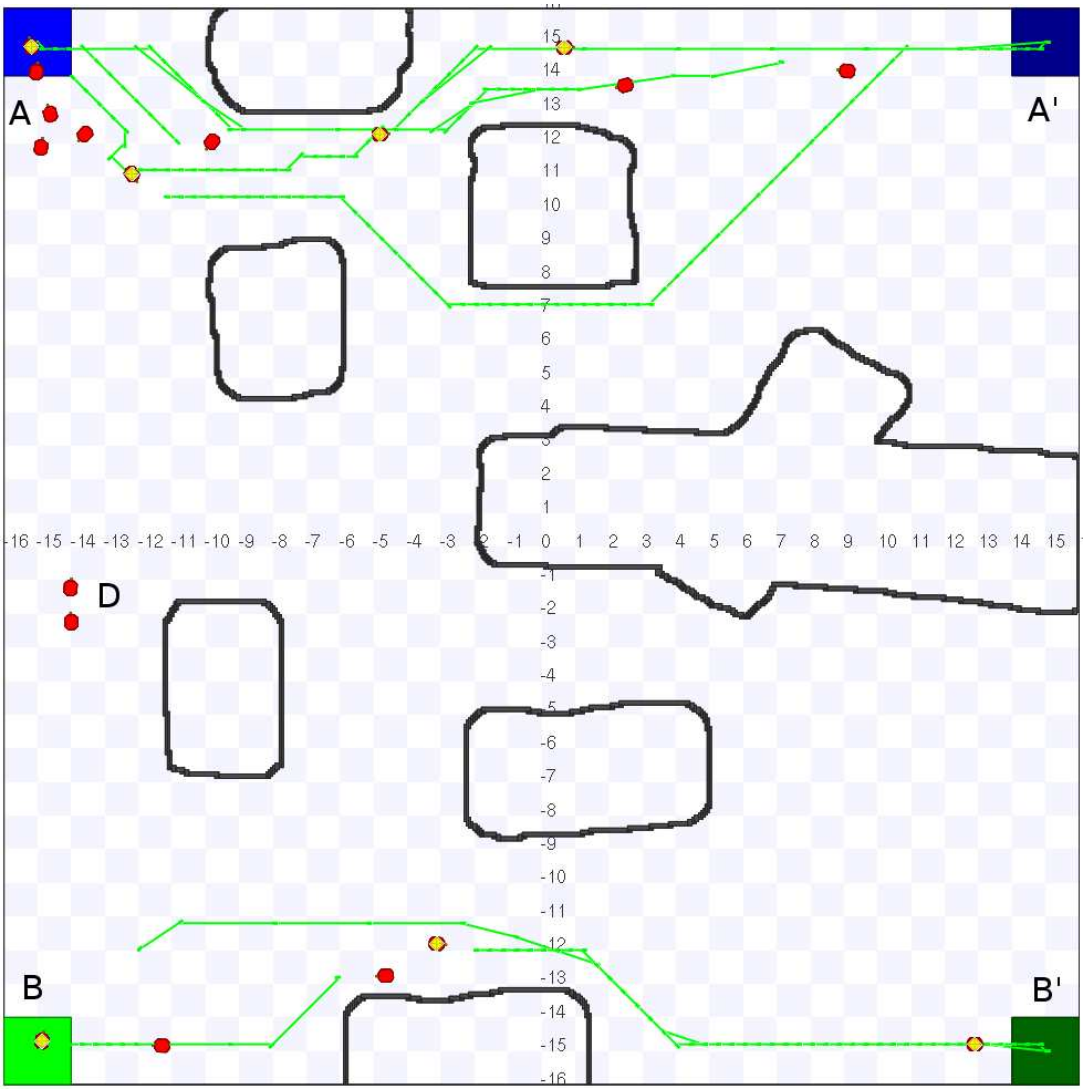


Figure 9.1: Screenshot of a Stage simulation with two tasks, transporting pucks from A to A' and from B to B'. D marks the robot depot for unassigned robots. The thin (green) lines show the path planed by each robot. Robots carrying a puck are displayed with a (yellow) diamond on top.

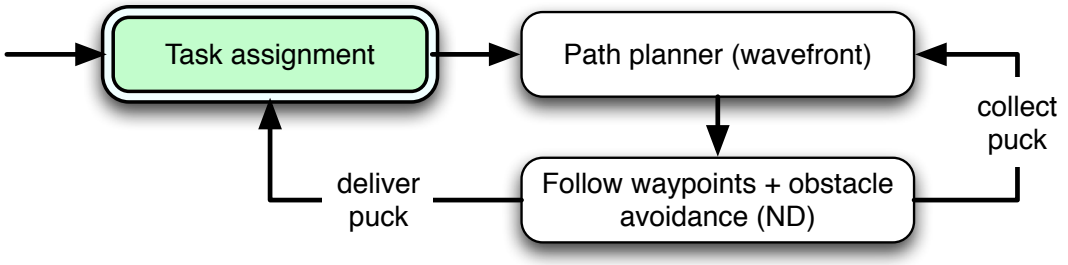
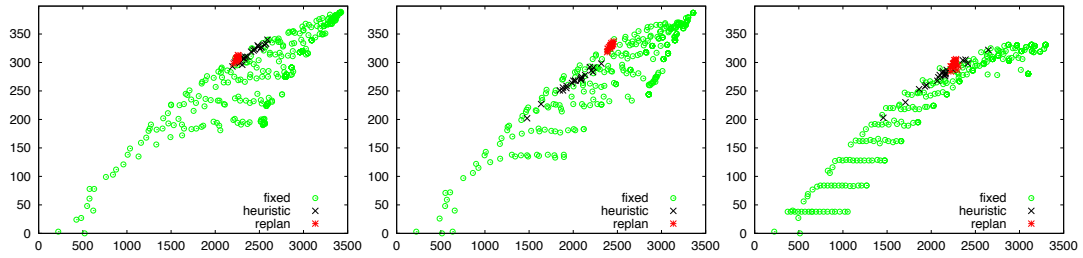
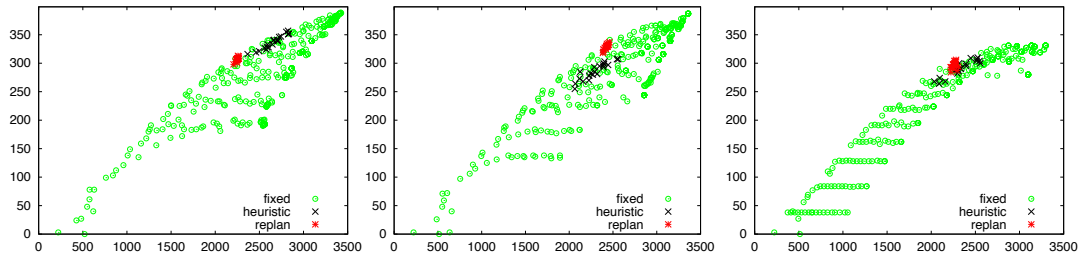


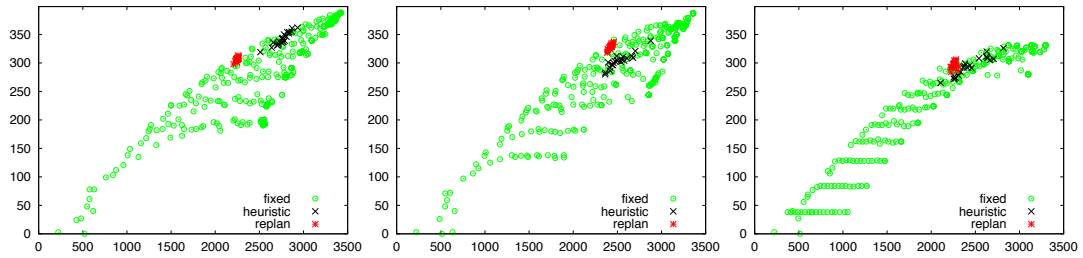
Figure 9.2: Overview of the robot system



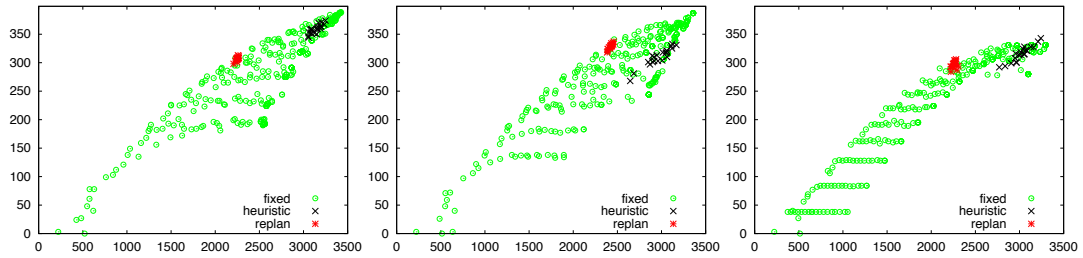
(a) Ratio 1:1, broadcast probability 0.0 (b) Ratio 2:1, broadcast probability 0.0 (c) Ratio 10:1, broadcast probability 0.0



(d) Ratio 1:1, broadcast probability 0.1 (e) Ratio 2:1, broadcast probability 0.1 (f) Ratio 10:1, broadcast probability 0.1

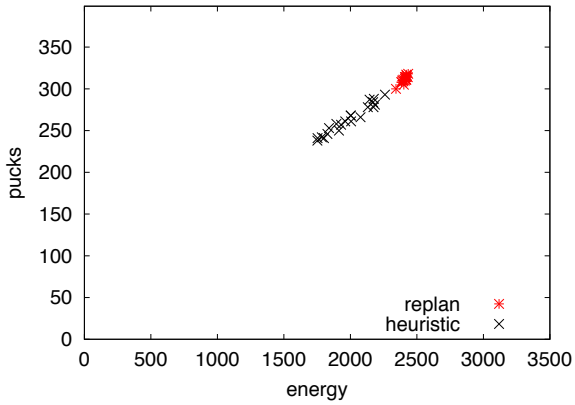


(g) Ratio1:1, broadcast probability 0.2 (h) Ratio 2:1, broadcast probability 0.2 (i) Ratio 10:1, broadcast probability 0.2

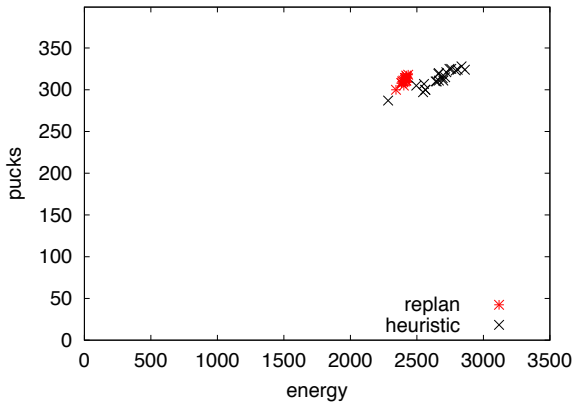


(j) Ratio 1:1, broadcast probability 1.0 (k) Ratio 2:1, broadcast probability 1.0 (l) Ratio 10:1, broadcast probability 1.0

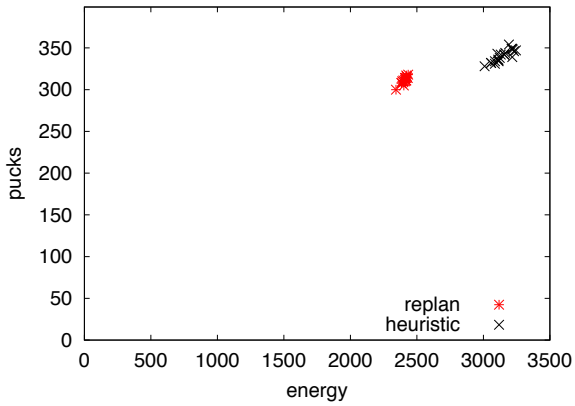
Figure 9.3: Energy-Work graphs for different production rate ratios and broadcasting probabilities. Energy expended is plotted on the x-axis and pucks transported on the y-axis. The green circles mark results from fixed robot assignments to visualize the possible performance space. The performance of 20 trials of the re-planner is shown in red asterixes and the performance from 20 trials of the allocation heuristic is shown in black crosses.



