Building Terrain-Covering Ant Robots:
A Feasibility Study

Jonas Svennebring and Sven Koenig

[1] College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
[2] Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781

Abstract. Robotics researchers have studied robots that can follow trails laid by other robots. We, on the other hand, study robots that leave trails in the terrain to cover closed terrain repeatedly. How to design such ant robots has so far been studied only theoretically for gross robot simplifications. In this article, we describe for the first time how to build physical ant robots that cover terrain and test their design both in realistic simulation environments and on a Pebbles III robot. We show that the coverage behavior of our ant robots can be modeled with a modified version of node counting, a real-time search method. We then report on first experiments that we performed to understand their efficiency and robustness in situations where some ant robots fail, they are moved without realizing this, the trails are of uneven quality, and some trails are destroyed. Finally, we report the results of a large-scale simulation experiment where ten ant robots covered a factory floor of 25 by 25 meters repeatedly over 85 hours without getting stuck.

1 Introduction

In this article, we describe how to build robots that cover closed terrain repeatedly. We say that a team of robots covers terrain repeatedly if and only if every location is swept by the body of some robot repeatedly. Dynamic coverage is an important task in mobile robotics, for example in the context of mine sweeping, surveillance, search and rescue, surface inspection, cleaning hazardous waste, and guarding terrain [13, 16]. Consequently, researchers have developed many coverage methods. Most of these methods make the unrealistic assumption that the robots know their location with certainty. The currently popular POMDP-based robot architectures [21] attempt to overcome this problem by providing robots with the best possible location estimates [34]. However, this approach is complicated and can be brittle for robots that are small and cheap and thus have extremely noisy actuators and sensors. In this article, we explore an alternative, namely trail-laying and trail-following robots (= ant robots). Our inspiration comes from researchers who have studied ant robots that lay trails

and follow the trails laid by other ant robots [29], similar to ants that lay and follow pheromone trails [1, 32]. Ant robots that follow trails arrive at their destinations without having to know their exact locations, which eliminates solving difficult and time-consuming localization tasks. They need only simple sensors, namely sensors that are able to sense the trails, which are artificial landmarks that can be carefully designed to simplify sensing. We utilize a similar idea to build ant robots that cover terrain repeatedly without knowing where they are in the terrain. Similar to ant robots that lay and follow trails, they only need to leave trails in the terrain and sense trails in their neighborhood. Different from ant robots that follow trails, however, they need to move away from trails rather then follow them (to cover terrain that they have not yet covered or not recently covered). We are interested in both single ant robots and teams of ant robots. Teams of ant robots have the potential to cover terrain faster and be more fault tolerant than single ant robots if ant robots can fail. The ant robots do not need to communicate with each other except via the trails, which coordinate the ant robots implicitly and allow them to cover terrain faster than without any communication.

In previous work, we and other researchers have studied theoretically how to build ant robots that cover terrain repeatedly. Unfortunately, the resulting approaches are not well suited for building physical ant robots. In this article, we study more realistic approaches, using both ant-robot simulations that are more realistic than those that have been used previously and a Pebbles III robot. Our ant robots robustly cover terrain even if they do not have any memory, do not know the terrain, cannot maintain maps of the terrain, nor plan complete paths. In particular, they cover terrain even if some ant robots fail, they are moved without realizing this (say, by people running into them and pushing them accidentally to a different location), the trails are of uneven quality, or some trails are destroyed. A possible application of our ant robots is as guard robots that repeatedly cover the rooms of a museum at night. In this case, it is important that the ant robots cover terrain even if partitions or furniture are moved, that they robustly cover terrain even in the presence of error conditions, and that their trajectories are not completely repetitive so that thieves cannot easily predict the movement of the ant robots. This rules out robots that follow fixed tracks [42].

The main contribution of this article is to show the capabilities of robots with extremely limited sensing, processing, and communication capabilities – in the spirit of [12]. We first discuss related work and then real-time search, which provides a solid theoretical foundation for our approach [23, 40]. Second, we describe the physical robot that we use in our experiments. Third, we describe the trail-laying and trail-sensing ant-coverage hardware that we modeled in simulation and the corresponding schema-based ant-coverage software. Fourth, we show how a modified version of node counting, a real-time search method, can model the coverage behavior of our ant robots and thus provides a solid theoretical foundation for them, which is one of our major insights. Fifth, we show that our ant robots continue to cover terrain fast in the long run if they remove their

trails with a cleaning method. Sixth, we report the results of simulation experiments that we performed to understand their efficiency and robustness in error situations. Finally, we describe how we augmented a physical robot with trail-laying and trail-sensing hardware, how we adapted the ant-coverage software to it, and which experiments we performed on it.

