Whistling in the Dark: Cooperative Trail Following in Uncertain Localization Space

Richard T. Vaughan, Kasper Støy, Gaurav S. Sukhatme and Maja J. Matarić

Robotics Research Laboratories
University of Southern California
Los Angeles, California 90089-0781

[vaughan, kaspers, gaurav, mataric]@robotics.usc.edu

ABSTRACT

We demonstrate that a simulated group of robots can cooperate to robustly transport resource between two areas in an unknown environment using an algorithm inspired by the trail following of ants and the waggle dance of honey bees. Rather than directly marking their environment, the robots announce their successful paths through a common localization space. It is found that the algorithm is robust to significant localization error, suggesting that the method will be viable for teams of real robots.

1. INTRODUCTION

Ants form supply columns to relocate valuable items through complex, dynamic environments. Their remarkable effectiveness is due to a highly robust strategy of local interactions among a large number of autonomous agents. The chemical trails formed by ants along their supply routes are robust with respect to changes in the environment and to the 'failure' of many individual ants [11; 16]. These properties are attractive to agent designers in general and in particular to robot builders who can see immediate applications for robot supply columns in hazardous or tedious environments.

A defining characteristic of 'ant-inspired' algorithms is the exploitation of stigmergy; "the production of certain behaviour in agents as a consequence of the effects produced in the local environment by previous behaviour"[20]). Ant-inspired solutions to various search problems have been demonstrated [6; 4; 2; 9; 5]. Chemical trail laying and following has been demonstrated in real robots [19; 18], as have chaining [23], foraging and object sorting behaviours [14]. However it is often impractical and sometimes undesirable for robots to physically mark their environment, so we suggest a method whereby a group of robots deposit landmarks in a shared *localization space*. We define localization space as any consistent spatial or topological representation of position. Such a space is *shared* if there is some (probably imperfect) correlation between the representations maintained by two or more individuals. A prime example is the Global Positioning System (GPS). Two systems equipped with GPS share a metric localization space in planetary coordinates. Similarly two robots that start out with known positions in the same coordinate system and maintain a position estimate via odometry share a localization space. In both examples each robot has only an *estimate* of its position in the true space, but the true space is common to both. More abstract localization spaces can be considered, such as the location of a data byte in a hierarchical database or a URL on the Internet. In these cases too, there can be some uncertainty in position; for example if position is described by a fuzzy matching rule or an ambiguous data query.

In this paper we demonstrate a cooperative transportation task in a group of simulated mobile robots that communicate by leaving landmarks in shared localization space. The method is shown to be robust with respect to significant localization error; indicating that it should be suitable for use in real robots.

2. TASK AND APPROACH

We examine the ant-like task of having multiple autonomous robots find and repeatedly traverse a path between a known 'home' and an initially unknown 'goal' position in an unknown environment.

Achieving this task reliably with robots would meet a real-world need. For example a factory may require a supply of widgets manufactured at position A to be transferred by robot to position B. In a Flexible Manufacturing System, the layout of the factory floor is expected to change over time. There will also be occasional robot breakdowns, perhaps blocking the supply route. There may be considerable benefit from a team of robots that can automatically find a new route without an up-to-date map, or exploit several routes in parallel. There is also a military need for supplies (medicine, food, ammunition) to be transported over hazardous and uncertain terrain. The start location and the existence of one or more goal locations may be the only known features, and these may be a few meters or several kilometers apart. Establishing a reliable automated supply column robust with respect to loss of individual robots could be very valuable.

The experiments described in this paper demonstrate a simple algorithm for resource transportation by robot teams using mechanisms loosely analogous to the ant trail following and the honey bee 'waggle dance' [22]. Instead of directly modifying their physical environment like ants, or performing an information-transmitting dance like bees, our robots communicate localization space landmarks over a wireless network. This communication requires very little bandwidth and is well within the capabilities of current networks. Our communication schemes is informed by the minimal strategies described in [10; 15] but are more complex because we are tackling a more realistic task in a more complex environment. Our eventual goal is to perform the task with real robots in a variety of environments.

3. MULTI-ROBOT SIMULATION

We have written a multi-robot simulator called 'Arena' which is used as a teaching and research tool at the University of Southern California (USC). It simulates the movement and sensors of many Pioneer-like robots in a two-dimensional rectangular arena (Pioneers are small, differential steering robots produced by ActiveMedia Inc.). Each robot can be provided with any or all of the sensors shown in Table 1. The navigation strategies presented here use only the odometry, sonar and region detectors.

The robots' sensors and wheel motors are modeled with low fidelity to achieve a very high update frequency (≈200Hz with 20 robots on a modest 400MHz Pentium II). We believe this is justified by the presence of noise and wide variation in characteristics of real transducers. For these experiments we do not impose artificial sensor noise on top of the low accuracy values reported by the simulator, except in the position estimates as described below. Collisions are treated simply; if the robots bump into something they are stuck unless they turn away from the obstacle.