(a) Ratio 3:1 → 1:3, broadcast prob. 0.0



(b) Ratio 3:1 → 1:3, broadcast prob. 0.3



(c) Ratio 3:1 → 1:3, broadcast prob. 1.0

Figure 9.4: Stepwise change of production rates from 3:1 to 1:3

Chapter 10

Publishing Identifiable Experiment Code And Configuration Is Important, Good and Easy

Richard T. Vaughan and Jens Wawerla [4]

10.1 Introduction

A few months ago, a graduate student in another country called me (Vaughan) to ask for the source code of one of my multi-robot simulation experiments. The student had an idea for a modification that she thought would improve the system's performance. By the standards of scientific practice this was a perfectly reasonable request and I felt obliged to give it to her. With our original code, the student could (i) re-run our experiments to verify that we reported the results correctly; (ii) inspect the code to make sure that it actually implements the algorithm described in our paper; (iii) change parameters and initial conditions to make sure our results were not a fluke of the particular experimental setting; (iv) modify the robot controllers and quantitatively compare her new method with our originals. It would cost me nothing to make her a copy of our code, and her methodology would be impeccable. Why then do we read so few papers using this methodology?

It turned out to be impossible to identify exactly which code was used to perform the experiments in our years-old paper. We had not labeled the source code at that moment, and it had subsequently been modified. All the code was under version control, so we could obtain approximately the right code by looking at revision dates. But having only *approximately* the right code strictly invalidates the replication of the experiments. The user has no way of knowing what the differences are between the code she has and the code we used. So we were able to offer the requesting student some code that may or may not be that used in the paper. This was better than nothing, but not good enough, and we suspect this is quite typical in our community.

This disappointing episode was a wake-up call for me, and our group has been discussing how we can make sure this doesn't happen again. We propose to *routinely* publish the exact code for each experiment that we use to justify any claims at all. This short paper explains why we think complete experiment publication is **important**, why it is **good** for the originating researchers as well as subsequent users, and outlines our protocol to show how **easy** it is to do.

10.2 Publishing code is easy

The complete source code, build scripts, configuration files, maps, log analysis scripts, list of critical external dependencies, details of run-time environment and any other instructions and resources necessary for a skilled researcher to replicate the experiment should be packaged, labeled with a unique identifier and placed in public for free and anonymous download by any reader. The paper should contain the identifier.

This can be easily and cheaply achieved as follows. The code is assembled into an archive file (tarball, gzip, etc), and a digital signature is obtained using a cryptographic hash function such as MD5[118] or similar[1] The archive is published at some reliable Internet host, and its URI and signature are published in the paper. The archive can also be linked from the authors' web publication list.

Upon downloading the file, the user can determine that the archive matches the signature in the paper. Use of a good hash identifier makes it very difficult for authors to modify the code by mistake or design, without this being detectable by the user/reader.

Modern software development tools make for an even easier process. Revision control systems like Git[2] automatically generate a SHA1 cryptographic hash key for each committed version, such that there is a low probability of any two packages or versions having the same key. The entire revision control database can be easily cloned from an URI, and users can check out the correct version by its signature, while still having access to later versions. The differences between versions are easy to inspect using Git's tools. We have chosen this approach, and are hosting our Git repositories at the independent host GitHub[3]. While GitHub's tools and convenience are currently compelling, the ideal host would be a reliable and long-lived independent institution such as a university, national library or professional organization such as the IEEE.

10.3 Publishing code is important

10.3.1 Falsifiability and shared artifacts

Publishing the actual experiment alongside the paper which describes and interprets it increases the scientific and practical value of the work. It goes a long way to solving a problem our field faces from a philosophy of science point of view: the fact that we are a synthetic science that creates and studies artifacts, rather than a natural science that studies an extant universe common to all scientists. By reproducing and sharing our artifacts we synthesize a common environment.

Scientific claims are required to be falsifiable. If I make a claim in a paper about a system I created, and to which you do not have access, my claim is not falsifiable in practice. My claims are more scientifically valuable if I make them as easy to falsify as possible, which I can achieve by publishing the artifacts.

10.3.2 Repeatability and quantitative comparison

In the natural sciences experimental results gain credibility after they are independently repeated at least once. In order to be able to repeat an experiment, we often require many details that are not available in the paper. As we are often able to make the exact and entire experiment available for replication at negligible cost, we can achieve the best possible repeatability.

[1] MD5 has been shown not formally collision resistant[119]. However it is likely to be good enough for the purpose described here, and its near-ubiquity makes it a reasonable choice.

[2] http://git-scm.com/

[3] http://github.com

Of course, we can not prove experiments are correct and while simply re-running a program is not a strong validation, even this alone can show up mistakes. A stronger validation is obtained by completely re-implementing the code, or the important parts of it, but by testing the new version using the original setting, as determined by inspecting the original code, we can improve our confidence in the results.

As in all of science, much work in robotics can be considered incremental improvement over the work of another. This usually requires reimplementing the original experiment from natural language and formal mathematical descriptions. This re–implementation step usually allows only qualitative comparison, since the details of parameters and initial conditions, etc, are rarely published. It can also be a source of error and raises question such as "did the new author really find the very best parameter set?". Experiments made public in an executable form will improve fairness to the original author and will allow quantitative comparison of results.

A second level of repeatability is available to us. Components of experiments can be re-used in different experiments and settings. If the component performs as expected in this new setting, our confidence in it increases. In fact this re-use of code is a cheap way of reproducing experiments.

10.4 Publishing code is good

10.4.1 Efficiency

Having access to data sets and software implementations increases the efficiency of the scientific process in several ways. In the case of incremental work, it saves a great deal of re-implementation effort. While the use of middleware like Player[120], ROS[4], and Microsoft Robotics Developer Studio[5] has increased the rate of code reuse in recent years, these systems focus on low-level components and it is still unusual for a robot controller or an implementation of an algorithm to be substantially reused. Making code available by default would encourage reuse, particularly if the code is of good quality.

10.4.2 Quality

The quality of a research contribution is a function of the soundness and originality of its theoretical foundation, the depth of analysis given and the clarity and thoroughness of its presentation. It is assumed that the software that produces the results is correct. Yet it is all too easy to make implementation mistakes that grossly influence the outcome of an experiment. Even when a paper presents a complete formal algorithm, discrepancies between the description and the implementation that produced the results are possible. Such discrepancies are impossible to detect without access to the source code. We can very easily make code available for peer review, and so we should.

Further, it is often argued that well written and documented software has fewer bugs. Developing software with the expectation that it will be peer reviewed and reused is likely to cause roboticists to write better code, thus increasing the overall quality of the work even before external review. We should write code as we write papers: to be read and understood; to contribute to knowledge.

10.5 Issues and objections

Achieving code publication requires a number of issues to be addressed. Some of the most significant are:

[4] http://www.ros.org
[5] http://msdn.microsoft.com/robotics

1. *"I object! All that extra work takes too long...."*

   There are three arguments. First, while producing peer-reviewable code may *feel* like it takes longer, the additional discipline and code review should result in improved code quality. By reducing bug-hunting and re-runs of faulty experiments, the experimenter could actually save time compared to a typical messy code base. Second, starting with others' published code saves time in the first place. Third, extra work is justified by the methodological advantages: the main role of the "extra" work is to improve the quality and usefulness of the research results, thus it should not be considered overhead.

2. *Real and unique robots:* Complete code publication may be straightforward when experiments are done in simulation only. Yet real-robot experiments are essential. The arguments for experiment source sharing still hold for real robot systems, and the value of the work is maximized if the authors facilitate replication and extension. This can be done by using a well-known robot e.g. Pioneer, Khepera, which can be assumed to be widely available in research labs around the world. Well-known robots also have the advantage that respected simulation models are readily available.

   If a custom robot is essential, we suggest providing either (i) a model for a well-known simulator, or (ii) a dedicated simulator including source code. Also when using custom hardware, using a well-known and open API for controller code, (e.g. Player, ROS) makes porting to another robot or simulator as easy as currently possible. Ideally, in all cases where a simulation can produce similar results to the real robot with a reasonable amount of effort that simulation should be provided.

3. *Licensing:* The free reading, copying, modification and subsequent redistribution of modified code is absolutely required. In most jurisdictions copyright law automatically applies, so the code must be explicitly licensed to allow redistribution. The community already makes extensive use of Free and Open Source Software, so we have experience with suitable licenses.

4. *Trade secrets and competitive advantage:* Some authors feel that since their code is precious, by "giving it away" they give away their competitive advantage. If a "competing" lab needs six months to replicate my experiment, I can get further ahead in the meantime. While this position is tempting for the individual, we are seeking advantages in efficiency and quality for the entire community, including our taxpayer-supported funding agencies. Companies are under no obligation to serve the community, but they can get the benefits described above by first protecting their ideas with patents before publication. If groups withhold their code for their own interest and against the interest of the community, their work is manifestly less valuable than it could be, and should be evaluated accordingly. Conversely, releasing high quality code should enhance a group's reputation and success rates. This provides a feedback mechanism that reinforces code publication.

10.6 Encouraging Code Publication

How can the publication of source code be made a community norm? Assuming the existence of a few suitable protocols, how can researchers be encouraged to use them? Though we believe the research quality and efficiency benefits should persuade many researchers, achieving such a large cultural change is likely to require activism at various levels in the community.

At the most executive level, organizations such as the IEEE could make paper publication conditional on code publication, perhaps with exceptions in extenuating circumstances. Such a policy seems impossibly heavy-handed at the moment, though it might be possible for individual journals and conferences. Perhaps a new journal or conference could adopt this strategy as a differentiating feature: if the arguments above are true, such a venue could expect to become disproportionately influential. We cite some evidence of this effect from other fields below.

If *requiring* code publication seems too ambitious, it is straightforward to *prefer* it. Publishers, editors, program committees and individual reviewers can state that, all else being equal, submissions that provide code are preferred over those that do not. In practice, editors would need to advise reviewers on the weighting of this preference, as with any other major criteria.

One simple concrete proposal is that the major conferences offer a new prize for "best" (in quality, novelty or significance) published code, along with the usual best paper and service prizes. This would be a low cost, high visibility measure that recognizes this as a new and significant way to contribute to the community.

At the most grassroots level, professors can expect their students to back up all written work with published code. Generations of grad students are short, and norms can be quickly established by generational change.

Another idea is that when an experiment is substantially re-used and the modifications reported, the original author could be named as a co-author on the new paper. This is not appropriate for middleware and simulation platform code (e.g. Player and Stage), where normal citation is enough, but rather when the code that embodies the idea of a specific experiment is inherited.

10.7 The Trend Toward Experiment Publishing

We have argued that publishing code and experimental data is important for the robotics research community, is good for researchers and easy to do. Yet it is not standard practice in our field. The idea of publishing experimental data and other artifacts beyond finished papers is not new but it seems to be becoming popular. Here we survey some government policies, practice in other scientific disciplines and editorial policies of high impact journals.

10.7.1 Government and Funding Agency Policies

The US National Science Foundation (NSF)...

> ...expects investigators to share with other researchers, at no more than incremental cost and within a reasonable time, the data, samples, physical collections and other supporting materials created or gathered in the course of the work. It also encourages awardees to share software and inventions or otherwise act to make the innovations they embody widely useful and usable.[121]

Since October 2003 the US National Institute of Health (NIH) has required grant applications for $500K per year and above to include a plan for data sharing or a statement why data sharing is not possible [122]. While the form of data sharing is not considered during the proposal assessment, the NIH sends a clear signal to encourage publication of data.