2   Related Work

Empirical robotics researchers study ants for two reasons, namely to learn more about them [6] and to build better robots [4]. In the second case, they have studied ant robots that follow trails, either by imitating nature closely [25] or by only getting inspiration from nature. The ant robots have typically left short-lasting trails in the terrain [30], such as heat trails [11, 28], alcohol trails [32] or odor trails [31]. Some ant robots have also used virtual trails only [5, 26, 35]. We, on the other hand, study ant robots that leave actual trails in the terrain to cover it repeatedly. The different task demands both longer-lasting trails (to be able to mark terrain that has already been covered) and different ant-coverage software. Our system is minimalistic but makes assumptions that are practically feasible and robust in the presence of different error conditions, resulting in a physical ant robot implementation. We know of only one other effort to build terrain-covering ant robots, namely ant robots that lay trails of an evaporating liquid by Gabrieli, Katan and Rogel at the Technion, but no results have been reported yet on the success of this project. We will mention other ideas for implementations of ant robots throughout this article.

Robots that deploy markers to map unknown terrain are related to our ant robots. Theoreticians have studied how many markers are needed to map discrete graphs with indistinguishable vertices that can be labeled temporarily by dropping markers, which requires the robots to be able to execute complex reasoning procedures and pick up markers [10, 9, 15, 14]. Robot practitioners, on the other hand, have studied how to map continuous terrain by dropping a large number of smart radio markers that have a small communication range but can store information, such as how many robots have already been within its communication range [8]. In this case, the robots can use much simpler reasoning procedures and do not need to pick up markers. However, they leave visible and expensive markers in the terrain and their communication potentially suffers from cross-talk between the markers.

3   Theoretical Foundation: Node Counting

Theoretical researchers have studied for gross robot simplifications how to build ant robots that cover terrain repeatedly [40]. Real-time search methods are able to cover graphs repeatedly with a small cover time [39, 23], which suggests that they can be used to build ant robots that cover terrain repeatedly. Consequently, we have studied real-time search methods [24] for this purpose in earlier work

[23]. A good overview of real-time search methods is given in [18] and newer developments can be found in [20]. Basically, real-time search methods are graph search methods that associate values with the vertices of the graphs (initially: zero) and update them during plan execution to prevent cycling. They restrict their searches to small parts of the graphs around their current vertices and thus perform agent-centered searches [19]. They first decide on the local search spaces, search them, and determine which edges to traverse within them. Then, they traverse these edges and repeat the process from their new vertices. Figure 1 gives pseudo code for node counting, probably the simplest real-time search method [27]. It can be used to build ant robots by imposing graphs on the terrain, for example, grids. Grids are finite regular four-connected graphs, possibly with some vertices deleted, where the ant robots can always move from their current vertex to each neighboring vertex. We assume for now that ant robots can maintain a discretization of the terrain into cells but will show later that this assumption is unnecessary. Ant robots that use node counting then work as follows: They associate a value with each vertex that counts how often they have visited the vertex already. These values can be interpreted as markings that they leave at the vertices and thus share. When an ant robot enters a vertex, it increases the value of the vertex by one. It then moves to the neighboring vertex with the smallest value, that is, the neighboring vertex that has been visited the least number of times. This way, it attempts to reach a vertex quickly that has been visited the least number of times so far, which is necessary to cover the graph (again). Ties can be broken according to an arbitrary rule. Our ant robots break ties randomly and execute Lines 2 and 3 from Figure 1 together and atomically, for example, because their body covers both their current vertex and all neighboring vertices. Figure 1 demonstrates the coverage behavior of three ant robots that use node counting. White grid cells are empty and gray grid cells are blocked. For simplicity, we make the (unrealistic) assumption in the figure that the ant robots move in a given sequential order and that several ant robots can be in the same grid cell at the same time. If a grid cell contains ant robots, some of its corners are marked. Different corners represent different ant robots. Eventually, the three ant robots cover the grid. In general, one can prove that teams of ant robots that use node counting cover any strongly connected graph repeatedly, by assuming that they do not cover the graph (again) at some point in time. Then, there is some point in time when they only visit those vertices again that they visit infinitely often; they cycle on part of the graph. The values of all vertices in the cycle then increase beyond every bound since the value of each vertex in the cycle is increased by one every time an ant robot enters the vertex. But then the values of all vertices in the cycle increase above the values of all vertices that border the cycle. Such vertices exist since the graph is strongly connected and will not be covered again. Then, however, at least one ant robot is forced to leave the cycle, which is a contradiction [27]. This argument holds no matter in which order the ant robots move even if some ant robots move less often than others. In previous work, we have analyzed the cover time of ant robots that use node counting, where the (first) cover time is the time it takes

Algorithm 1 Node Counting.