Arena is implemented as a TCP/IP network server. Robot controllers are independent client processes communicating with Arena via a socket. Arena sends a message to each connected controller at 10Hz indicating the corresponding robot's current speed, turn rate, position estimate and sensor readings. Controllers asynchronously send messages back to Arena indicating the latest speed and turn rate demands for their robot.

Environments are specified as occupancy grids, imported as black and white bitmaps. We can draw simple environments with boulder-like obstacles, as used in the experiments in this paper, or more interestingly, we can import CAD models of real buildings to provide rich, realistic environments.

4. ALGORITHM

Robots start from a home position and must search for the goal. On reaching the goal, they receive a unit of resource and must return home with it, then return to fetch more resource repeatedly for the length of the trial. Each robot records its movements as a sequence of landmarks in localization space, each with a position, heading and timestamp. After each successful traverse, the robot shares the complete path with its team mates. The algorithm consists of three decision processes running in parallel. A schematic is shown in Figure 1.

The trail laying algorithm is independent of the method used to drive the robot's wheels; rather it can provide a hint of a 'good' heading to go in from the robot's current location. In this implementation the robots initially search the environment by a random walk. This was chosen because it is simple, requires no *a priori* knowledge of the environment and will always find a path if one exists, given sufficient time. As it moves through the environment each robot records its current position and heading estimate at regular time intervals. We refer to this as 'crumb dropping'. As time goes by, a robot stores many such records on its initially empty *private crumb list*.

If a path to the goal exists, a robot will eventually reach the goal. Whenever a robot reaches the goal it announces the contents of its private crumb list on the broadcast channel of the shared network. All the robots (including the sender) receive the crumbs and add them to their initially empty *global crumb list*. The sender's private crumb list is cleared. Communicating the route only upon completing a successful trip corresponds loosely to the honey bee's waggle dance, as compared to the ant which leaves a trail as it goes along.

The first robot to achieve the goal must have used the most time efficient path yet discovered and all robots now have a list of waypoints describing this path. At each timestep each robot examines its global crumb list and computes the average heading of all those crumbs that lie within a threshold radius of its current position estimate. If there are no such crumbs, the robot performs random walk, otherwise it attempts to follow their average heading. If a robot is carrying resource, i.e. it has reached the goal, it will follow the opposite heading back to the home position. On reaching home a robot also broadcasts and clears its private list. A crumb stays on the global crumb list for a short time (we used 20 seconds) after which time it is destroyed, corresponding to the evaporation of ant pheremone trails.

Thus a robot whose position estimate is similar to the position of a dropped crumb will tend to move in the same direction as the successful robot that passed that way earlier. By following a trail of crumbs each robot can find the goal much faster than by random walk.

5. EXPERIMENT 1: EFFECTIVENESS

This experiment was performed to evaluate the performance of the trail-following algorithm described above, by comparing it with a random exploration strategy and a conventional gradient descent approach.

5.1 Robot controllers

We devised and tested three robot controllers; one (**'Random'**) which has no heading bias and explores the arena at random; a second (**'Directed'**) which performs gradient descent towards the goal; and a third (**'Ant'**) implementing the crumb trail following algorithm described above; Controllers were designed in the behavior-based paradigm [3], and composed of the following basis behaviors [13]:

1. *RandomWalk:* If there is an obstacle nearby, turn to a random heading. Otherwise move forward.

Sensor	#	Field of view	Range	Freq.	Emulated device
location (x, y)	1	-	-	10Hz	Pioneer odometry or GPS
heading θ	1		$\pm 180^o$	10Hz	Magnetic compass
forward speed v	1	-	$\pm 256 ms^{-1}$	10Hz	Pioneer wheel encoders
rotation speed ω	1	-	$\pm 256 rad.s^{-1}$	10Hz	Pioneer wheel encoders
low-res range finder	7	-90,-30,-15,0,+15,+30,+90^o	$0.5 \rightarrow 5$m	10Hz	Pioneer sonar turret
high-res range finder	361	-90 to +90^o	$0 \rightarrow 8$m	5Hz	SICK LMS200 laser scanner
region detector	8	binary in/out of rect. region	-	5Hz	various

Table 1: Sensors provided by the Arena simulator.

