Try To See It My Way: Using Task-level Features to Transform Between Reference Frames in Robot Teams

Richard T. Vaughan* Kasper Støy† Gaurav S. Sukhatme Maja J. Matarić

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

Abstract

We describe a team of robots that uses a trail of landmarks to navigate between places of interest. The landmarks are not physical; they are waypoint coordinates generated on-line by each robot and shared with team-mates over the network. Waypoints are specified with reference to features in the world that are relevant to the team's task and common to all robots. Using such task-level features as landmarks avoids the need to sense and name physical landmarks. Using these common landmarks, each robot can transform waypoint coordinates into its local reference frame, avoiding the cost of maintaining a fixed global coordinate system.

The algorithm is tested in an experiment in which a team of 4 autonomous mobile robots run in our office building for more than 3 hours, travelling a total of 8.2km (5.1 miles). Despite significant divergence of their local coordinate systems, they are able to share waypoints, forming and following a common trail between two fixed locations.

1 Introduction

We are interested in cooperative strategies for teams of autonomous robots as a means to improve overall performance. Cooperation is a means whereby a whole system can be 'more than the sum of its parts'.

In this paper we describe a method for generating and using trails of landmarks to navigate between places of interest. The landmarks are not physical; they are waypoint coordinates generated on-line by each robot and shared with team-mates over the network. Waypoints are specified with reference to features in the world that are relevant to the team's task and common to all robots. Using such task-level features as landmarks avoids the need to sense and name physical landmarks. Using these common landmarks, each robot can transform waypoint coordinates into its local reference frame, avoiding the cost of maintaining a fixed global coordinate system.

*corresponding author, email vaughan@usc.edu.
†Now at the University of Southern Denmark, Odense, Denmark.

The method is applied to an example 'resource transportation' task, in which multiple autonomous robots find and repeatedly traverse a path in an unknown environment between a known 'home' and a supply of resource at an initially unknown position.

1.1 Previous work, limitations

An earlier version of our trail following strategy was described in [13] and developed in [12]. It was inspired loosely by the foraging mechanisms of ants and bees [6, 14]. The effectiveness of ant foraging has inspired solutions to a variety of optimization problems expressible as path-finding [4, 3]. Chemical trail laying and following has been demonstrated in robots [9, 8], as have other applications of stigmergic communication [7]. Further useful references can be found in [2] and our previous papers.

Our trail following algorithm has some attractive properties: it finds good paths between locations, is reliable, is independent of the localization method used, is adaptive to dynamic environments, and requires modest computation and communication resources. However, because it is impractical for our robots to leave physical trails, our trails are purely informational; they are lists of waypoints in *localization space*. This is an important drawback compared to the physical trail marking of ants as it requires that a position estimate is maintained by each robot. Maintaining an accurate position estimate in a global coordinate frame is often complex and always costly, with the cost rising with the accuracy required.

The most readily available sensors for localization are rate-measuring devices; odometry, accelerometers, gyroscopes, which have no fixed frame of reference. The rates must be integrated over time to obtain position estimates. The magnetic compass and Global Positioning System (GPS) are counter-examples with fixed reference frames, but these are restricted to certain (magnetically neutral, satellite line-of-sight) environments, so we do not consider them here. Rate-integrating methods suffer unbounded drift and are therefore nonstationary; this means that a population of robots which start from known global positions and update their position estimates by integration will have their coordinate systems diverge over time.

Some methods for sharing information between agents are more tolerant to misaligned frames than others; our trail following scheme was designed to be robust to significant error, both in terms of variance [13] and in mean [12]. However, once two agents' localization spaces have diverged past a certain point, they can no longer share position information. In an experiment with real robots localized by odometry, the system broke down after 28 minutes as the coordinate systems diverged drastically [ibid.].

The obvious solution to this problem is for all agents in a team to share a common global coordinate system. Unfortunately, global localization is a general complex problem. Some of the most successful approaches [5, 11] are computationally intensive and many use a map of the environment, either supplied as a prerequisite, or generated as a by-product or an explicit system goal [10].