We use the following notation: S denotes the finite set of vertices of the graph, and $s_{start} \in S$ denotes the start vertex. $A(s) \neq \emptyset$ is the finite, nonempty set of (directed) edges that leave vertex $s \in S$. $succ(s, a)$ denotes the successor vertex that results from the traversal of edge $a \in A(s)$ in vertex $s \in S$. We also use two operators with the following semantics: Given a finite set X, the expression "one-of X" returns an element of X according to an arbitrary rule. A subsequent invocation of "one-of X" can return the same or a different element. The expression "$\arg\min_{x \in X} f(x)$" returns the elements $x \in X$ that minimize $f(x)$, that is, the set $\{x \in X | f(x) = \min_{x' \in X} f(x')\}$, where f is a function from X to the non-negative integers.

Initially, the values $u(s)$ are zero for all $s \in S$.

1: $s := s_{start}$.
2: $a := $ one-of $\arg\min_{a \in A(s)} u(succ(s, a))$.
3: $u(s) := 1 + u(s)$.
4: Traverse edge a.
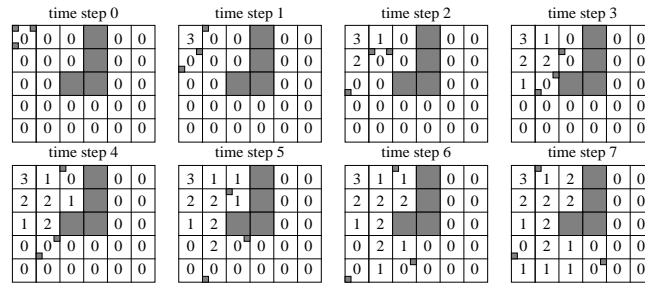5: $s := succ(s, a)$.
6: Go to 2.

Fig. 1. Coverage Behavior of Ant Robots that Use Node Counting.

to visit each vertex at least once. For example, we proved the following theorem in [22] and then adapted it to ant robots in [23].

Theorem 1. *The cover time of teams of ant robots (of a given size) that use node counting on strongly connected undirected graphs can be exponential in the square root of the number of vertices.*

It is currently unknown whether the cover time on grids is smaller than the exponential lower bound of Theorem 1. However, experimental results suggest that teams of ant robots that use node counting cover grids repeatedly with a reasonable cover time that seems to increase only linearly with the number of grid cells. The ant robots do not even need to communicate with each other except via the markings. They only need to have very limited sensing and computational capabilities and do not need to know or learn the graph they are operating on. They only have to sense the markings at their neighboring vertices and change
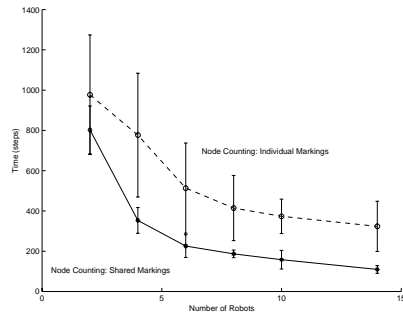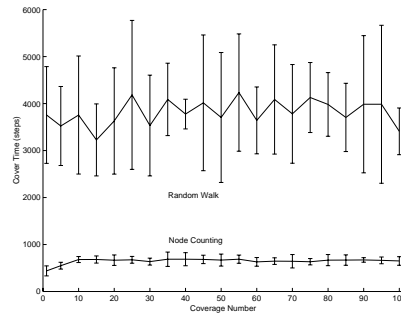
Fig. 2. Number of Ant Robots.