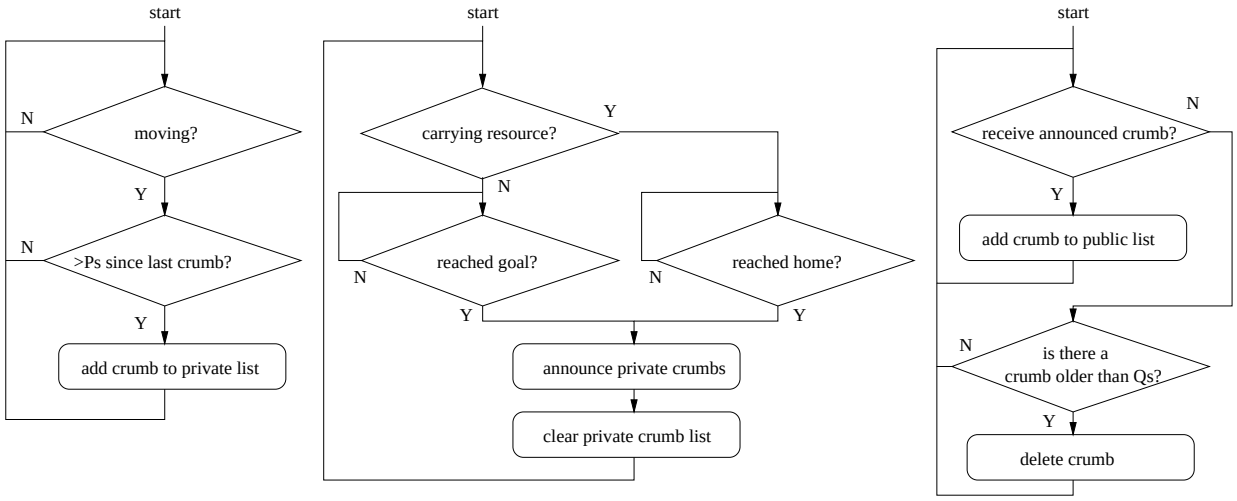

Figure 1: Schematic of the trail laying algorithm. Three decision processes run in parallel to manipulate the private and public crumb lists.

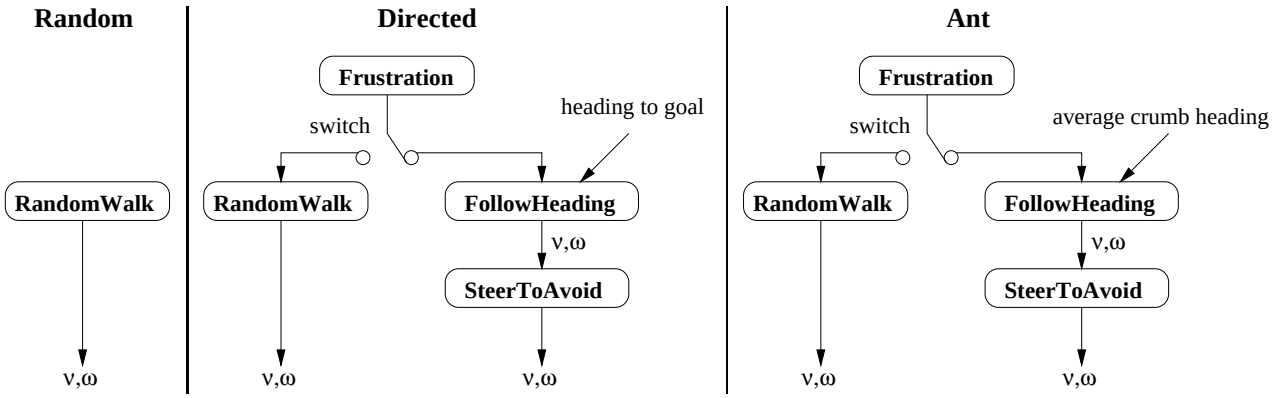

Figure 2: Schematics of the three robot controllers.

2. *FollowHeading:* The robot is supplied with a desired heading. Turn towards the desired heading until the heading error is small, then move straight forward.

3. *SteerToAvoid:* If there is an obstacle to the left, steer right; if there is an obstacle to the right, steer left. If there is an obstacle close in front, stop.

4. *Frustration:* monitor the robot's speed. If the speed has been very low for a short time, perform *RandomWalk* for 3 seconds.

The composition of each complete controller is shown in schematic in Figure 2.

1. **Random:** *RandomWalk* alone is sufficient to explore the environment and avoid crashes with walls and other robots. This stochastic behavior will explore the whole environment (except very narrow spaces) given sufficient time.

2. **Directed:** *FollowHeading* and *SteerToAvoid* work together to drive the robot towards a desired heading, steering past most obstacles. However, the steering is crude, and it is common for the robot to stop facing an obstacle or another robot. *Frustration* detects this condition and allows the robot to become unstuck using *RandomWalk*.