Our thesis is that many tasks, including cooperative resource transportation can be achieved without computing full, map-like representations of the environment, thus saving the time and power required for the computation.

For reasons of generality, simplicity, efficiency and robustness we aim to design robot systems that can work in these local, nonstationary coordinate systems in novel environments. In multi-robot systems with symbolic communication this becomes a problem of referring to places in the world without maintaining a common coordinate system.

1.2 Contributions

This paper describes an extended version of our basic trail following method which overcomes some earlier limitations, subject to certain constraints:

1. By referring to task-level features common to all robots, waypoints can now be shared between unrelated coordinate systems. This removes the earlier requirement for a common frame of reference within the localization system.

2. The method is generalized to N source/sink locations by means of a more sophisticated representation.

3. The probability that robots will take the shortest path is increased by labeling waypoints with an estimate of time-to-goal, instead of the suggested heading used in our previous work.

With these additions and a small modification to our controller, our real robot implementation becomes highly robust. The extended algorithm is described below (Section 2). The experiment of [12] is repeated; this time the algorithm is shown to work indefinitely in a team of four robots (Section 3).

2 Task and Approach

A team of robots works to transport resource in an unknown environment. Robots start from a home position and search for a source 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. Each robot records its movements as a sequence of waypoints in localization space and shares this path with its team mates. Thus each robot in the team has access to information about parts of the environment it has never visited, improving the performance of the system compared to non-communicating robots.

The trail laying algorithm guides the robot between home and goal. It is independent of the method used to drive the robot's wheels; rather it can provide a hint of a 'good' heading to pursue from the robot's current location. The details of efficiently navigating through the environment and avoiding obstacles are left to a low-level controller. We view the independence of high level navigation from the detailed control of the robot as a significant advantage of this method.

The rest of this section describes the trail following mechanism; the details of robots and low-level controller are briefly described in the experimental section.

2.1 Trails, Places and Crumbs

Our previous work assumed that there were two places of interest in the environment (referred to as 'Home' and 'Goal') connected by a single crumb trail. The trail was followed forwards towards the 'goal' and backwards towards 'home'. We have since generalized the method to handle N places by recording N trails (though we will only present results here for an N=2 experiment).

The enhanced trail representation consists of *Crumbs* (the waypoints) and *Places* (the task-level common landmarks) collected into a *Trail*. Information is shared by combining Trails according to certain rules. Each of these elements is described below.

2.1.1 *Places:* global task-level landmarks

The main aim of this work is to remove the requirement that a group of robots maintain a conventional global coordinate system or map in order to communicate information about places of mutual interest. The method is based on the following observation:

A group of robots engaged in a common task will share some events or experiences in common, independent of their localization space. For example, in our transportation task illustrated by Figure 1, the source A and sink B of resources are common to all robots. At the source the robot finds a unit of resource; at the sink it its relieved of the unit. The coordinate at which this happens is different for each robot as each has an independent frame of

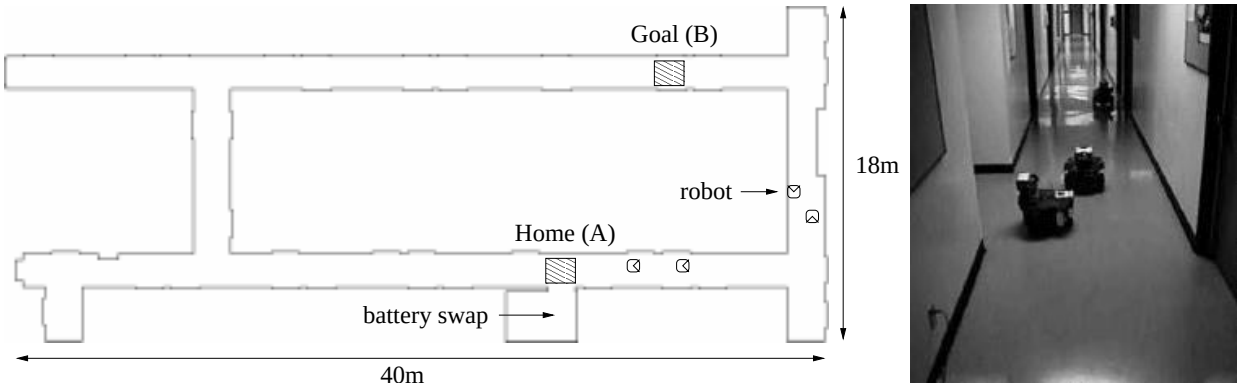

Figure 1: Diagram of the experimental environment; part of our office building; robots are drawn approximately to scale (left). Photograph of the environment (right).