Fig. 3. Number of Coverages.

the marking of their current vertex. Furthermore, ant robots that share their markings cover terrain much faster than ant robots that do not share their markings, that is, where each ant robot maintains its own values at the vertices. Thus, the markings coordinate them implicitly. Figure 2 illustrates the advantage of ant robots that share their markings for obstacle-free grids of size 25 by 25 cells and teams of 2 to 14 ant robots, where 14 ant robots that share their markings cover the grid almost up to three (to be precise: 2.95) times faster than 14 ant robots that do not share their markings.[3]

Ant robots that use node counting have advantages over ant robots that use other methods to cover graphs repeatedly. For example, ant robots that use node counting differ from ant robots that use random walks in that they navigate far more systematically. All of the following cover times for grid-based simulations refer to single ant robots that cover an obstacle-free grid of size 15 by 15 cells for the first time, unless stated otherwise. The graphs labeled "Node Counting" and "Random Walk" in Figure 3 show the cover times of single ant robots that use either node counting or random walks in obstacle-free grids of size 15 by 15 cells, together with plus and minus twice the standard deviation. (Note that

---

[3] Different graphs report on independently performed experiments. Graphs labeled "Node Counting," "Modified Node Counting" and "Random Walk" refer to grid-based simulations where single ant robots cover an obstacle-free grid of size 15 by 15 cells for the first time (measured in steps), unless stated otherwise. Graphs labeled "TeamBots Simulation of Pebbles" and "TeamBots Simulation of Random Walk" refer to simulations in TeamBots where single ant robots cover an obstacle-free terrain of size 10 by 10 meters for the first time (measured in seconds or minutes), unless stated otherwise. Finally, graphs labeled "Pebbles" refer to robot experiments on Pebbles where it covers an obstacle-free terrain of size 2 by 2.5 meters for the first time (measured in seconds or minutes), unless stated otherwise. We need to use different scales for the three kinds of graphs because the cover times are not comparable, not only because the terrain sizes are different but also because they use different clocks. This explains why some of our figures have several x- or y-axes. *In particular, the cover times of simulations in TeamBots and on Pebbles cannot be compared directly.*

this is *not* a confidence interval for the average cover time.) The cover time of ant robots that use node counting remains approximately constant and is much smaller than the one of ant robots that use random walks, even after a large number of coverages. This is interesting because the cover time of random walks on undirected graphs is a small polynomial in the number of vertices [2]. Ant robots that use node counting differ from ant robots that use chronological backtracking (depth-first search) in that they can be suspended and restarted somewhere else, without even knowing that they were moved or where they got restarted. This is important because sometimes ant robots might get pushed accidentally and will not even realize this most of the time.

Unfortunately, despite these advantages, it is very difficult to build physical ant robots based on node counting and other real-time search methods. Their unrealistic assumptions include that the ant robots only move in discrete steps, that the ant robots have markings of a large number of different intensities available, that they can mark vertices uniformly, and so on. Methods that cover continuous terrain [41] assume no actuator or sensor noise or largely depend on random motion. In this article, we describe for the first time how to build physical ant robots that behave similarly to node counting in that they mark the terrain, move in the direction of the smallest value, and always increase the value of their current grid cell. However, they do not write the values directly on the floor. Rather, they lay continuous trails of markings, and the marking densities in an ad-hoc discretization of the terrain around the robot into grid cells encode the values. Thus, they do not need to maintain a discretization of the terrain into cells, which is consistent with the assumption that they do not need to maintain maps of the terrain nor know where they are in the terrain.

4   Physical Robot

Figure 4 (left) shows the physical robot that is the target of our design. Pebbles is a Pebbles III robot from IS Robotics. Its size with the tracks is 45 (width) by 44 (length) centimeters, and the size of its main body is 26 by 41 centimeters. It has two motors for moving on two tracks, and six infrared proximeters (in its front, front-left, front-right, left, right, and rear) as well as bump sensors for sensing obstacles. (It also has sonar sensors for sensing obstacles but we did not use them in our experiments.) The power of its on-board computer is relatively weak. It is based on a Motorola 68000 micro-controller, that executes a special version of Common Lisp and uses several peripheral micro-controllers to control the sensors and actuators. Two re-chargeable 7.2V NiCd batteries enable Pebbles to operate for about 30 minutes.