The 2003 Berlin Declaration on Open Access to Knowledge [123] may come to be seen as an important milestone. At the time of writing the declaration has been signed by 264 funding agencies, universities and research organizations, including CERN, the Chinese Academy of Sciences, the Indian National Science Academy, and the German Research Foundation. The declaration states:

> A complete version of the work and all supplemental materials [...] in an appropriate standard electronic format is deposited (and thus published) in at least one online repository using suitable technical standards (such as the Open Archive definitions) that is supported and maintained by an academic institution, scholarly society, government agency, or other well established organization that seeks to enable open access, unrestricted distribution, inter operability, and long-term archiving. [123]

In a 2004 statement by the Organization for Economic Co-operation and Development (OECD) numerous governments including those of North America and Europe agreed on a declaration on access to research data from public funding. The OECD recognizes that

> an optimum international exchange of data, information and knowledge contributes decisively to the advancement of scientific research and innovation. [...] Open access to, and unrestricted use of, data promotes scientific progress and facilitates the training of researchers. [124].

In 2007 the US government passed the "America COMPETES Act" [125] requiring federal civilian agencies that conduct scientific research to openly exchange data and results with other agencies, policymakers and the public.

All of these national and international governmental efforts are aimed at improving the quality and efficiency of the science performed at public expense, and each requires or requests that experimental data and artifacts are shared.

10.7.2 Practice in non-robotics disciplines

According to Nielsen [126] since 1991 physicists have made extensive use of the preprint server *arXiv*, which makes papers freely available at the same time as they are submitted to a journal for publication. Nielsen views *arXiv* as an important tool to speed up the transfer of knowledge, but goes further by calling for the next generation of openness in science by "... making more types of content available than just scientific papers; allowing creative reuse and modification of existing work through more open licensing.".

Arguably the life sciences are leading the trend. For example a US National Academy of Science document on life science best practice [127] requires authors to be consistent with the principles of publication. This means that anything that is central to a paper is to be made available in a way that enables replication, verification and furtherance of science. When it comes to publishing algorithms, these guidelines are very explicit:

> ...if the intricacies of the algorithm make it difficult to describe in a publication, the author could provide an outline of it in the paper and make the source code [...] available to investigators... [127]

In epidemiology usually more is at stake than in everyday robotics. Epidemiological findings often influence policymakers, thus society requires highly reliable results from this field. Peng et al. [128] acknowledge the sensitive nature of this kind of work, and argue that *reproducibility* is the minimum standard for epidemiological research. Reproducibility allows independent investigators to subject the original data to their own analysis and interpretation. To enable reproducibility Peng

> ...calls for data sets and software to be made available for 1) verifying published findings, 2) conducting alternative analyzes of the same data, 3) eliminating uninformed criticisms that do not stand up to existing data, and 4) expediting the interchange of ideas among investigators. [128]

10.7.3 Journals

While scientists like Leonardo da Vinci, Galileo Galilei and Christiaan Huygens kept their discoveries secret [126], modern science is characterized by publication. Britain's Royal Society mandated peer-review of published scientific articles in its Philosophical Transactions, first published in 1665. This policy reflected the philosphy of the Royal Society, expressed in their motto *Nullius in Verba* (nothing in words / take nobody's word for it), that scientific claims are only valid if reproducible. Since then, peer-reviewed journals have been the most important way to communicate scientific results. Now, as the cost of distributing large amounts of digital data

becomes very low - a small fraction of the total cost of an experiment, or of paper publication - many journals require or at least encourage publication of data, samples, code and detailed method descriptions alongside the traditional paper.

Science

In 2004 and 2005 *Science* published two stem cell papers (*Science* 303, 1669 (2004) and *Science* 308, 1777 (2005)) which were later discovered to be fraudulent and retracted by the journal. As a consequence *Science* enlisted the help of an outside committee to investigate the handling of the two papers and suggest improvements to the editorial process of their journal. The committee concluded that *Science* had correctly followed their policies and that no procedure could protect against deliberate fraud [129]. Interesting in the context of our paper is the committee's recommendation for improving the editorial process "*Science* should have substantially stricter requirements about reporting the primary data". Today *Science* requires that

- large data sets are deposited in approved public databases prior to publication and an accession number is being included in the published paper.
- all data necessary to understand, assess, and extend the conclusions of the paper must be made available to the reader, flowing the policies of [121] and [127].
- all reasonable requests for sharing materials are to be fulfilled. [130]

Nature

The Nature Publishing Group's policy on availability of data for all their *Nature* publications is very similar:

> An inherent principle of publication is that others should be able to replicate and build upon the authors' published claims. Therefore, a condition of publication in a *Nature* journal is that authors are required to make material, data and associated protocols promptly available to readers without preconditions. [131]

As with *Science*, *Nature* requires depositing dataset in publicly accessible databases. Of high interest in relation to our paper is *Nature's* policy on sharing biological materials. It reads

> For materials such as mutant strains and cell lines, the *Nature* journals require authors to use established public repositories whenever possible [...] and provide accession numbers in the manuscript. [131]

Nature recently highlighted the importance of this issue on the cover of an issue that featured a related editorial and three news/opinion articles. The editorial described "data's shameful neglect", calling for funding agencies to boost support for (and pressure on) researchers to make data available [132]. Various reasons why many researchers choose not to share, despite the existence of purpose-built infrastructure, are discussed in the context of the perceived failure of a digital archive project at the University of Rochester[133]. A distinction is made between the issues of pre-publication [134] and post-publication [135] sharing of data and tools. We are advocating *simultaneous* sharing and publication, which is a special case of post-publication sharing.

Others

Other journals like Nucleic Acids Research [136] and the Public Library of Science journal series [137] have very similar policies on data sharing and access to research material.

A less rigorous procedure is employed by the Annals of Internal Medicine. To foster reproducible research and to enhance trust in scientific results of publications, the journal encourages authors to make their data publicly available and mandates authors to include a statement of whether materials are being made available or not and if under which conditions [138].

Robotics

The journal Autonomous Robots appears to have no stated policy on code and data publication, though it does support the bundling of supplementary material such as videos and data spreadsheets. The IEEE Transactions on Robotics is similar, with no policy recommending data or code sharing. However, it does mention these explicitly in the "multimedia" instructions:

> Multimedia can be "playable" files [...] or "dataset" files (e.g., raw data with programs to manipulate them). Such material is intended to enhance the contents of a paper, both in clarity and in added value.

10.7.4 Impact on citation rates

Citation counts are commonly used to assess the impact of an author's work [139]. A 2007 meta-study of cancer micorarray clinical trials revealed that papers which shared their mircoarray data were cited about 70% more frequently [140] than those that did not. If this effect generalizes to robotics, authors would have an interest in publishing code in order to boost their citation counts. An increase in citations can also be achieved by publishing in open-access journals [141].

10.7.5 Related attempts

A system for sharing and reproducing computations is proposed by Schwab et al. [142], who use their system, based on GNU *make*, as the principal means for organizing and transferring scientific computations in their geophysics laboratory. The motivation for ReDoc is essentially the same as that described in this paper. However, perhaps unfortunately, this tool does not appear to have made a large impact in computer science so far. ReDoc is clever and powerful, but requires the user to learn to read special Make macros. The method we advocate in this paper is more simple and does not prescribe a particular build system, and can be thought of as a subset of the ReDoc workflow.

10.8 Conclusion

We have shown that meta-government organizations like the OECD see scientific data exchange as an important tool for the efficient advancement of scientific research. We have also argued that code is a form of data that is particularly important for robotics research. Thus making experimental data including code and configurations publicly available is **important** for the progress of robotics.

Building upon other peoples work is an integral part of the scientific process. Increasing the efficiency of this process increases community productivity. We suggest that making experimental code identifiable will be helpful. We have also argued that the direct and indirect effects of publishing identifiable code are **good** for the researcher that shares, as well as for the wider community.

Other research fields, especially life sciences, have strong requirements to share data sets and provide free access to supporting materials alongside with traditional paper publications. In the case of *Nature* a submission of biological material to a public repository may be required. In robotics, while sharing physical robots may be prohibitively expensive, sharing digital resources takes little time or treasure. Freely available infrastructure allows the upload of a complete experiment (software, build scripts, data, analysis scripts etc.) to a public repository in a few seconds with a few button presses. Code sharing in robotics is **easy**.

The issues discussed here are not new, and a subset of robotics researchers does publish source code implementations of their algorithms, to the benefit of everyone. The main contribution here is to point out the importance of distributing uniquely identifiable versions and

not just the latest and "best" version. We have argued that this methodological issue is important, that originators and subsequent users can both benefit, and suggested an easy-to-follow publishing protocol. Our group will follow this protocol and observe its effects.

Acknowledgements

Thanks to Greg Mori, Alex Couture-Beil, Yaroslav Litus, Brian Gerkey, Gaurav Sukhatme, and the organizers and attendees of the RSS'09 Workshop on Methodology in Experimental Robotics for useful discussions on this issue.

Chapter 11

Selecting and Commanding Individual Robots in a Multi-Robot System

Alex Couture-Beil, Richard T. Vaughan and Greg Mori [5]

11.1 Abstract

We present a novel real-time computer vision-based system for facilitating interactions between a single human and a multi-robot system: a user first selects an individual robot from a group of robots, by simply looking at it, and then commands the selected robot with a motion-based gesture. Robots estimate which robot the user is looking at by performing a distributed leader election based on the "score" of the detected frontal face.

11.2 Introduction

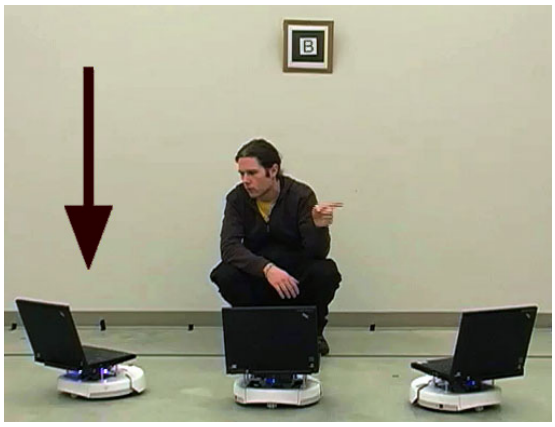
Selecting and commanding individual robots in a multi-robot system can be a challenge: interactions typically occur over a conventional human-computer interface (e.g. [143]), or specialized remote control (e.g. [144]). Humans, however, can easily select and command one another in large groups using only eye contact and gestures. Can similar non-verbal communication channels be used for human-robot interactions?

In this work, we describe a novel human-robot interface designed to use face engagement as a method for selecting a particular robot from a group of robots. Face detection is performed by each robot; the resulting score of the detected face is used in a distributed leader election algorithm to guarantee a single robot is selected. Once selected, motion-based gestures are used to assign tasks to the robot. In our demonstration, robots are commanded to drive to one of two predefined locations. An example of a typical interaction is shown in Fig 11.1; a video demonstration, from which these images originate, can be seen in [145].

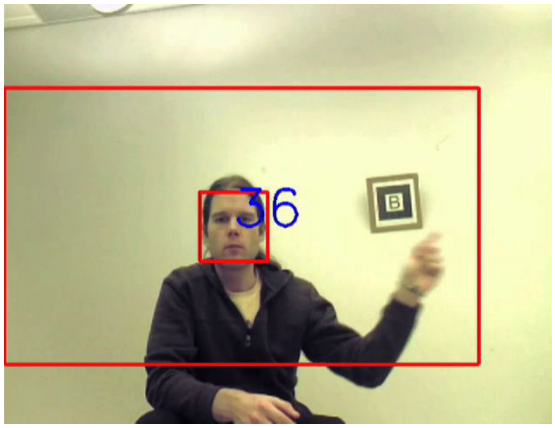
The system presented in this paper, which uses face engagement to select a particular robot from a multi-robot system, is the first of its kind.