reference; it even changes over time as the local coordinate frame drifts. References to another place C can be made relative to the current estimates of A and B. Any member of the population that maintains estimates of A and B can interpret a reference to C in local coordinates in terms of A and B. The accuracy of the transformation is determined by the accuracy of estimates A and B held by both parties.

If we assume (i) that the local coordinate system drift is small over the interval between occurrences of any event; and (ii) that the system can tolerate some position error when using shared coordinates, then robots can share information even when their natural coordinate systems are otherwise unrelated. The most simple way to maintain estimates of positions A and B is to maintain a current position estimate, and when an event (such as finding resource) occurs, to set the event position estimate equal to the current position. As the goal of the transportation task is to shuttle between A and B, their position estimates are updated frequently.

This is essentially a common landmark method; unique environmental features can be distinguished and the population agrees on a naming convention (A, B) for each feature. The advantage of using task-related events (finding/dropping resource) instead of conventional landmarks (the Empire State Building, the North Pole) is that *no overhead is required to detect the landmark* and *the agents already agree on what the landmarks will be.* The ability to detect the states **have-resource** and **do-not-have-resource** is already required by the system; the robot must behave differently depending on the state. Thus any robot that is capable of doing the task has sufficient information to share coordinate systems with any other. The user does not need to implement a separate sensing/processing landmark-recognizing-and-naming subsystem on each robot. Indeed, we assume nothing about the sensing and internal architecture of any member of the population. All that is required is that each robot maintain estimates of the positions of common task-related events.

Places are represented as tuples containing the name P of the event (e.g. A,B), and the location estimate (x,y).

```
Place -> [P x y]
```

2.1.2 *Crumbs:* local waypoints

Named for the trails of Hansel and Gretel, our trail waypoints are called Crumbs. Our Crumbs are data structures that describe the distance (in time) to a Place from a particular position in the local coordinate system.

A Crumb is composed of the name of the Place P to which it refers, a position (x, y), an estimate s of the time required to get to P, and the time t when the Crumb was created:

```
Crumb -> [P x y s t]
```

2.1.3 *Trails:* shared local data

A Trail is a set of Crumbs and Places. Each robot starts with an empty trail which is built up over time according to the trail-laying algorithm. A very simple trail might look like this:

```
[A 0 0]          #Place
[B 10 5]         #Place
[A 3 4 20 201] #Crumb
```

This could be expressed in English as: in the coordinate system in which Place A is at (0,0) and Place B is at (10,5), A was 20 seconds away from (3,4) at t=201 seconds.

2.1.4 Adding Places to a Trail

When a robot detects a task level event, it checks its Trail to see if a Place already exists with the name of that event. If not, a new Place is created. If a Place already exists for that event, its position estimate is updated to match the robot's current position. Only one Place can exist for each

type of event. For example in the resource transportation experiments, there are two Places, one for receiving resource (A) and one for dropping resource (B).

2.1.5 Combining trails

Two Trails can be combined if they have two Places in common. If Trail α contains Places A_α and B_α and Trail β contains places A_β and B_β, there is a unique transformation T that maps the vector from A_α to B_α onto the vector from A_β to B_β:

$$T(A_\alpha\vec{B}_\alpha) = A_\beta\vec{B}_\beta$$

where T is a simple linear coordinate transform consisting of a translation, scaling and rotation.

Once T is determined using the Places common to both Trails, the same transformation can be applied to map the coordinates of any point P (specified as a vector from the origin to P) from one Trail to the other. To combine the trails each Crumb is copied from one Trail to another, with its position coordinate transformed through T. Similarly, any Places that are not already common to both Trails can also be copied from one to the other, with their positions transformed through T. Any Places in common other than the ones used to determine T are not copied; they are assumed to be consistent.