5   Simulation Setup and Experiments

We decided to design ant robots that create trails by dropping markings, for example, dripping drops of chemical substances such as fluorescence or phosphorescence chemicals. We perform experiments on Pebbles in the second part
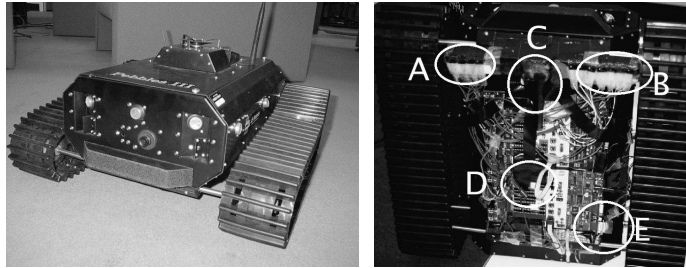
Fig. 4. Pebbles (left: birds-eye view; right: bottom view).

of the article but start out by performing experiments in simulation. We simulate Pebbles in the TeamBots simulator, a realistic multi-robot simulator [7]. Figures 16 and 17 show snapshots of the simulator. For simplicity, we modeled a robot of size 50 by 50 centimeters. The test terrains used in our simulation experiments are discretized into grid cells with size 1 by 1 centimeters. Each grid cell either does not contain the trail chemical or is saturated by it. If a drop of the trail chemical drips onto a grid cell that is already saturated, it might spill to a randomly selected grid cell within a ten centimeter range but has no effect otherwise. We say that the first coverage of the terrain is complete when every grid cell has been swept by the body of some robot at least once (except for cells very close to obstacles). In general, we define an additional coverage of the terrain to be complete when every grid cell has been swept by the body of some robot at least once after the previous coverage was complete. We report cover times that have been averaged over five runs each, unless stated otherwise.

5.1   Ant-Coverage Hardware

We first decide how large the markings should be, how frequently they should be dropped, and how large the trail-sensor field (the sensed floor area below the ant robots) should be. All of these design decisions are interrelated.

Trail-Sensor Field We would like the trail-sensor field to be as small as possible to keep the cost of the sensors small. However, our experiments show that larger trail-sensor fields allow ant robots to cover terrain faster since they increase the information that the ant robots have available about where the markings are. We now list the cover times (in minutes) for TeamBots simulations of single ant robots that cover an obstacle-free terrain of size 10 by 10 meters for the first time. We varied the size of the trail-sensor field (in square centimeters). All other parameters were set to their best values as described in this section. A trail-sensor field of 20×20, 30×30, 40×40, or 50×50 square centimeters resulted in a cover time of 31.3, 24.2, 21.7 or 18.9 minutes, respectively. Thus, we use a trail-sensor field of 50 by 50 centimeters in our experiments, the size of the simulated Pebbles.

Marking Size We would like to keep the markings as small as possible. This way, we avoid having to refill the trail chemical frequently and saturate the terrain less quickly with markings. Fortunately, our experiments show that smaller marking sizes allow ant robots to cover terrain faster. Furthermore, we have found no advantage in using patterns of markings. Consequently, we use the smallest marking size in our experiments that can still be detected reliably. Thus, we use markings of size 1 by 1 centimeters in our experiments, the size of the grid cells used in simulation.

Marking Drop Frequency We would like to drop markings as infrequently as possible, for the same reason why we would like to keep them as small as possible. Fortunately, our experiments show that even marking drop frequencies of only one marking every three seconds result in an acceptable cover time. The optimal marking drop frequency turns out to be about 6.6 markings per second. These results can be explained as follows: Dropping no markings makes ant robots perform random walks. Dropping markings allows ant robots to cover terrain more systematically and thus decreases the cover time. A small marking drop frequency is sufficient as long as the number of markings in the trail-sensor field is small since then the influence of each marking on the movement of the ant robots is high. If the marking drop frequency is too high, trails of markings can become barriers that are time consuming to cross, which can increase the cover time. We now list the cover times (in minutes) for TeamBots simulations of single ant robots that cover an obstacle-free terrain of size 10 by 10 meters for the first time. We varied the marking drop frequency (in markers per second). All other parameters were set to their best values as described in this section. A marking drop frequency of 100.0, 20.0, 10.0, 6.7, 5.0, 3.3, 2.0, 1.0, 0.5 or 0.3 markers per second resulted in a cover time of 22.7, 20.4, 19.3, 18.9, 19.5, 20.1, 19.7, 20.5, 24.1 or 26.1 minutes, respectively. The difference in cover time is fairly small for the marking drop frequencies, given that the standard deviation is large, which makes it difficult to choose the exact best one. However, based on these results, we use a constant marking drop frequency of 6.6 markings per second in our experiments.