11.3 Background

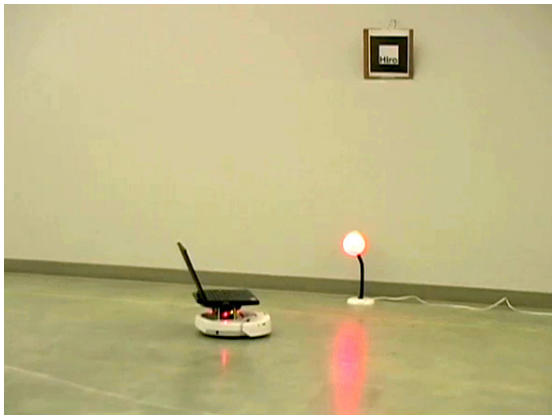
Before two or more people can enter into a focused interaction, they must somehow mutually signal their cognitive focus and readiness. Eye contact and eye gaze play an important role in



(a) A user selects an individual robot by looking at it, and assigns it a task by waving his hand.



(b) A user-centric region is identified; a learned classifier uses optical flow from the region to discriminate between gestures



(c) Robots travel to one of two zones as commanded by the user; colours are only used to illustrate different zones – robots use fiducial markers for localization

Figure 11.1: An example of selecting and commanding an individual robot from a group of robots.

initiating and regulating communication between people [146]. Throughout this work, we will use the term *face engagement*, as coined by Goffman, to describe the process in which people use eye contact, gaze and facial gestures to interact with or engage each other [147].

The role of eye contact plays such an important role in the development of humans that the ability to detect eye contact is present at birth [148]. We therefore believe that face engagement could be an effective non-verbal communication channel for human-robot interactions.

11.3.1 Gaze as an input device

There is a large literature on gaze tracking techniques; Morimoto and Mimica provide an in-depth survey [149]. Applications of gaze trackers can be found in fields ranging from psychology to marketing to computing science; many interesting examples are given in the survey provided by Duchowski [150].

In addition to tracking participants' gazes for subsequent analysis in usability studies, the human-computer interaction (HCI) community has studied using eye gaze tracking devices as hands-free real-time input devices (e.g. [151, 152]).

11.3.2 Gaze and interactive robots

Researchers argue that anthropomorphizing robots, and therefore exploiting human familiarity, will lead to more natural human-robot interactions; however *too* much anthropomorphization may lead to unrealistic prior expectations [153].

In an experiment by Mutlu et al., gaze is used to regulate conversations between Robovie, a humanoid robot with two controllable eyes[1], and two human participants. Their study showed that a) participants who made eye contact with Robovie liked the robot significantly more than those who were never acknowledged by Robovie's gaze, and b) gaze was an effective tool for yielding speaking turns and reinforcing conversation roles [155].

Besides yielding speaking roles and regulating conversation, gaze can also be used to establish joint attention between a speaker and addressee. The experiments of computation linguists Staudte and Crocker showed that people's cognitive response times increased when a robot used both gaze and speech to refer to objects presented on a table; however, when the robot was programmed to gaze incorrectly (at an irrelavant object), response times were significantly slower than when only speech was used [156]. Similar work by Mutlu et al. showed that participants were capable of picking up nonverbal leakage, that is, seemingly unintentional cues containing information, in a guessing game between a single human and Robovie [157].

Kuno et al. present a museum tour-guide that only responds when directly looked at [158]. Rather than truly performing gaze detection, their "eye-contact" detector, or perhaps what should be referred to as a "face engagement" detector relies on the detection of frontal faces. A telephoto lens is used to capture a high quality image; the robot then estimates if the user is looking at it by detecting if the nostrils are centered between the eyes. A similar method is used by Yonezawa et al. which detects the positions of a user's pupils before responding to voice commands [159].

Literature on eye gaze or face engagement aware human-robot interfaces is limited. While some of the robots discussed here only respond when looked at, it is not absolutely clear how precise their gaze tracking system is or how well it would fare in multi robot situations.

11.3.3 Robot selection and task delegation

There is little work on human-robot interfaces for multi-robot systems. Examples can be broken up into two general cases:

[1] Robovie was developed by Ishiguro et al. at ATR [154]

World-embodied interactions

World-embodied interactions occur directly between the human and robot, through either physical or sensor-mediated interfaces. These interfaces allow the user to walk freely among the robots, and does not require any form of robot localization. Examples include work by Payton that uses an omnidirectional IR LED to broadcast messages to all robots, and a narrow, directional IR LED to select and command individual robots [144], work by Naghsh et al. present a similar system designed for firefighters, but does not discuss selecting individual robots [160], and work by Zhao et al. which proposes the user interacts with the environment by leaving fiducial-based "notes" (for example, "vacuum the floor" or "mop the floor") for the robots at work site locations [161].

Traditional human-computer interfaces

Rather than interacting directly with robots, a traditional human-computer interface is used to represent the spatial configuration of the robots and allow the user to remotely interact with the robots. Examples of human-robot interactions which occur through a traditional interface include work by McLurkin et al. that presents a overhead-view of the swarm in a traditional point and click GUI named "SwarmCraft", and work by Kato that displays an overhead live video feed of the system on an interactive multi-touch computer table, which users can control the robots' paths by drawing a vector field over top of the world [162].

11.3.4 Gesture-based robot interaction

There is a vast computer vision literature on the gesture recognition domain: Mitra and Acharya [163] provide a survey. Several gesture-based robot interfaces exist; we do not attempt to provide an exhaustive survey, but rather mention some interesting examples. Systems may use static gestures – where the user holds a certain pose or configuration – or dynamic gestures – where the user performs a combination of actions.

Waldheer at al. use both static and motion-based gestures to control a trash-collecting robot [164]. Loper et al. demonstrate a indoor/outdoor person-following robot that uses an active depth sensing camera to recognize static gestures [165]. Earlier work by Kortenkamp et al. presents a mobile robot that uses an active vision system to recognize static gestures by building a skeleton model of the human operator; a vector of the human's arm is used to direct the robot to a particular point [166]. Perzanowski et al. present a multimodal speech and gesture-based interface; an active vision system is used to interpret pointing gestures as directional vectors, and to measure distance between the user's two hands [167]. In a subsequent paper, Perzanowski et al. discuss the idea of using gaze for directing an utterance at a particular robot; however instead they choose to use a unique name to verbally select a robot [168].

All gesture-based systems discussed so far are designed to work with a single robot, with exception of the work of Perzanowski et al.; however, there are no examples of gesture-based interfaces designed for multi-robot systems which rely solely on non-verbal communication. In this paper, we present such a system: a user first selects an individual robot with face engagement, then uses motion-based gestures to command it. Our novel system allows a user to interact with multiple robots in a shared environment by only using visual cues.

11.4 Robot selection

Before assigning a task to an individual robot, the human operator must first somehow designate a particular robot of interest as the selected robot he or she will be addressing. We will refer to this as the *robot selection problem*: how does a user interact with a particular robot within a group of robots without accidentally selecting or issuing commands to multiple robots.

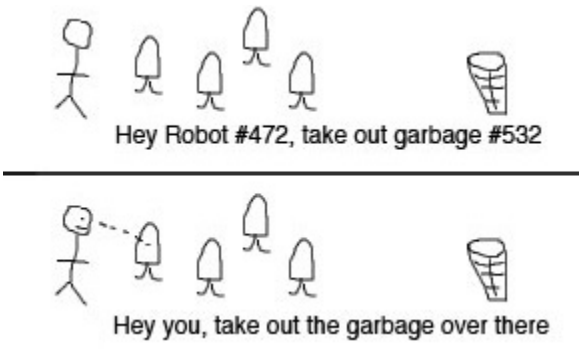
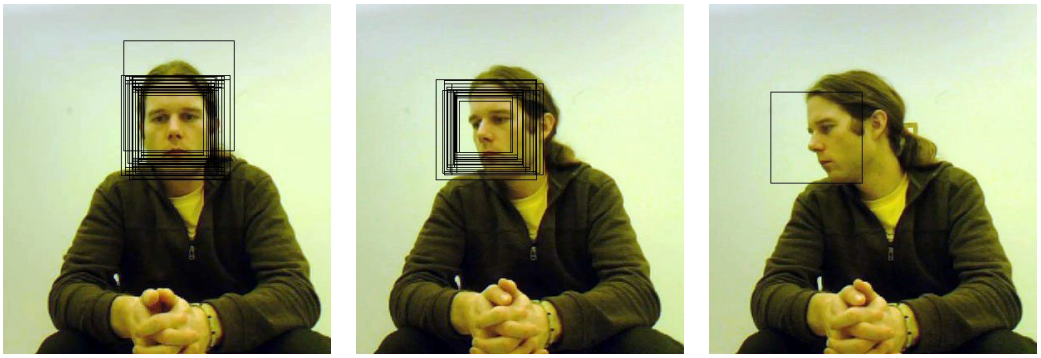


Figure 11.2: We suggest using face engagement is much more natural than a unique identifier



(a) front - score: 34 (b) 45 degree - score: 19 (c) side - score: 1

Figure 11.3: Candidate rectangles detected by the OpenCV Haar classifier cascade for frontal faces. The number of candidate rectangles are used to indicate how likely the face is a frontal face.

The difficulty of the robot selection problem depends on the particular human-robot interface of the system. For example, interfaces that have physical buttons or touch screens located on each robot are immune to the problem since there is no disambiguation when the user issues a command to the robot. In this case a private communication channel exists between the human and each robot since the user must physically approach and touch each robot. However, systems that do not have a private communication channel for each robot and rely on broadcasting commands through a shared medium are susceptible to the robot selection problem. These media include audio, infrared, radio and vision. Most systems assign a unique name or identifier which can be used to specify which robot the message is intended for. The Internet, for example, uses IP addresses to deliver a message over a shared medium to a particular computer; however, while IP addresses are easy for computer-to-computer communication, long unique identifiers are not appropriate for HRI as suggested by Fig 11.2. Assigning names to each robot (akin to the idea of hostnames) would provide a more usable interface; however users would still have to learn each robot's name.

Our approach to the robot selection problem is focused on maintaining face to face communication between a human and an individual robot. Face engagement serves as the means for designating *that* robot (the one the user is looking at) as the selected robot. However, since face engagement occurs in a shared communication channel between the user and all robots within line of sight, the robots must collectively agree upon a single robot designation to ensure only one robot will ever respond to the user at any given time.

The human operator should be able to select and interact with a robot at different distances; however, implementing eye gaze on a mobile robot for use at larger distances can be a costly endeavour since the use of a telephoto lens or high resolution camera must be used to capture a high quality image of the human's eyes [169]. Our system, on the other hand, use face detection rather than estimating eye gaze; this allows us to use smaller (and cheaper) cameras without zooming capabilities. Our system assumes only a single human will be interacting with the system at any given time; however, this single human will be simultaneously visible to multiple robots. Our system is designed to work at distances varying from 1 to 4 meters. The challenging aspect of our proposed solution to the robot selection problem is disambiguating which robot is currently being looked at through means of a distributed leader election algorithm based on the score of the detected face.

11.4.1 Face detection

The first phase of robot selection involves face detection. Each robot is equipped with a Lenovo ThinkPad R61 7744 laptop with an Intel Core 2 Duo 2.2GHz dual-core processor and 2GB of memory; we use the built in 640x480 resolution video-camera to capture images. Given an image such as the one presented in Fig 11.3(a), we are interested in locating a rectangular region in the image that contains a face. Furthermore, we want to extract a corresponding score indicating how likely is it that a frontal face has been detected.

Faces are detected with the Viola-Jones method [170]. We use an implementation provided by the OpenCV software library [171].