   The goal heading is obtained by calculating the heading to the center of the goal zone (or the home zone if the robot is on its way home). This simulates a sensor which can detect a goal landmark such as a visible target or radio beacon. The directed strategy can be expected to find its way to the goal, but like all such methods it will be impeded by any dead-ends it encounters. However, *Frustration* latches *RandomWalk* for long enough to allow the robot to escape any local minima in the environments studied here. **Directed** does not require a localization estimate.

3. **Ant:** identical to **Directed** except that the desired heading input to *FollowHeading* is obtained by examining the global crumb list as described above. Note that **Ant** does not need to know the direction of the goal, but does need a localization estimate.

5.2 Experimental design

5.2.1 Environments

We experimented in two square environments of side corresponding to 33.33m. Robots are 0.6m square. The environments are shown in Figure 3.

The first environment (Figure 3 left) consists of a number of large boulder-like obstacles of various sizes between a 'home' square and a 'goal' square, each of side 5m. This environment is designed such that there are no significant dead-ends between home and goal; gradient descent towards the goal will be successful unless the path is blocked by another robot. We refer to it as the 'easy' environment.

The second environment (Figure 3 right) is similar to the first, except that the narrow gap between two boulders on the right hand side is now closed. This reduces the number of possible paths between home and goal, and also creates a significant local minimum for our gradient descent algorithm. Approximately 50% of possible routes from home to goal will encounter this obstacle. We refer to this as the (relatively) 'hard' environment.

5.2.2 Procedure and data gathered

At the beginning of each experiment a number of robots are placed at random orientations in the home zone in the bottom left corner of the environment. The controllers are started and the robots start to move around in the environment. After some time one of the robots reaches the resource area in the upper right corner of the environment. The time it takes to get the first robot to the goal area is recorded and after that the simulation is allowed to run for another two minutes. During the experiment the number of round trips from home to goal zone and back is recorded and used as a measurement of the experiment's success.

To evaluate the effect of group size on performance, the experiment was performed with each of the three controllers for group sizes of 5, 10, 15 and 20 robots. Twelve trials were performed for each combination of controller and group size to allow statistical evaluation of the results.

5.3 Results

Figure 3 shows screenshots of typical trials. The light dashes on the black background indicate crumbs on the global crumb list. Trails of crumbs can be observed running between the home and goal zones. In the easy environment (left) a trail can be seen passing through the narrow passage on the right hand side of the picture. In the hard environment (right), this passage is blocked and a dense trail can be seen passing by the top left corner of the obstacle. Due to lack of space we will not discuss the patterns of behavior that can be observed in the simulation; we focus here on statistical observations.

5.3.1 Performance

Figure 4 shows the mean and standard deviation of the number of units of resource transported in a two minute period for each of the controllers and group sizes. As could be expected, the **Random** controller performed badly in both environments. In the easy environment (left) the **Directed** controller performed better than the **Ant** controller. This is to be expected as a gradient descent in a environment with no dead-ends is well known to produce good paths (but see [12] for discussion of the general limitations of such methods). The ant method does slightly worse, achieving approximately 85% of **Directed**'s score.

In all cases the success increases with the group size, but the increase appears to slow down with the larger robot groups. This is due to the trade-off between enhanced exploration efficiency and interference effects, where robots get in each other's way. For any fixed environment there is a 'critical mass' of robots that achieves maximum success [1; 7].

In the hard environment (right) we observe that the success of all strategies is reduced, but the success of **Ant** is now greater than that of **Directed**, indicating that it is less effected by the changed environment.

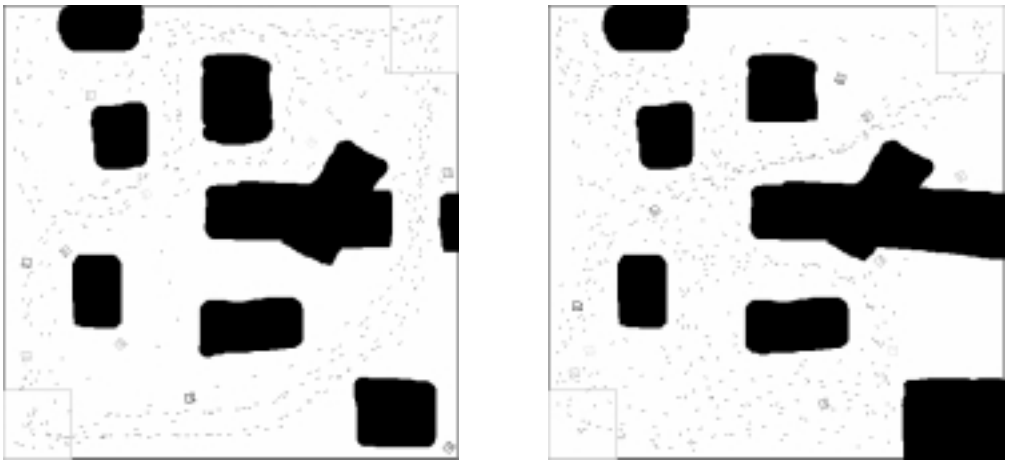

Figure 3: Screenshots of the trail laying robots at work in the two environments used in the experiments; 'easy' (left) with no dead-ends between home and goal and 'hard' (right) with a significant dead-end area. The 'home' zone is the square at the bottom left; 'goal' zone is the square at the top right. Light dashes correspond to dropped crumbs.