If two trails do not have two Places in common, two different actions can be taken depending on an adopted policy. If we adopt a *pessimistic* policy, the coordinate systems are assumed to be different and the trails can not be combined. If we adopt an *optmistic* policy the coordinate systems are assumed to be the same and the trails are combined as-is, by simply copying Crumbs and Places from one Trail to the other. If the Trails have one Place in common and the optimistic policy holds, orientation and scale are assumed to be similar to both Trails, and the Crumbs and Places can be transformed by translation alone into the local coordinate system.

2.2 Trail-laying algorithm

1. If a robot experiences an Event, it searches its Trail for a Place with the name of that Event. If it exists, the Place has its coordinate set to the current position. If it does not exist, a new Place is created with the name of the new Event and the current position and inserted into the Trail.

2. If a robot is moving, it creates a temporary Trail every S seconds (we used $S = 2$). The temporary trail consists of copies of all the robot's Places, plus a single Crumb filled in with the current location, the name of the last Place visited, the time elapsed since visiting the Place, and the current time. [Alternatively, a Crumb can be generated for each Place in the Trail rather than just the most recent Place visited; we will examine this option in future work]. This temporary trail is broadcast on the network, then deleted.

3. If a robot receives a trail on the network (including trails it sent itself) the received trail is combined with the local Trail according to the rules in Section 2.1.5 above. In this way Crumbs and Places generated by individual robots are incorporated into the Trails of all robots on the network.

   In the system described in the experiment below, the received trail is transformed into the local coordinate system. Another possibility is to transform the contents of the local Trail to match the received trail; this will cause all the Trails to converge to a common coordinate system. However, as the incoming trails are very small, and the local trails are relatively large, this is computationally expensive. As we do not seek a global representation, we choose to do the least expensive transformation.

	Tanis	Bug	Fly	Ant	Total
Trips	36	59	78	86	262
Distance (m)	1102	2102	2414	2592	8211
Lifetime (min)	81.3	138.1	176.6	188.4	583.6

Table 1: Summary of results of robot trial

2.3 Trail-using algorithm

1. At each timestep each robot examines its Trail and finds those Crumbs which reference its current goal (Place A or B) and have a position within a fixed radius of the current position. The preferred direction of movement is towards the Crumb with the lowest time-to-Place estimate. 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 most successful robot to pass that way earlier. By following a trail of crumbs each robot can find the resource much faster than by random exploration.

2. Any Crumbs with timestamp older than some threshold (we used 4 minutes) are destroyed. This allows for dynamic update of the trail, as out-of-date information is deleted. The dynamic response of the trail to changing environments is a function of this age threshold.

3 Experiment

The following experiment was performed to test the effectiveness of the trail algorithm in a simple version of the resource transportation task.

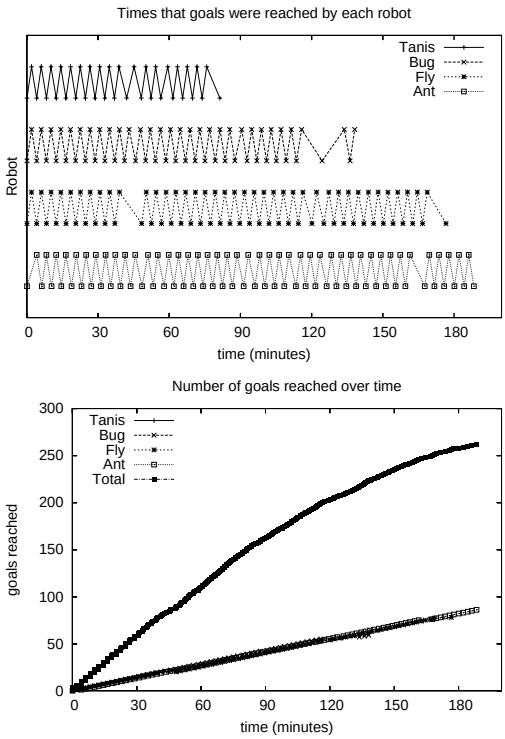

Figure 2: Reaching Places: the time at which each robot reached home and goal (top), and the total number of trips over time (bottom).