11.4.2 Face score

The face detector is trained on frontal faces only[2]. Therefore, the best matches occur when the detected face is looking directly at the camera. Since the face detector is insensitive to small changes in scale or position, multiple sub-windows are often clustered around faces. We use the number of neighbouring sub-windows in each cluster as a score to assess the quality of the detected face[3]. The score, however, does not necessarily indicate how frontal the face is. An obscured frontal face, for example, may receive a lower score than a visible and well lit non-frontal face. However, if the *same* face is captured simultaneously by multiple cameras (and thus under the same lighting conditions), then the scores *can* be used to detect the most frontal face. This observation is a novel contribution of this work.

Fig 11.3 provides an example of three different images of a person looking in three different directions. A frontal face is captured in the first image (Fig 11.3(a)) which has the highest score; as the person looks away from the camera the score decreases. In the extreme case where only a profile of the face is captured, the face is barely detected.

11.4.3 Leader election

The second phase of our solution to the robot selection problem is to perform a distributed leader election algorithm; this ensures only a single robot will ever be designated as the selected robot. The election determines which robot is most likely being looked at "head-on" by the user, as estimated by the highest detected face score.

Since the user might be visible to multiple robots, it is crucial that only a single robot ever respond to the user at any given time. This in effect requires some form of mutual exclusion among the robots, which are hereafter referred to as nodes. To solve this problem we use a variation of the ring-based election algorithm first described by Chang and Roberts [172]. Each node is assigned a unique IP address, hereafter referred to as UID, and is located in a virtual

[2]We use the pretrained frontal face Haar classifier supplied with OpenCV

[3]The Viola-Jones classifier score could also be used; however, the OpenCV implementation makes it difficult to retrieve, and using multiple detections is arguably more robust.

(a) wave-left (b) wave-right (c) idle

Figure 11.4: Example frames from our robot-command dataset used for training

ring comprised of all other nodes. Each node creates tuple $(UID, score)$ and forwards it to its neighbour. When a tuple is received at a node it either:

1. recognizes itself as the elected node if the received tuple contains its own UID,
2. passes the unmodified tuple to the next node if the contained score is greater than its own, or
3. replaces the contents of the tuple with its own score and UID if the contained score is less than or equal to its own.

Even though our network is totally ordered, and could break any ties by comparing the UIDs, we choose to replace tuples with *equal* scores, therefore resulting in no elected robot. We do this to force the user to move closer to the intended robot, thus selecting the robot the user really wanted rather than arbitrarily breaking the tie.

11.5 Gesture recognition

Once a robot has been selected by the user (thus winning the election), it can then be commanded by the user with motion-based gestures. Our classifier uses motion cues to discriminate between different gestures. Examples of the set of gestures used to command the robots are shown in fig 11.4.

A detailed description of our classifier, with experimental results, is presented in [173]. Our algorithm is summarized:

1. Motion features are first calculated by computing the optical flow for each frame. The optical flow vector field F is then split into four non-negative channels F_{x^+}, F_{x^-}, F_{y^+}, F_{y^-} representing the half-wave rectified horizontal and vertical components of the flow; this process is similar to Efros et al. [174]. These channels are first box-filtered, to reduce sensitivity to small translations, then aggregated over a temporal history of the last k frames, for some k which is large enough to capture all frames from a gesture period[4].
2. Face detection is used to create a normalized, user-centric view; motion features within this user-centric rectangle are cropped and resized to 30×40; all channels are then flattened into a single vector v.
3. The aforementioned motion features describe the user's entire motion. Given the labelled training data, we have a multi-class classification problem. Using the multi-class boosting

[4]In practice, we set k to be the FPS of our capture data, i.e., the number of frames required to capture one second of history.

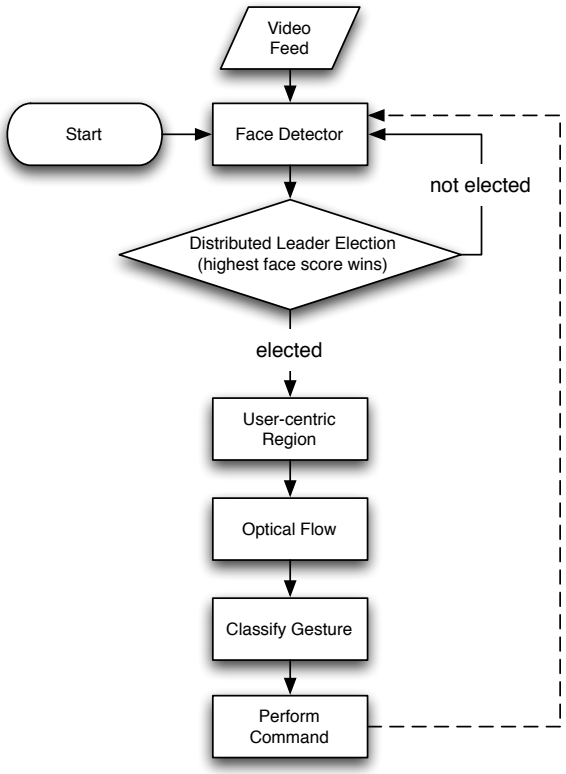


Figure 11.5: Flowchart of the leader election and motion-based gesture recognition process

algorithm AdaBoost.MH [175], we learn a discriminative classifier that only uses a subset of the motion feature vector v.

A schematic summary of the leader election and motion-based gesture recognition algorithms is provided in Fig 11.5.

11.6 The robots

We use three modified iRobot Create robots, pictured in Fig 11.1(a), which feature six IR range sensors, five colourful RGB LEDs, and a single-board Gumstix computer with an 802.11 wireless network adapter. The modified Creates, hereafter referred to as Chatterbox robots, were designed and built by the Autonomy Lab at SFU[5]. A laptop is mounted to each robot for video capture and processing as discussed in section 11.4.1.

11.6.1 Demonstration task

To demonstrate our system, we perform a robot navigation task. Three robots and a human operator are located in a 7x10 meter room clear of static obstacles. Robots navigate around the user and each other using the nearness diagram obstacle avoidance algorithm [176]. The robots: 1) first approach the user, who is located at a predefined location, 2) wait to be selected by the user. Once a robot has been selected by the user, it begins to glow random colours; the selected robot then 3) receive a command, and 4) travel to a predefined zone which corresponds to the issued command.

Robots either travel to the *red* zone or a *green* zone which corresponds to the received gesture: *wave-left* or *wave-right* respectively. Upon reaching the two meter wide circular zone,

[5]See `http://autonomy.cs.sfu.ca/robots.html` for more information.

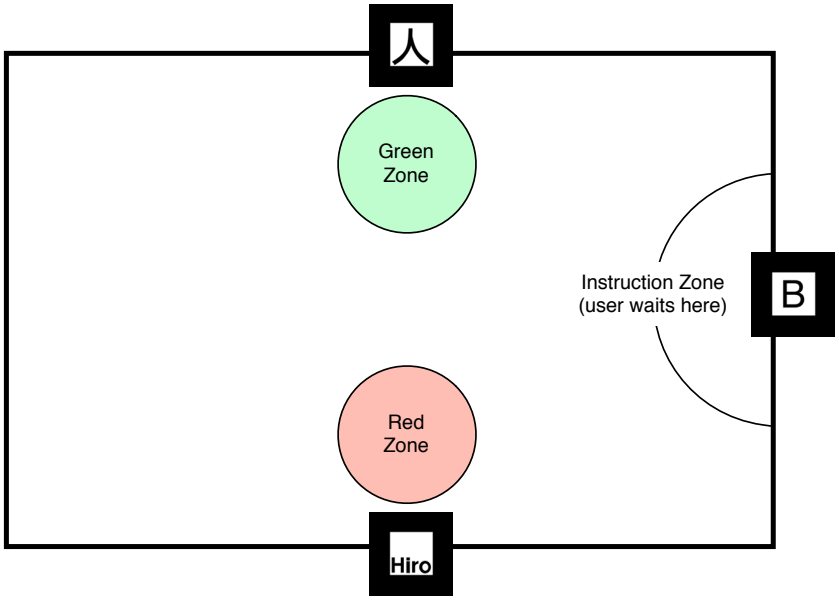


Figure 11.6: Robots wait at the instruction zone until they are selected and commanded to drive to either the red or green zone. Fiducial markers are placed on the walls near each zone.

each robot then return to the user to await a further command. Three unique ARtoolKit fiducial markers [177] provide global robot localization. An overview of the zones and room layout is shown in Fig 11.6.

11.7 Discussion

We have tested our system with 7 participants who were not involved with the development of the system. A single participant interacted with the robots at any given time. Each participant was instructed to:

1. first select a robot by looking at it, then, once the robot starts to glow;
2. direct the robot to one of two zones: a green zone, by waving your right hand, or a red zone, by waving your left hand.

Each participant was asked to command two robots to the same zone, and the third to the other zone. Once the robots reached their goal zone, they returned to the user. Participants were then encouraged to assign new tasks to returning robots as they saw fit. Some participants waited for all three robots to return before assigning tasks, where as others decided to immediately assign new tasks as each robot arrived.

Throughout the demonstration, a total number of 77 tasks were assigned to the robots. The next two sections are an informal description and discussion of the performance of the human-robot interface and is split up between robot selection, and task designation. A comprehensive user study will be the subject of future work.

11.7.1 Robot selection

Our leader election algorithm performed as intended: only a single robot ever responded, by glowing, at a given time. In some cases no robots were elected due to equal face scores resulting in a tie. In these cases, we encouraged participants to reposition themselves to break the tie.

Ties occurred when two robots awaiting instructions were located very close to each other; however, once the user approached a particular robot, the angle at which the user had to turn his or her head increased, resulting in a single robot seeing a full frontal face. In some cases, rather than re-engage one of the two side by side robots, users appeared to be discouraged by a tie and simply tried to select a third robot which they did not originally intend to select.

The face detector implementation provided with OpenCV worked well provided the faces were not too far away from the camera. In our demonstration, the user was at most 4 meters away from the camera and was in a well lit environment. Even with the camera pointing upwards towards the overhead lights, the system was still able to detect faces.

To provide visual feedback, the selected robot would glow with randomly alternating colours. Unfortunately the laptop partly covered the LED which made it hard to see for taller participants. We tried using the laptop's screen as a giant "LED" to provide feedback, but users reported that it was not as satisfying as the glowing LEDs.

Initially, two of the seven participants were unsure which robot was selected and issued commands before any robot had started to glow. However, after we encouraged them to move closer to the robot (and into its video-frame), these two participants were able to command the robots. In other cases, the robots were too close to both the wall and the user, ultimately forcing the user's back up against the wall. These localization-related problems forced the participants to squat down in order to be in the robot's field of view.

11.7.2 Task designation

Participants then assigned tasks to the selected robot with a motion-based hand gesture. We explicitly demonstrated the two gestures: left hand waving, and right hand waving, hereby referred to as wave-left and wave-right respectively. Using hand waving gestures to assign robots location dependent tasks proved to be challenging in three cases:

1. two of our participants, at first, extended their hands to point left or right rather than wave their hands in a continuous motion,

2. the full hand waving motion of participants who were located too close to the robot, was never captured by the video camera, and

3. the one second optical flow history window required for gesture recognition gave the interface a slow feel.

The accuracy of the system was good: 74 out of 77 commands were correctly executed. The 3 errors occurred when a user issued a command to a robot, and then quickly selected a different robot. This resulted in the newly selected robot classifying the previously issued gesture based on the motion features stored in the optical flow history window. This unintended behaviour could be remedied by reinitializing the optical flow history window whenever a robot is *not* elected.

To avoid classification errors, robots used a high classification threshold. Choosing a high threshold gives a high level of precision, which prevents the robots from incorrectly classifying a command resulting in opposite behaviour; however, setting the threshold too high limits our level of recall which became an irritant to some participants. After some exposure to the system participants were able to fine tune their gestures to achieve quicker recognition. Providing some sort of feedback mechanism may have decreased the interface's learning curve.