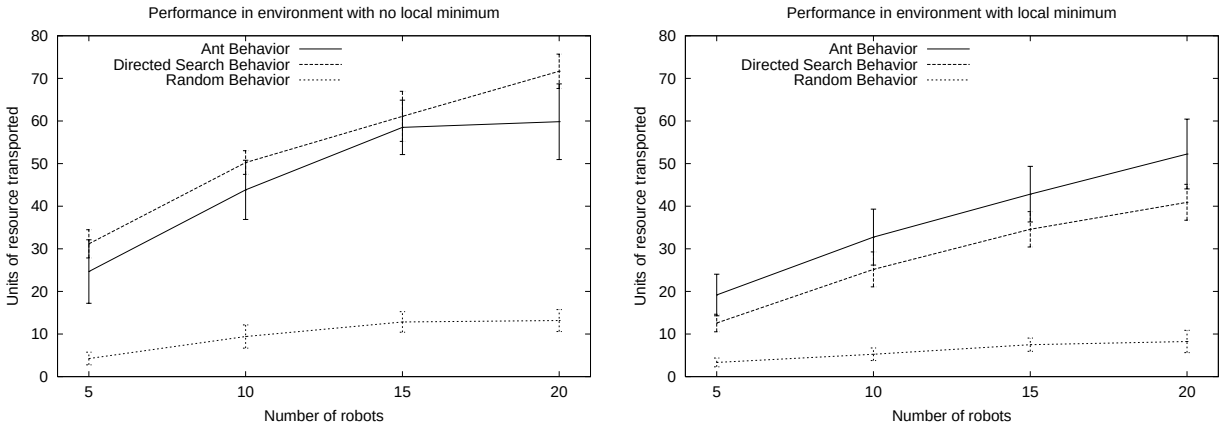

Figure 4: Units of resource transported plotted against group size for the three robot controllers in the easy (left) and hard (right) environments.

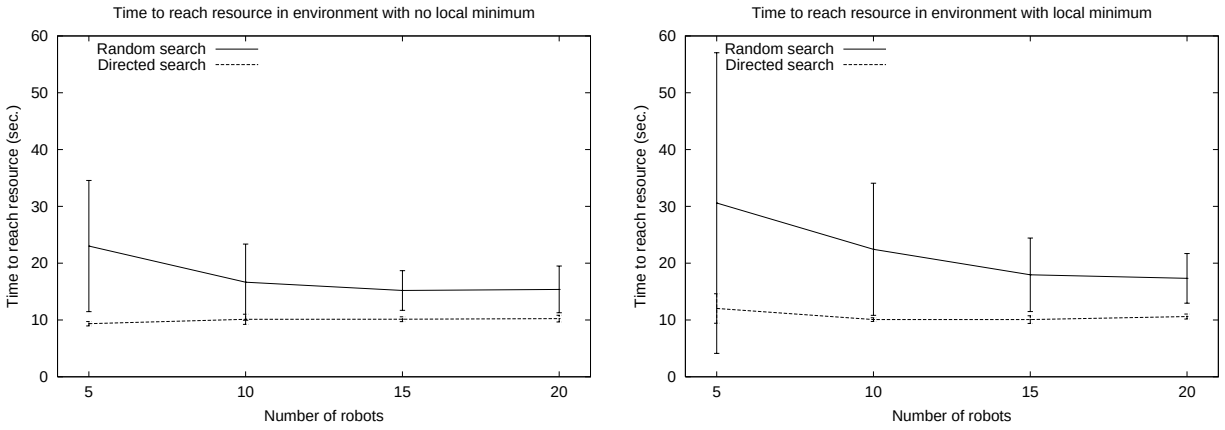

Figure 5: Time taken for the first robot to reach the goal position plotted against number of robots in the easy (left) and hard (right) environments.

f-test	Ant, Dir	Ran, Dir	Ant, Ran
5	0.024	0.025	<0.001
10	0.010	0.095	0.009
15	0.420	0.013	0.006
20	0.029	0.308	<0.001

Table 2: Probabilities of equal variance of data from controller pairs in the easy environment.

t-test	Ant, Dir	Ran, Dir	Ant, Ran
5	0.018	<0.001	<0.001
10	0.013	<0.001	<0.001
15	0.334	<0.001	<0.001
20	0.001	<0.001	<0.001