3.1 Robots

The platform for this experiment was four identical ActivMedia Pioneer 2DX robots (named Bug, Ant, Fly, and Tanis). These are fairly small (50x50x40cm), differential-steering mobile robot bases, fitted with PC104+ Pentium II computers running Linux. Each robot has front and rear sonar rings and wheel encoders as its basic sensors. For these experiments a SICK LMS scanning laser rangefinder was fitted to the front of each robot, providing good quality range readings over a 180^o field of view. The SICK gives two samples per degree and a range accuracy of a few millimeters to most surfaces. 802.11 wireless Ethernet connected the robots and our workstations with an effective bandwidth of 1Mbit/sec.

3.2 Environment

The system was tested in our unmodified office building, with the resource separated from the home position by two corners and 31m of corridor. To follow the optimal route to the resource, the robots must make two correct turning decisions based on the hint given by the crumb trail. The layout is shown in Figure 1(left) and is the same as used in our previous work, so the results can be directly compared. Figure 1(right) shows the robots navigating in the test environment.

3.3 Robot Behaviour

All robots run an identical controller, designed in the behaviour-based paradigm [1]. Navigation and obstacle avoidance are provided by two high-level 'behaviors': *Navigate* and *Panic*.

Navigate drives the robot around the environment following walls, turning down corridors and following crumb trails. The navigate behaviour is fairly sophisticated and contains several sub-behaviours, described in detail in [12]. It exploits the accuracy of the SICK laser scanner to make precise movements, turning tight corners and closely following walls.

Panic is designed to cope with situations that confound *Navigate*. It 'unsticks' the robot from obstacles, dead ends and traffic jams. *Panic* runs in parallel with *Navigate*, monitoring the laser scanner; if the robot gets too close to any obstacle, *Panic* overrides ('subsumes') *Navigate*, taking control of the robot until it finds enough space to *Navigate* again.

The controller is carefully designed to make the most of the limited space available in our environment. The robots successfully navigate small rooms and narrow corridors, passing within as little as 20cm of each other before interference occurs.

3.4 Initial search strategy

In this implementation the robots initially search the environment to find goal B by random exploration. They follow walls and turn at random at intersections. 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.

If a path to the resource exists, a robot will eventually reach it. The first robot to reach the resource must have used the most time efficient path yet discovered and all robots now have a list of waypoints describing this path.

3.5 Procedure and data gathered

The robots are started one at a time from the same spot at the home position A, with the same orientation. This position is taken as the the origin of their odometric localization space. On start-up each robot is told it is at Home/A, so it creates a Place with name A at the origin. The robots' local coordinate systems are therefore initially aligned, so we adopt the *optimistic* strategy for receiving trails as described above. This allows a robot to use received trails before it has visited both Places A and B. This speeds up the convergence to a common trail, but makes the system vulnerable to failure if coordinate systems diverge very quickly. The optimistic strategy was found to be acceptable in practice with our implementation.

After initialization, the controller is run and the robot starts to explore the environment. When a robot comes

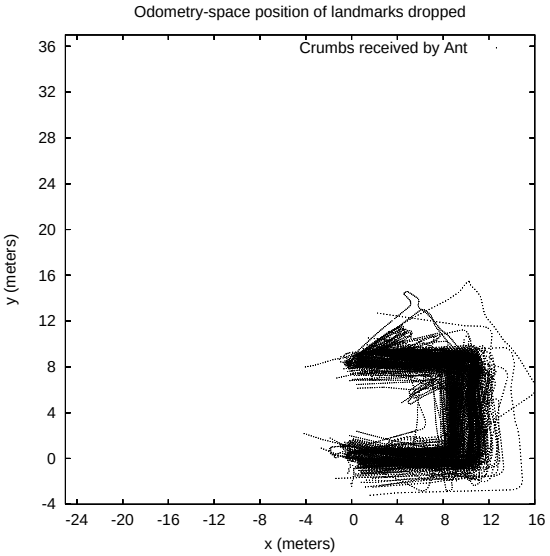

Figure 4: Experiment D: goal searching - paths

within 1m of the home or resource position the experimenter 'gives' or 'takes' a unit of resource (in practice the robot is informed of the state change by a network message). The robots are fully autonomous for the entire trial. The trial was stopped when 3 out of the 4 robots had stopped working.