11.8 Conclusion

In this paper, we presented a computer vision-based human-robot interface for selecting and commanding an individual robot from a multi-robot system. A user first selects a robot with face engagement by simply looking at it. We employed a standard frontal face detector to detect

the user's face. The detected-face score of each robot is used in a distributed leader election algorithm to guarantee at most a single robot is selected. Once a robot has been selected by the user, it can then be commanded by using a motion-based gesture. We retrained a previously developed real-time classifier which uses motion-cues to discriminate between gestures corresponding to robot commands.

A demonstration task was described to investigate the feasibility of using face engagement and motion-based gestures for commanding an individual robot in a multi-robot system. Our demonstration showed that our face engagement-based leader-election could be effectively used to select an individual robot, which could then be commanded with motion-based gestures.

11.8.1 Future work

A proper user-study with a larger number of participants would be the next step for evaluating the system; however, the observations so far suggest some useful improvements: the human-robot interface could first be improved by providing a better feedback mechanism. The use of LEDs works well for quickly determining the current robot state; however, it would be valuable to see how users respond to an anthropomorphized robot with eyes. This could easily be implemented with virtual eyes on the laptop screen.

An extension to this system would be to allow users to first select a subset of the robots. The set of gestures could also be extended to allow a user to point to *any* arbitrary place in the environment, and have the robots drive to that location. This has been done for a single robot system (e.g. [166, 178]); however, a challenging task would be to coordinate multiple robots to cooperatively estimate the vector given the system's ability to simultaneously capture images of the user from multiple angles.

Bibliography

[1] R. T. Vaughan, Y. Litus, J. Wawerla, and A. Lein, "Autonomous sustain and resupply: Phase 1 report," tech. rep., Defense R&D Canada, 2008. Contract Scientific Authority Greg Broten. Copyright ©Her Majesty the Queen as represented by the Minister of National Defence, Canada.

[2] Y. Litus, P. Zebrowski, and R. T. Vaughan, "A distributed heuristic for energy-efficient multi-robot multi-place rendezvous," *IEEE Transactions on Robotics*, vol. 25, no. 1, pp. 130–135, 2009.

[3] R. T. Vaughan, "Massively multi-robot simulations in stage," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 189–208, 2008.

[4] R. T. Vaughan and J. Wawerla, "Publishing identifiable experiment code and configuration is important, good and easy.," *Autonomous Robots*, To appear, accepted subject to revision February 2010.

[5] A. Couture-Beil, R. T. Vaughan, and G. Mori, "Selecting and commanding individual robots in a vision-based multi-robot system," in *Proceedings of the Canadian Conference on Computer and Robot Vision*, May 2010. (to appear).

[6] Y. Litus and R. T. Vaughan, "Fall in! sorting a group of robots with a continuous controller," in *Proceedings of the Canadian Conference on Computer and Robot Vision*, May 2010. (to appear).

[7] J. Wawerla and R. T. Vaughan, "A fast and frugal method for team-task allocation in a multi-robot transportation system," in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2010. (to appear).

[8] A. Sadat and R. T. Vaughan, "Blinkered lost: Restricting sensor field of view can improve scalability in emergent multi-robot trail following," in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2010. (to appear).

[9] J. Wawerla and R. T. Vaughan, "Robot task switching under diminishing returns," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS'09)*, (St. Loius, MO), October 2009.

[10] A. Couture-Beil and R. T. Vaughan, "Adaptive mobile charging stations for multi-robot systems," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS'09)*, (St. Loius, MO), October 2009.

[11] A. Lein and R. T. Vaughan, "Adapting to non-uniform resource distributions in robotic swarm foraging through work-site relocation," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS'09)*, (St. Loius, MO), October 2009.

[12] J. Wawerla and R. T. Vaughan, "Optimal robot recharging for time discounted labour," in *Proceedings of the Eleventh International Conference on Artificial Life (ALife XI)*, August 2008.

[13] Y. Litus and R. T. Vaughan, "Distributed gradient optimization with embodied approximation," in *Proceedings of the Eleventh International Conference on Artificial Life (ALife XI)*, August 2008.

[14] A. Lein and R. T. Vaughan, "Adaptive multi-robot bucket brigade foraging," in *Proceedings of the Eleventh International Conference on Artificial Life (ALife XI)*, August 2008.

[15] J. Wawerla, *Task Switching in Self-Sufficient Robots*. PhD thesis, Simon Fraser University, 2010.

[16] Y. Litus, *USING SPATIAL EMBEDDEDNESS AND PHYSICAL EMBODIMENT FOR COMPUTATION IN MULTI-ROBOT SYSTEMS*. PhD thesis, Simon Fraser University, in preparation, 2010.

[17] A. Couture-Beil, R. T. Vaughan, and G. Mori, "Selecting and commanding individual robots in a vision-based multi-robot system," Master's thesis, Simon Fraser University, 2010.

[18] A. Lein, "Adaptive foraging in robotic swarms," Master's thesis, Simon Fraser University, 2010.

[19] D. W. Stephens, J. S. Brown, and R. C. Ydenberg, *Foraging*. Chicago: University of Chicago Press, 2007.

[20] P. Nonacs and J. L. Soriano, "Patch sampling behaviour and future foraging expectations in argentine ants, *linepithema humile*," *Animal Behavior*, vol. 55, pp. 519–527, March 1998.

[21] S. P. Hubbell, L. K. Johnson, E. Stanislav, B. Wilson, and H. Fowler, "Foraging by bucket-brigade in leaf-cutter ants," *Biotropica*, vol. 12, no. 3, pp. 210–213, 1980.

[22] S. H. D. M. Gordon, R. J. Schafer, "Forager activation and food availability in harvester ants," *Animal Behaviour*, vol. 71, pp. 815–822, April 2006.

[23] L. Panait and S. Luke, "A pheromone-based utility model for collaborative foraging," in *Proceedings of the 2004 Conference on Autonomous Agents and Multiagent Systems*, 2004.

[24] J. L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chrétien, "The dynamics of collective sorting: Robot-like ants and ant-like robots," in *Proceedings of the first international conference on simulation of adaptive behavior: From animals to animats*, (Cambridge, MA, USA), pp. 356–363, MIT Press, 1990.

[25] D. A. Shell and M. J. Matarić, "On foraging strategies for large-scale multi-robot systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Beijing, China.), pp. 2717–2723, Oct 2006.

[26] W. Liu, A. F. T. Winfield, J. Sa, J. Chen, and L. Dou, "Towards energy optimization: Emergent task allocation in a swarm of foraging robots," *Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems*, vol. 15, no. 3, pp. 289–305, 2007.

[27] M. S. Fontán and M. J. Matarić, "A study of territoriality: The role of critical mass in adaptive task division," in *In From Animals to Animats 4: Proceedings of the Fourth International Conference of Simulation of Adaptive Behavior*, pp. 553–561, MIT Press, 1996.

[28] A. Lein and R. Vaughan, "Adaptive multi-robot bucket-brigade foraging," in *Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems, Artificial Life*, pp. 337–342, MIT Press, August 2008.

[29] D. Goldberg and M. Matarić, "Maximizing Reward in a Non-Stationary Mobile Robot Environment," *Autonomous Agents and Multi-Agent Systems*, vol. 6, no. 3, pp. 287–316, 2003.

[30] E. Østergaard, G. Sukhatme, and M. Matarić, "Emergent bucket brigading: a simple mechanisms for improving performance in multi-robot constrained-space foraging tasks," in *Proceedings of the fifth international conference on Autonomous agents*, pp. 29–30, ACM Press New York, NY, USA, 2001.

[31] D. M. Gordon, J. Chu, A. Lillie, M. Tissot, and N. Pinter, "Variation in the transition from inside to outside work in the red harvester ant, *pogonomyrmex barbatus*," *Insectes Sociaux*, vol. 52, pp. 212–217, 2005.

[32] T. Schmickl and K. Crailsheim, "Trophallaxis among swarm-robots: A biologically inspired strategy for swarm robotics," in *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, 2006. BioRob 2006.*, pp. 377–382, 2006.

[33] C. R. Kube and E. Bonabeau, "Cooperative transport by ants and robots," *Robotics and Autonomous Systems*, vol. 30, pp. 85–101, 2000.

[34] C. Anderson, J. J. Broomsma, and I. J. J. Bartholdi, "Task partitioning in insect societies: bucket brigades," *Insected Sociaux*, vol. 49, pp. 171–180, 2002.

[35] R. T. Vaughan, K. Stóy, G. S. Sukhtame, and M. Matarić, "Whistling in the dark: Cooperative trail following in uncertain localization space," in *Agents*, (Barcelona, Spain), pp. 187–194, 2000.

[36] M. Zuluaga and R. Vaughan, "Reducing spatial interference in robot teams by local-investment aggression," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2798–2805, August 2005.

[37] R. A. Brooks, "Intelligence without representation," *Artificial Intelligence*, vol. 47, pp. 139–159, 1991.

[38] S. D. Fretwell and H. L. Lucas, "On territorial behavior and other factors influencing habitat distribution in birds," *Acta Biotheoretica*, vol. 19, no. 1, pp. 16–36, 1969.

[39] J. H. Fellers, "Interference and exploitation in a guild of woodland ants," *Ecology*, vol. 68, pp. 1466–1478, October 1987.

[40] J. Wawerla and R. T. Vaughan, "Optimal robot recharging strategies for time discounted labour," in *Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems, Artificial Life*, August 2008.

[41] A. Sadat and R. T. Vaughan, "So-lost: An ant-trail algorithm for multi-robot navigation with active interference reduction," in *Proceedings of the Twelfth International Conference on Artificial Life (ALife XII)*, August 2010.

[42] R. T. Vaughan, K. Støy, G. S. Sukhatme, and M. J. Matarić, "Whistling in the dark: cooperative trail following in uncertain localization space," in *Proceedings of the Fourth International Conference on Autonomous Agents*, (Barcelona, Spain), pp. 187–194, June 2000.

[43] R. T. Vaughan, K. Støy, A. Howard, G. Sukhatme, and M. J. Matarić, "Lost: Localization-space trails for robot teams," *IEEE Transactions on Robotics and Autonomous Systems*, vol. 18, no. 5, pp. 796–812, 2002.

[44] M. Dorigo, *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992.

[45] P. S. Heck and S. Ghosh, "The design and role of synthetic creative traits in artificial ant colonies," *J. Intell. Robotics Syst.*, vol. 33, no. 4, pp. 343–370, 2002.

[46] A. Russell, D. Thiel, R. Deveza, and A. Mackay-Sim, "Sensing odour trails for mobile robot navigation," in *Proc. Int. Conf Robotics and Automation*, pp. 2672–2677, IEEE, 1994.

[47] R. Fujisawa, H. Imamura, T. Hashimoto, and F. Matsuno, "Communication using pheromone field for multiple robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1391–1396, Sept. 2008.

[48] D. Payton, M. Daily, R. Estowski, M. Howard, and C. Lee, "Pheromone robotics," *Auton. Robots*, vol. 11, no. 3, pp. 319–324, 2001.

[49] K. Lerman and A. Galstyan, "Mathematical model of foraging in a group of robots: Effect of interference," *Auton. Robots*, vol. 13, no. 2, pp. 127–141, 2002.

[50] M. Zuluaga and R. Vaughan, "Reducing spatial interference in robot teams by local-investment aggression," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Edmonton, Alberta), August 2005.

[51] M. M. Alexander Scheidler, Daniel Merkle, "Congestion control in ant like moving agent systems," *Biologically-Inspired Collaborative Computing*, vol. 268, pp. 33–43, 2008.

[52] M. Zuluaga and R. T. Vaughan, "Modeling multi-robot interaction using generalized occupancy grids, with application to reducing spatial interference," in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2008.