Table 3: Probabilities of equal mean of data from controller pairs in the easy environment.

f-test	Ant, Dir	Ran, Dir	Ant, Ran
5	0.016	0.059	<0.001
10	0.268	0.004	<0.001
15	0.297	0.006	<0.001
20	0.074	0.130	0.001

Table 4: Probabilities of equal variance of data from controller pairs in the hard environment.

t-test	Ant, Dir	Ran, Dir	Ant, Ran
5	0.001	<0.001	<0.001
10	0.004	<0.001	<0.001
15	0.002	<0.001	<0.001
20	<0.001	<0.001	<0.001

Table 5: Probabilities of equal mean of data from controller pairs in the hard environment.

5.3.2 Start-up time

Figure 5 shows the amount of time taken for the first robot to reach the goal for **Ant** and **Directed** controllers.

In the easy environment (left) **Directed** search reaches the goal in around 10s in each group size. In contrast **Ant** takes longer, with the mean and the standard deviation both reducing rapidly with increasing group size.

The result for the hard environment (right) has similar characteristics, but for small group sizes the mean time to goal is longer, as could be expected. For larger group sizes, the mean time to goal is similar in both environments.

5.4 Statistical Analysis

To determine which algorithm performs best in each environment, we must establish that there is a significant difference in the distributions of their success scores.

First, we use a two tailed F-test to see if the variances of the observations are significantly different. Tables 2 and 4 show the probabilities for each pair of controllers that the variance of their distributions is the same.

The conclusions of the first F-test enable us to choose the appropriate T-test for significance difference of means. Tables 3 and 5 show the probability for each pair of controllers that their observed means belong to the same distribution. We can conclude that the samples come from different distributions for each pair except **Ant** and **Directed** in the easy environment with fifteen robots. We conclude that **Directed** is found to perform better than **Ant** in the easy environment, but the opposite is true in the hard environment, confirming the qualitative analysis.

6. EXPERIMENT 2: ROBUSTNESS

The presence of sensor noise means that localization is usually inaccurate. The error in a position estimate can be large but approximately constant as with GPS, or start off small but accumulate significantly as with odometric methods. The affordability and accuracy of GPS is improving rapidly. Several authors have described methods for reducing the growth rate of odometric error and for keeping errors within reasonable bounds [21; 17].

In this section we demonstrate that the **Ant** controller is robust with respect to significant error in its localization estimate. The controller is identical to that used in the previous experiment and we emphasize that there is no explicit mechanism to model or compensate for localization error.

6.1 Experimental design

In order to test the robustness of the **Ant** controller, we measured its success at the transportation task for increasing levels of localization error. The error was modeled by adding Gaussian noise to the correct robot position generated by Arena before being passed to the controller program. Twelve trials were performed for each combination of 5, 10, 15 and 20 robots, with noise variance of 0, 1, 2, 4 and 8m.

A success score is measured for each trial, along with the proportion of time the controllers spend in *RandomWalk*, indicating time spent *not* doing useful transportation.

6.2 Results

Figure 6 shows a plot of the mean success of 12 trials for each group-size/noise pair, in the easy (left) and hard (right) environments. A decrease in the success score can be seen as the localization error increases, apparently independent of the group size. In both environments a localization error of variance 8m results in a success rate around 80% of that when perfect localization is available. As there are several choices of real-world localization scheme that can exceed this accuracy, we can be confident that this method will be reasonably successful when implemented on our real robots.

Figure 7 shows the proportion of time spent in *RandomWalk*. It is seen to increase with the number of robots, the localization error and the difficulty of the environment. The behavior is activated when the robot is unable to steer around an obstacle, so its usage gives an (inverse) measure of how smoothly the robot could navigate its environment [8].

Two factors can influence the smoothness; interference from other robots and the correctness of the crumb trail. The increase in *RandomWalk* with group size is expected and is due to interference. The increase with localization noise is due to the widening of the trails as the crumbs become more scattered in localization space. This makes it more difficult