Every time a robot completed an A-B or B-A trip the time and the name of the robot is noted. A log is also kept of all 1-Crumb Trails broadcast by each robot.

3.6 Results

Table 1 summarizes the results of the experiment, showing the number of trips, distance and lifetime of each robot. They made a total of 262 trips of ≈31m each over 3 hours, 8 minutes, travelling a total of 8.2km (5.1 miles) before the experiment was stopped. Two of the three robots that failed ran out of power; the other suffered catastrophic odometry failure - an occasional problem with the Pioneer robot.

Figure 2(top) shows the time that each robot reached Places A and B, where a visit to Place A is represented by a point plotted low, and Place B by a point plotted high. The regular interval between points in each case shows that the robots were consistently following the trail between the Places. Where there are larger gaps between Places (Fly at 45min, Ant at 160min) the robot was paused to have its battery hot-swapped. The long gap for Bug at 140min was a symptom of its odometry failure - it made two more trips by chance before it failed completely. Figure 2(bottom) shows the accumulated number of trips made over time for each robot, and in total. The gradient of the slope for each robot is very similar, indicating a similar trip frequency. The gradient of the total trip curve decreases as robots fail, as would be expected.

The average time between Places was $(9h43m40s/262) = 2m13s$. This gives an average speed of $0.23ms^{-1}$. The Navigate behavior drives the robots at $0.25ms^{-1}$ when cruising freely. This is the highest speed output by the controller, so we conclude that the team was working at approximately 92% of maximum efficiency, i.e. the robots were following walls in free space in the right direction 92% of their operating time.

Figure 3 shows plots of the Crumbs generated by each robot, both on the plane (top) and extended over time in 3D (bottom). The large variation in the plots indicates that their coordinate systems drifted independently. For example, Bug's coordinate system rotated clockwise; the other robots' systems drifted anti-clockwise. Tanis' system rotated only by around 0.35^o per minute, while Fly's rotated at 2^o per minute. Despite the divergence of the robots' coordinate systems, they were able to share trails throughout the trial.

Figure 4 shows a plot of all 34,743 Crumbs received by Ant, plotted in a fixed global coordinate system; i.e., in which Places A and B are held at fixed points. The trail is noisy, with occasional rotated sections, but the noise is well within the tolerance of the trail following algorithm. The plot shows all 188 minutes worth of Crumbs. At any time during the run, Ant's actual Trail contained many fewer Crumbs, as Crumbs were deleted 4 minutes after generation.

4 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. Waypoints can be shared between divergent localization spaces by specifying them with reference to landmarks common to all robots. Unlike previous landmark-based localization strategies, we use task level features that are necessarily common to all robots. This avoids the cost of sensing and naming physical landmarks.

Our robot team performs a useful general task without supervision for long periods. The system operated autonomously in an unmodified office building, with a total robot run time of 9 hours 45 minutes before being defeated by hardware failure. The trail following method was shown to be robust to individual robot failure, battery hot-swaps and highly-diverged frames of reference. We believe this technique will be of use in a variety of path-finding scenarios, and its simplicity and low computational load make it an attractive alternative to instrumented environments, map building and other techniques that require an explicit global-coordinate system.

The Trail representation and algorithm were designed to generalize to N locations; the next stage is to examine the system's ability to work with multiple locations in a mail-delivery application.