[53] S. Camazine, N. R. Franks, J. Sneyd, E. Bonabeau, J.-L. Deneubourg, and G. Theraula, *Self-Organization in Biological Systems*. Princeton, NJ, USA: Princeton University Press, 2001.

[54] Y. Litus and R. Vaughan, "Distributed gradient optimization with embodied approximation," in *Proc. Int. Conf. on Simulation and Synthesis of Living Systems* (S. Bullock, J. Noble, R. Watson, and M. A. Bedau, eds.), pp. 359–365, MIT Press, Cambridge, MA, 2008.

[55] H. Hamann and H. Wörn, "Embodied computation," *Parallel Processing Letters*, vol. 17, pp. 287 – 298, Sept. 2007.

[56] R. A. Brooks, "Elephants don't play chess," *Robotics and Autonomous Systems*, vol. 6, pp. 3–15, June 1990.

[57] C. Paul, "Morphology and computation," in *Proc. Int. Conf. on Simulation of Adaptive Behavior*, pp. 33–38, MIT Press, 2004.

[58] R. Pfeifer, F. Iida, and G. Gómez, "Morphological computation for adaptive behavior and cognition," *International Congress Series*, vol. 1291, pp. 22–29, 2006.

[59] R. Brockett, "Dynamical systems that sort lists, diagonalize matrices and solve linear programming problems," in *Decision and Control, 1988., Proceedings of the 27th IEEE Conference on*, pp. 799–803 vol.1, Dec 1988.

[60] Y. Kodama and B. Shipman, "The finite non-periodic toda lattice: A geometric and topological viewpoint," 2008.

[61] N. Saxena and J. Clark, "Analogue system for eigenvalue computation and sorting based on an isospectral matrix flow," *Electronics Letters*, vol. 31, pp. 24–26, Jan 1995.

[62] M. Zavlanos and G. Pappas, "A dynamical systems approach to weighted graph matching," in *Decision and Control, 2006 45th IEEE Conference on*, pp. 3492–3497, Dec. 2006.

[63] A. M. Bloch and P. E. Crouch, "Optimal control, optimization, and analytical mechanics," in *Mathematical control theory*, pp. 268–321, New York, NY, USA: Springer-Verlag New York, Inc., 1999.

[64] E. Bahceci, O. Soysal, and E. Sahin, "A review: Pattern formation and adaptation in multi-robot systems," Tech. Rep. CMU-RI-TR-03-43, Carnegie Mellon University, 2003.

[65] A. M. Bloch, R. W. Brockett, and T. S. Ratiu, "A new formulation of the generalized Toda lattice equations and their fixed point analysis via the momentum map," *Bulletin of the American Mathematical Society*, vol. 23, no. 2, pp. 477–485, 1990.

[66] M. Toda, "Vibration of a chain with nonlinear interaction," *Journal of the Physical Society of Japan*, vol. 22, no. 2, pp. 431–436, 1967.

[67] R. T. Vaughan, "Massively multi-robot simulations in Stage," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 189–208, 2008.

[68] M. Zavlanos and G. Pappas, "Dynamic assignment in distributed motion planning with local coordination," *Robotics, IEEE Transactions on*, vol. 24, pp. 232–242, Feb. 2008.

[69] M. A. Pruett-Jones and S. G. Pruett-Jones, "Food caching in the tropical frugivore, MacGregor's bowerbird (amblyornis macgregoriae)," *The Auk*, vol. 102, pp. 334–341, Apr. 1985.

[70] P. Zebrowski and R. Vaughan, "Recharging robot teams: A tanker approach," in *Proceedings of the International Conference on Advanced Robotics (ICAR)*, (Seattle, Washington), July 2005.

[71] "iRobot home base," 2008.

[72] "Robot docking / charging system," 2008.

[73] M. C. Silverman, D. Nies, B. Jung, and G. S. Sukhatme, "Staying alive: a docking station for autonomous robot recharging," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, vol. 1, pp. 1050–1055, May 2002.

[74] J. Wawerla and R. T. Vaughan, "Near-optimal mobile robot recharging with the rate-maximizing forager," in *ECAL*, pp. 776–785, 2007.

[75] G. Parker and O. Izmirli, "Choosing a charging station using sound in colony robotics," in *World Automation Congress, 2006. WAC '06*, (Budapest), pp. 1–6, July 2006.

[76] E. Takeuchi and T. Tsubouchi, "Portable effector docking mechanism for a service mobile robot and its positioning," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 3380–3386, May 2006.

[77] A. Drenner and N. Papanikolopoulos, "Docking station relocation for maximizing longevity of distributed robotic teams," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 2436–2441, May 2006.

[78] Y. Litus, R. T. Vaughan, and P. Zebrowski, "The frugal feeding problem: Energy-efficient, multi-robot, multi-place rendezvous," in *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2007.

[79] D. Goldberg and M. J. Matarić, "Interference as a tool for designing and evaluating multi-robot controllers," in *In Proceedings, AAAI-97*, pp. 637–642, AAAI Press, 1997.

[80] M. Schneider-Fontan and M. J. Matarić, "Territorial multi-robot task division," *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 815–822, Oct. 1998.

[81] A. Munoz, F. Sempe, and A. Drogoul, "Sharing a charging station in collective robotics," Tech. Rep. LIP6 2002/026, 8, rue du Capitaine Scott, 75015 Paris, 2002.

[82] D. J. McFarland and E. Spier, "Basic cycles, utility and opportunism in self-sufficient robots," *Robotics and Autonomous Systems*, vol. 20, pp. 179–190, June 1997.

[83] D. W. Stephens and J. R. Krebs, *Foraging Theory*. Princeton University Press, 1986.

[84] D. W. Stephens, J. S. Brown, and R. C. Ydenberg, eds., *Foraging - Behavior and Ecology*. University of Chicago Press, 2007.

[85] W. Liu, A. F. T. Winfield, J. Sa, J. Chen, and L. Dou, "Towards energy optimization: Emergent task allocation in a swarm of foraging robots," *Adaptive Behavior*, vol. 15, no. 3, pp. 289–305, 2007.

[86] P. Ulam and T. Balch, "Using optimal foraging models to evaluate learned robotic foraging behavior," *Adaptive Behavior*, vol. 12, no. 3-4, pp. 213–222, 2004.

[87] E. Østergaard, G. S. Sukhatme, and M. J. Matarić, "Emergent bucket brigading - a simple mechanism for improving performance in multi-robot constrained-space foraging tasks," in *Proceedings of the International Conference on Autonomous Agents*, pp. 29–30, May 2001.

[88] D. A. Shell and M. J. Matarić, "On foraging strategies for large-scale multi-robot systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2717–2723, October 2006.

[89] A. Lein and R. T. Vaughan, "Adaptive multi-robot bucket brigade foraging," in *Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems, Artificial Life*, August 2008.

[90] B. W. Andrews, K. M. Passino, and T. A. Waite, "Foraging theory for autonomous vehicle decision-making system design," *Journal of Intelligent and Robotic Systems*, vol. 49, pp. 39–65, 2007.

[91] É. Danchin, L.-A. Giraldeau, and F. Cézilly, eds., *Behavioural Ecology*. Oxford University Press, 2008.

[92] E. L. Charnov, "Optimal foraging: Attack strategy of a mantid," *The American Naturalist*, vol. 110, pp. 141–151, January 1976.

[93] E. L. Charnov, "Optimal foraging, the marginal value theorem," *Journal of Theoretical Biology*, vol. 9, no. 2, pp. 129–135, 1976.

[94] R. F. Green, "Stopping rules for optimal foragers," *The American Naturalist*, vol. 123, pp. 30–43, January 1984.

[95] J. M. McNamara, "Optimal patch use in a stochastic environment," *Theoretical Population Biology*, vol. 21, no. 2, pp. 269–288, 1982.

[96] J. K. Waage, "Foraging for patchily-distributed hosts by the parasitoid, nemeritis canescens," *Animal Ecology*, vol. 48, no. 2, pp. 353–371, 1979.

[97] R. T. Vaughan, "Massively multi-robot simulations in Stage," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 189–208, 2008.

[98] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*. MIT Press, 1998.

[99] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *Proceedings of the 16th European Conference on Machine Learning (ECML)*, pp. 437–448, October 2005.

[100] J. Wawerla and R. T. Vaughan, "Online robot task switching under diminishing returns," in *Proceedings of the Twelfth International Conference on Artificial Life (ALife XII)*, August 2010.

[101] J. Wawerla and R. T. Vaughan, "Robot task switching under diminishing returns," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5033–5038, 2009.

[102] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng, "Cooperative mobile robotics: Antecedents and directions," *Autonomous Robots*, vol. 4, pp. 226–234, 1997.

[103] E. L. Charnov and G. H. Orians, "Optimal foraging: Some theoretical explorations." Unpublished manuscript `http://hdl.handle.net/1928/1649`, 1973.

[104] J. A. Gibb, "Predation by tits and squirrels on the eucosmid *Ernarmonia conicolana*," *Anim. Ecol.*, vol. 27, pp. 375–396, 1958.

[105] J. R. Krebs, *Perspectives in Ethology*, vol. 1, ch. Behavioral aspects of predation, pp. 73–111. Plenum New York, 1973.

[106] J. Krebs, J. Ryan, and E. Charnov, "Hunting by expectation or optimal foraging: A study of patch use by chickadees," *Animal Behaviour*, vol. 22, pp. 953–964, 1974.

[107] L. Real, S. Ellner, and L. D. Harder, "Short-term energy maximization and risk-aversion in bumble bees: A reply to Possingham et al.," *Ecology*, vol. 71, no. 4, pp. 1625–1628, 1990.

[108] Wikipedia, "Port of Vancouver - Wikipedia, The Free Encyclopedia," 2009. `http://en.wikipedia.org/wiki/Port_of_vancouver` [Online; accessed 15-Sept-2009].

[109] E. Guizzo, "Three engineers, hundreds of robots, one warehouse," *IEEE Spectrum*, July 2008.

[110] B. P. Gerkey and M. J. Matarić, "Sold!: Auction methods for multi-robot coordination," *IEEE Transactions on Robotics and Automation, Special Issue on Multi-Robot Systems*, vol. 18, no. 5, pp. 758–768, 2002.

[111] T. H. Labella, M. Dorigo, and J.-L. Deneubourg, "Self-organised task allocation in a group of robots," in *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems*, pp. 371–380, 2004.

[112] M. J. B. Krieger and J.-B. Billeter, "The call of duty: Self-organised task allocation in a population of up to twelve mobile robots," *Robotics and Autonomous Systems*, vol. 30, pp. 65–84, 2000.

[113] C. V. Jones and M. J. Matarić, "Adaptive division of labor in large-scale minimalist multi-robot systems," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Las Vegas, Nevada), pp. 1969–1974, Oct 2003.

[114] T. S. Dahl, M. J. Matarić, and G. S. Sukhatme, "Multi-robot task allocation through vacancy chain scheduling," *Robotics and Autonomous Systems*, vol. 57, no. 6, pp. 674–687, 2009.

[115] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[116] J. Minguez and L. Montano, "Nearness diagram navigation (ND): Collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, pp. 45–59, 2004.

[117] A. K. Seth, "Competitive foraging, decision making, and the ecological rationality of the matching law," in *From Animals to Animats 7, 7th International Conference on Simulation of Adaptive Behavior, SAB*, (Cambridge, MA, USA), pp. 359–368, MIT Press, 2002.

[118] R. Rivest, "Rfc1321: The md5 message-digest algorithm," tech. rep., Internet Engineering Task Force: Network Working Group, 1992.

[119] X. Wang and H. Yu, "How to break md5 and other hash functions," in *Eurocrypt 2005*, vol. 3494, pp. 19–35, Lecture Notes in Computer Science, May 2005.

[120] B. P. Gerkey, R. T. Vaughan, K. Stoy, A. Howard, G. S. Sukhatme, and M. J. Matarić, "Most valuable player: A robot device server for distributed control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1226–1231, 2001.