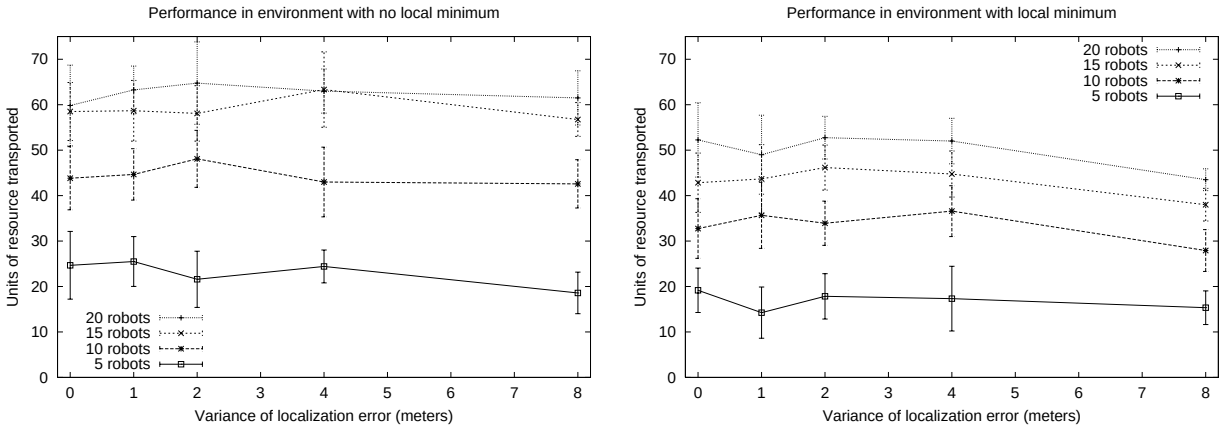

Figure 6: Units of resource transported against localization error and group size in each of the environments.

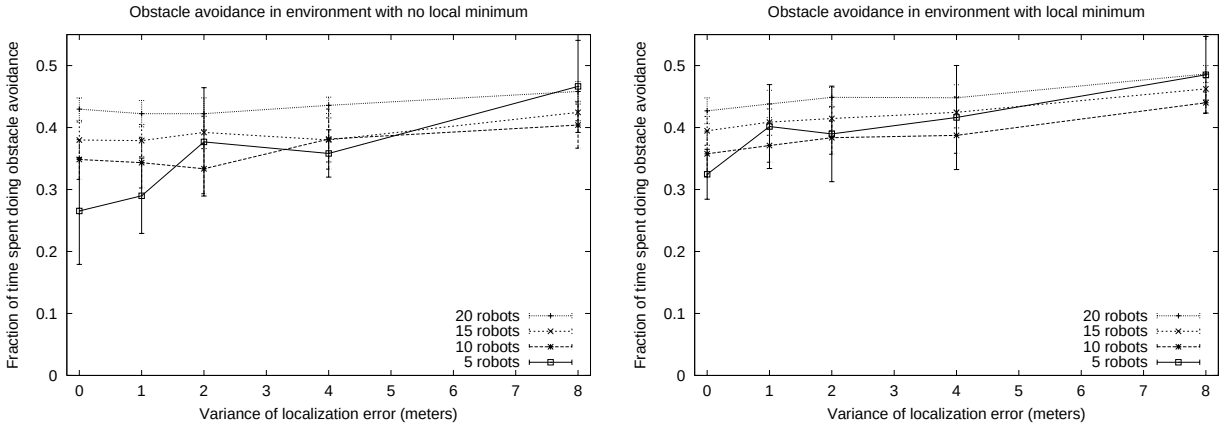

Figure 7: Time spent avoiding obstacles against localization error and number of robots in each of the environments.

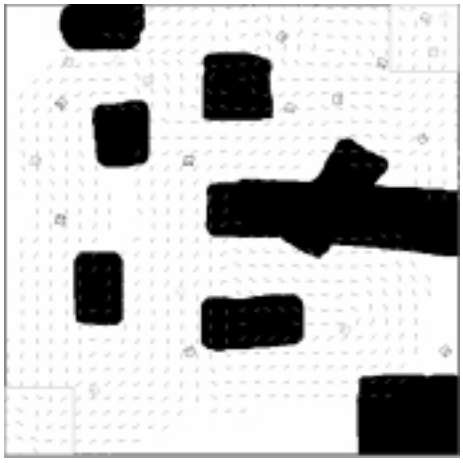

Figure 8: Example direction fields produced by the trail-laying algorithm in each environment.

for robots to follow trails around obstacles and through narrow spaces. We speculate that if localization noise is random and unbiased, the quality of trails will increase over time as larger numbers of crumbs are averaged. The data are not yet available to examine this.

6.3 Interpreting trails as maps

Over time the effect of the trail laying algorithm is to annotate (in localization space) the whole of the explored environment. The longer the lifetime of the crumbs on the public list, the better the coverage and the less variable the average heading, at the cost of some more memory and processing time. Wherever a robot finds itself it can obtain a hint of which way to go to achieve the goal. Those parts of the environment that are most often traversed will contain the most crumbs, and these will give an average heading most likely to point along a route towards the goal. If the local average crumb heading is obtained at regularly-spaced positions and the results plotted, we obtain a map of the **Ant**'s navigation environment (Figure 8). Notice that close to obstacles the recommended heading tends to be parallel to the obstacle surface; further away the heading tends to point between obstacles. We intend to examine the properties of these maps and to compare them with those produced by more formal path planning methods.