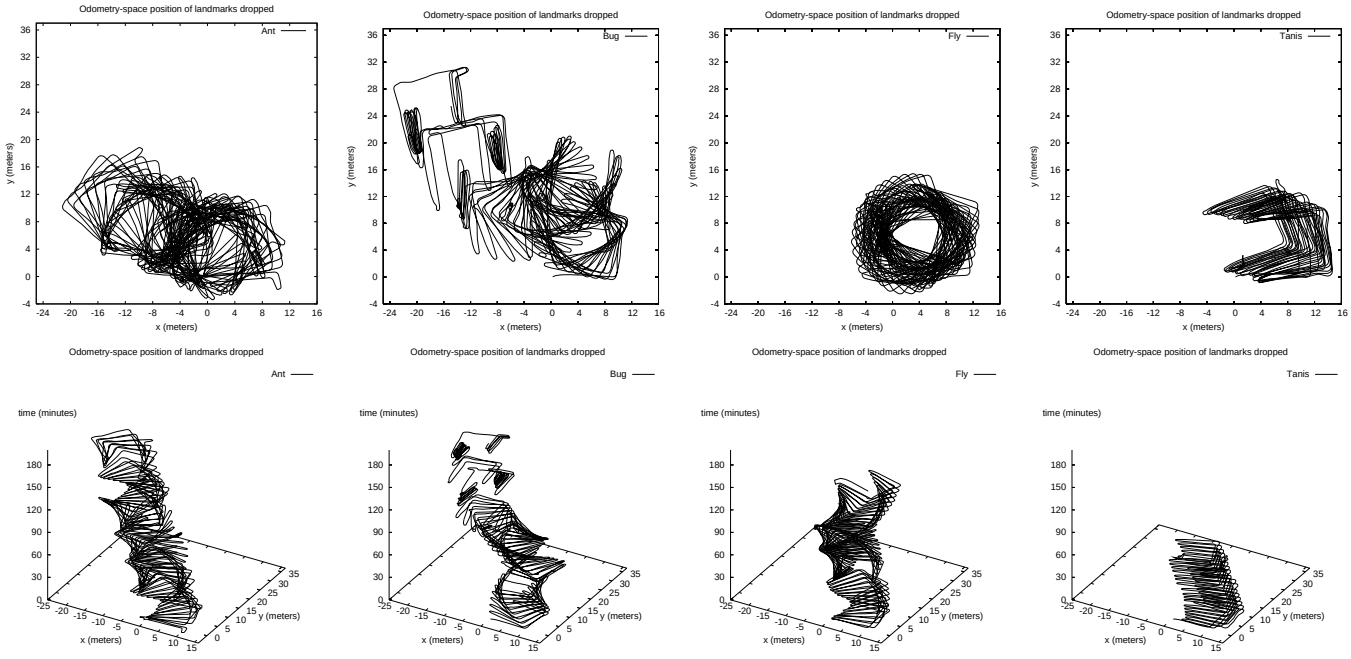

Figure 3: Experiment D: goal searching - paths

Acknowledgements

This work is supported by DARPA grant DABT63-99-1-0015, contract DAAE07-98-C-L028, and NSF grants ANI-9979457 and ANI-0082498 . Thanks to Esben Østergård for help with the *long* experiments, and to Brian Gerkey for his support of the robots.

References

[1] Ronald C. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, MA, 1998.

[2] Y. Cao, A. S. Fukunaga, A. B. Kahng, and F. Meng. Cooperative mobile robotics: Antecedents and directions. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS'95)*, 1995.

[3] 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.

[4] 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.

[5] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environment. *Artificial Intelligence*, (11):391–427, 1999.

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

[7] 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.

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

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

[10] S. Thrun, D. Fox, and W. Burgard. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning 31*, (31):29–53, 1998.

[11] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, to appear 2000.

[12] Richard T. Vaughan, Kasper Støy, Gaurav S. Sukhatme, and Maja J. Matarić. Blazing a trail: insect-inspired resource transportation by a robot team. In L. E. Parker, G. Bekey, and J. Barhen, editors, *Distributed Autonomous Robot Sytems 4*, pages 111–120. Springer, 2000.

[13] Richard T. Vaughan, Kasper Støy, Gaurav S. Sukhatme, and Maja J. Matarić. Whistling in the dark: cooperative trail following in uncertain localization space. In *Proc. Fourth Int. Conf. Autonomous Agents*, pages 187–194, 2000.

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