[121] National Science Foundation, "Grant general conditions," 2001. `http://www.nsf.gov/pubs/2001/gc101/gc101rev1.pdf` [Online; accessed 09-June-2009].

[122] National Institues of Health Office of Extramural Research, "Nih data sharing policy," 2003. `http://grants.nih.gov/grants/policy/data_sharing/` [Online; assessed 09-June-2009].

[123] "Berlin declaration on open access to knowledge in sciences and humanities," 2003. `http://oa.mpg.de/openaccess-berlin/berlin_declaration.pdf` [Online; accessed 09-June 2009].

[124] Organisation for Economic Co-operation and Development, "Science, technology and innovation for the 21st. century. meeting of the OECD commitee for scientific and technological policy at ministerial level," January 2004. `http://www.oecd.org/document/0,2340,en_2649_34487_25998799_1_1_1_1,00.html` [Online; accessed 09-June-2009].

[125] US Congress, "America COMPETES Act," 2007. `http://commdocs.house.gov/reports/110/h2272.pdf` [Online; accessed 09-June 2009].

[126] M. Nielsen, "Doing science in the open," *Physicsworld*, vol. 22, no. 5, p. 30, 2009.

[127] T. R. Cech *et al.*, *Sharing Publication-Related Data and Materials: Responsibilities of Authorship in the Life Sciences*. National Academy of Sciences, 2003.

[128] R. D. Peng, F. Dominici, and S. L. Zeger, "Reproducible epidemiologic research," *American Journal of Epidemiology*, vol. 163, no. 9, pp. 783–789, 2006.

[129] D. Kennedy, "Responging to fraud," *Science*, vol. 314, p. 1353, December 2006.

[130] "General information for authors," 2009. `http://www.sciencemag.org/about/authors/prep/gen_info.dtl\#datadep` [Online; accessed 10-June-2009].

[131] Nature, "Guide to publication policies of the nature journals: Editorial policies," 2009. `http://www.nature.com/authors/gta.pdf` [Online; accessed 09-June-2009].

[132] Editorial, "Data's shameful neglect," *Nature*, vol. 461, p. 145, 10 September 2009.

[133] B. Nelson, "Data sharing: Empty archives," *Nature*, vol. 461, pp. 160–163, 10 September 2009.

[134] T. I. D. R. W. Authors, "Prepublication data sharing," *Nature*, vol. 461, pp. 168–170, 10 September 2009.

[135] P. Schofield, T. Bubela, T. Weaver, L. Portilla, S. Brown, J. Hancock, D. Einhorn, G. Tocchini-Valentini, M. Hrabe de Angelis, and N. Rosenthal, "Post-publication sharing of data and tools," *Nature*, vol. 461, pp. 171–173, 10 September 2009.

[136] Nucleic Acids Research, "General policies of the journal," 2009. `http://www.oxfordjournals.org/our_journals/nar/for_authors/ed_policy.html` [Online; accessed 09-June-2009].

[137] PLos, "Editorial and publishing policies," 2009. `http://www.plosone.org/static/policies.action` [Online; accessed 10-June-2009].

[138] C. Laine, S. N. Goodman, M. E. Griswold, and H. C. Sox, "Reproducible research: moving toward research the public can really trust," *Annals of Internal Medicine*, vol. 146, pp. 450–454, March 2007.

[139] A. M. Diamond Jr., "What is a citation worth," *Journal of Human Resources*, vol. 21, no. 2, pp. 200–215, 1986.

[140] H. A. Piwowar, R. S. Day, and D. B. Fridsma, "Sharing detailed research data is associated with increased citation rate," *PLoS ONE*, vol. 2, no. 3, p. e308, 2007.

[141] G. Eysenbach, "Citation advantage of open access articles," *PLoS Biology*, vol. 4, no. 5, p. e157, 2006.

[142] M. Schwab, M. Karrenbach, and J. Clearbout, "Making scientific computations reproducible," *Computing in Science and Engineering*, vol. 2, no. 6, pp. 61–67, 2000.

[143] J. McLurkin, J. Smith, J. Frankel, D. Sotkowitz, D. Blau, and B. Schmidt, "Speaking swarmish: Human-Robot interface design for large swarms of autonomous mobile robots," in *Proc. of the AAAI Spring Symposium*, 2006.

[144] D. Payton, "Pheromone robotics." `http://www.swarm-robotics.org/SAB04/presentations/payton-review.pdf`, 2004. Slides from a presentation at the Swarm Robotics Workshop, SAB04. Retrieved September 28, 2009.

[145] A. Couture-Beil, R. T. Vaughan, and G. Mori, "Selecting and commanding individual robots in a vision-based multi-robot system," in *HRI '10: Proc. of the 5th ACM/IEEE intl. conf. on Human robot interaction*, p. (to appear), 2010.

[146] A. Kendon, "Some functions of gaze-direction in social interaction," *Acta psychologica*, vol. 26, pp. 22–63, 1967.

[147] E. Goffman, *Behavior in Public Places: Notes on the Social Organization of Gatherings*. Free Press, September 1966.

[148] T. Farroni, G. Csibra, F. Simion, and M. H. Johnson, "Eye contact detection in humans from birth," in *Proc. of the National Academy of Sciences of the United States of America*, pp. 9602–9605, 1999.

[149] C. H. Morimoto and M. R. Mimica, "Eye gaze tracking techniques for interactive applications," *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 4–24, 2005.

[150] A. T. Duchowski, "A breadth-first survey of eye-tracking applications.," *Behavior Research Methods, Instruments, and Computers*, vol. 34, no. 4, pp. 455–470, 2002.

[151] R. J. K. Jacob and K. S. Karn, "Eye tracking in human-computer interaction and usability research: Ready to deliver the promises," *The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research*, pp. 573–603, 2003.

[152] I. Starker and R. A. Bolt, "A gaze-responsive self-disclosing display," in *CHI '90: Proc. of the SIGCHI conf. on Human factors in computing systems*, pp. 3–10, ACM, 1990.

[153] B. R. Duffy, "Anthropomorphism and the social robot," *Robotics and Autonomous Systems*, vol. 42, no. 3–4, pp. 177–190, 2003.

[154] H. Ishiguro, T. Ono, M. Imai, T. Maeda, T. Kanda, and R. Nakatsu, "Robovie: an interactive humanoid robot," *Industrial robot: An intl. journal*, vol. 28, no. 6, pp. 498–503, 2001.

[155] B. Mutlu, T. Shiwa, T. Kanda, H. Ishiguro, and N. Hagita, "Footing in human-robot conversations: how robots might shape participant roles using gaze cues," in *HRI '09: Proc. of the 4th ACM/IEEE intl. conf. on Human robot interaction*, pp. 61–68, ACM, 2009.

[156] M. Staudte and M. W. Crocker, "Visual attention in spoken human-robot interaction," in *HRI '09: Proc. of the 4th ACM/IEEE intl. conf. on Human robot interaction*, pp. 77–84, ACM, 2009.

[157] B. Mutlu, F. Yamaoka, T. Kanda, H. Ishiguro, and N. Hagita, "Nonverbal leakage in robots: communication of intentions through seemingly unintentional behavior," in *HRI '09: Proc. of the 4th ACM/IEEE intl. conf. on Human robot interaction*, pp. 69–76, ACM, 2009.

[158] Y. Kuno, M. Kawashima, K. Yamazaki, and A. Yamazaki, "Importance of vision in human-robot communication understanding speech using robot vision and demonstrating proper actions to human vision," in *Intelligent Environments*, Advanced Information and Knowledge Processing, pp. 183–202, Springer London, 2009.

[159] T. Yonezawa, H. Yamazoe, A. Utsumi, and S. Abe, "Evaluating crossmodal awareness of daily-partner robot to user's behaviors with gaze and utterance detection," in *Casemans '09: Proc. of the 3rd ACM intl. Workshop on Context-Awareness for Self-Managing Systems*, pp. 1–8, ACM, 2009.

[160] A. M. Naghsh, J. Gancet, A. Tanoto, and C. Roast, "Analysis and design of human-robot swarm interaction in firefighting," in *IEEE intl. Symposium on Robot and Human Interactive Communication (RO-MAN'08)*, pp. 255–260, 2008.

[161] S. Zhao, K. Nakamura, K. Ishii, and T. Igarashi, "Magic cards: a paper tag interface for implicit robot control," in *CHI '09: Proc. of the 27th intl. conf. on Human factors in computing systems*, pp. 173–182, ACM, 2009.

[162] J. Kato, D. Sakamoto, M. Inami, and T. Igarashi, "Multi-touch interface for controlling multiple mobile robots," in *Proc. of the 27th intl. conf. extended abstracts on Human factors in computing systems*, pp. 3443–3448, ACM, 2009.

[163] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Transactions On Systems, Man, And Cybernetics - Part C: Applications And Reviews*, vol. 37, pp. 311–324, May 2007.

[164] S. Waldherr, R. Romero, and S. Thrun, "A gesture based interface for human-robot interaction," *Autonomous Robots*, vol. 9, pp. 151–173, 2000.

[165] M. M. Loper, N. P. Koenig, S. H. Chernova, C. V. Jones, and O. C. Jenkins, "Mobile human-robot teaming with environmental tolerance," in *HRI '09: Proc. of the 4th ACM/IEEE intl. conf. on Human robot interaction*, pp. 157–164, ACM, 2009.

[166] D. Kortenkamp, E. Huber, R. P. Bonasso, and M. Inc, "Recognizing and interpreting gestures on a mobile robot," in *Proc. of the Thirteenth National conf. on Artificial Intelligence*, pp. 915–921, AAAI Press/The MIT Press, 1996.

[167] D. Perzanowski, A. C. Schultz, W. Adams, E. Marsh, and M. Bugajska, "Building a multimodal human-robot interface," *Intelligent Systems, IEEE*, vol. 16, pp. 16–21, Jan-Feb 2001.

[168] D. Perzanowski, A. C. Schultz, W. Adams, M. Bugajska, E. Marsh, G. Trafton, D. Brock, M. Skubic, and M. Abramson, "Communicating with teams of cooperative robots," in *Proc. from the 2002 NRL Workshop on Multi-Robot Systems*, pp. 16–20, Kluwer, 2002.

[169] H. Yamazoe, A. Utsumi, T. Yonezawa, and S. Abe, "Remote gaze estimation with a single camera based on facial-feature tracking without special calibration actions," in *ETRA '08: Proc. of the 2008 symposium on Eye tracking research & applications*, pp. 245–250, ACM, 2008.

[170] P. Viola and M. Jones, "Robust real-time face detection," *intl. Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[171] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, 2008.

[172] E. J. H. Chang and R. Roberts, "An improved algorithm for decentralized extrema-finding in circular configurations of processes," *Communications of the ACM*, vol. 22, pp. 281–283, May 1979.

[173] M. Bayazit, A. Couture-Beil, and G. Mori, "Real-time motion-based gesture recognition using the gpu," in *Proc. of the IAPR conf. on Machine Vision Applications*, pp. 9–12, May 2009.

[174] A. A. Efros, A. C. Berg, G. Mori, and J. Malik, "Recognizing action at a distance," in *Proc. 9th Int. Conf. Computer Vision*, vol. 2, pp. 726–733, 2003.

[175] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.

[176] J. Minguez, A. Member, and L. Montano, "Nearness diagram (nd) navigation: Collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics and Automation*, vol. 20, p. 2004, 2004.

[177] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in *Proc. of the 2nd IEEE and ACM intl. Workshop on Augmented Reality*, p. 85, IEEE Computer Society, 1999.

[178] C. Martin, F.-F. Steege, and H.-M. Gross, "Estimation of pointing poses for visual instructing mobile robots under real-world conditions," *Robotics and Autonomous Systems*, vol. 57, p. (to appear), 2009.