7. CONCLUSIONS AND FURTHER WORK

We have demonstrated that trail-laying behaviors can be usefully implemented in a shared localization space, rather than directly in the real world, despite the presence of significant localization error. A simple algorithm was described that robustly performs a transportation task between a known start and initially unknown goal locations, using information readily available to real robots.

We aim to expand on several aspects of this work:

- Demonstrate the utility of this method for real robots;
- Characterize the maps produced and compare them to the results of other techniques;
- Examine the possibilities of this approach in more abstract localization spaces;
- Increase competence in the transportation task.

Acknowledgements

Thanks to Paolo Pirjanian for useful discussions. This work is supported by DARPA contract DAAE07-98-C-L028 and grant DABT63-99-1-0015.

8. REFERENCES

[1] R. C. Arkin and J. D. Hobbs. Dimensions of communication and social organization in multi–agent robotic systems. In *Proc. 2nd Int. Conf. Simulation of Adaptive Behaviour*, pages 486–493, 1992.

[2] R. Beckers, O. E. Holland, and J. L. Deneubourg. From local actions to global tasks: Stigmergy and collective robotics. In *Proc. Artificial Life IV*. MIT Press, 1994.

[3] R. A. Brooks. A robust layered control system for a mobile robot. In *IEEE Journal of Robotics and Automation*, volume RA-2 (1), pages 14–23, Apr. 1986.

[4] J. L. Deneubourg, S. Aron, S. Goss, J. Pasteels, and G. Duerinck. The dynamics of collective sorting: Robot-like ants and ant-like robots. In *Proc. 1st Int. Conf. Simulation of Adaptive Behaviour*, pages 356–363, 1990.

[5] J. L. Deneubourg, G. Theraulaz, and R. Beckers. Swarm-made architectures. In F. Varela and P. Bourgine, editors, *Proc. European Conference on Artificial Life*, pages 123–133. MIT Press, 1992.

[6] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, pages 26(1):29–41, 1996.

[7] M. S. Fontán and M. J. Matarić. Territorial multi-robot task division. *IEEE Transactions on Robotics and Automation*, 14(5), 1998.

[8] D. Goldberg and M. J. Matarić. Interference as a tool for designing and evaluating multi-robot controllers. In *Proceedings, AAAI-97*, pages 637–642, Providence, RI, 1997. AAAI Press.

[9] S. Goss, J. L. Deneubourg, R. Beckers, and J. Henrotte. Recipes for collective movement. In *Proc. European Conference on Artificial Life*, pages 400–410, 1993.

[10] O. Holland and C. Melhuish. Chorusing and controlled clustering for minimal mobile agents. In *Proc. European Conference on Artificial Life*, Brighton, UK, 1997.

[11] B. Holldopler and E. Wilson. *The Ants*. Springer-Verlag, 1990.

[12] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proc. 1991 IEEE International Conference on Robotics and Automation*, 1991.

[13] M. J. Matarić. Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4(1):50–81, December 1995.

[14] C. Melhuish, O. Holland, and S. Hoddell. Collective sorting and segregation in robots with minimal sensing. In *Proc. 5th Int. Conf. Simulation of Adaptive Behaviour*, 1998.

[15] C. Melhuish, O. Holland, and S. Hoddell. Using chorusing for the formation of travelling groups of minimal agents. In *Proc. 5th Int. Conf. Intelligent Autonomous Systems*, 1998.

[16] O. Richards. *The Social Insects*, page 113. Harper Torchbook, 1970.

[17] S. Roumeliotis, G. Sukhatme, and G. Bekey. Smoother based 3-d attitude estimation for mobile robot localization. In *In Proc. 1999 IEEE International Conference on Robotics and Automation*, 1999.

[18] R. Russell. Ant trails - an example for robots to follow? In *Proc. 1999 IEEE International Conference on Robotics and Automation*, pages 2698–2703, 1999.

[19] T. Sharpe and B. Webb. Simulated and situated models of chemical trail following in ants. In *Proc. 5th Int. Conf. Simulation of Adaptive Behavior*, pages 195–204, 1998.

[20] G. Theralauz, E. Bonaneau, and J. L. Deneubourg. The origin of nest complexity in social insects. *Complexity*, 3(6), 1998.

[21] S. Thrun. Learning maps for indoor mobile robopt navigation. *Artificial Intelligence*, 1:21 to 71, 1999.

[22] K. von Frisch. *Bees. Their Vision, Chemical Senses, and Language*. Cornell University Press, Ithaca N.Y., 1950.

[23] B. B. Werger and M. J. Matarić. Robotic "food" chains: Externalization of state and program for minimal-agent foraging. In *Proc. 4th Int. Conf. Simulation of Adaptive Behavior*, pages 625–634, 1996.