5.2   Ant-Coverage Software

Our ant robots use a schema-based navigation strategy [3] with three behaviors that are active at the same time, namely an obstacle-avoidance behavior, a trail-avoidance behavior, and random noise (to simulate the noise in the real world). The schema-based navigation strategy determines in which direction and how fast the ant robots move.

Obstacle-Avoidance Behavior The obstacle-avoidance behavior moves the ant robots away from obstacles and is fairly standard. Our ant robots use three obstacle sensors, corresponding to the front, front-left and front-right infrared proximeters of Pebbles. The obstacle-avoidance vector is the sum of three vectors,
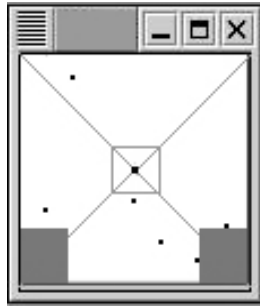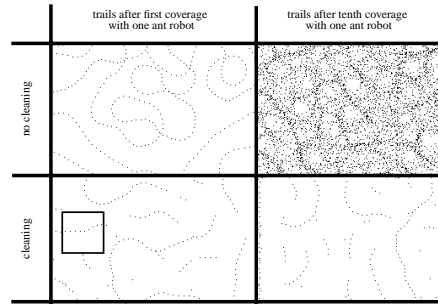
Fig. 5. Partitioning of Sensor Field.   Fig. 6. Trails.

one for each obstacle sensor. Each vector starts at its obstacle sensor and points towards the center of the ant robots. Its length is inversely proportional to the distance of the sensed obstacle from the ant robots.

Trail-Avoidance Behavior The trail-avoidance behavior moves the ant robots away from markings. The trail-avoidance vector has a fixed length. The trail-sensor field is partitioned as shown in Figure 5 to determine the direction of the trail-avoidance vector. The square in the center of the trail-sensor field remains unused to increase stability. The trail-avoidance vector can point in eight directions. It points forward (backward) if the fewest markings are in the front (rear) partition and the left and right partitions have approximately the same number of markings. It points to the left (right) if the fewest markings are in the left (right) partition. It points forward/left (forward/right) if the fewest markings are in the front partition and the left (right) partition has fewer markings than the right (left) partition. Finally, it points backward/left (backward/right) if the fewest markings are in the rear partition and the left (right) partition has fewer markings than the right (left) partition. The trail-avoidance behavior avoids turning the ant robots unnecessarily which makes them fast and allows them to cross trails of markings (because they cannot quickly turn away when they approach trails of markings), which prevents trails of markings from becoming barriers that are time consuming to cross and can decrease the cover time. This is consistent with previous results that also suggest that turns should be avoided unless they are absolutely necessary [17]. We experimented with several other trail-avoidance behaviors but the one described here outperformed them all. We therefore use it in our experiments.

Combining the Behaviors The obstacle-avoidance behavior, the trail-avoidance behavior, and the random noise all produce their own recommendations for how to move. The ant robots always calculate the weighted average of all three vectors and move in the direction of the resulting vector with a speed

that is proportional to the length of this vector. The weight of the obstacle-avoidance behavior cannot be set too low because otherwise ant robots can run into obstacles. On the other hand, it cannot be set too high either because otherwise ant robots do not cover terrain close to obstacles. Similarly, the weight of the trail-avoidance behavior cannot be set too low because otherwise ant robots do not avoid previously covered terrain well enough, which can increase the cover time. On the other hand, it cannot be set too high either because otherwise trails of markings can become barriers that are time consuming to cross, which can increase the cover time as well. To understand this phenomenon, assume that a trail of markings separates two parts of a room that does not contain other trails of markings. This trail of markings is hard to cross for ant robots because they get repelled from it. Furthermore, the trail of markings gets reinforced every time ant robots approach it but do not cross it. The emerging barrier can only be crossed easily once the marking density in the part of the room that ant robots are in has become sufficiently high. We thus optimized the weights of the three behaviors by hand for the physical characteristics of the simulated Pebbles. The weight of the obstacle-avoidance behavior is larger than the weight of the trail-avoidance behavior, and the weight of the trail-avoidance behavior is larger than the weight of the random noise. This reflects, for example, that obstacle avoidance is more important than trail avoidance. Consequently, the obstacle-avoidance behavior has a strong influence on the coverage behavior of our ant robots when they are close to obstacles and the trail-avoidance behavior has a large influence on them when they are farther away from obstacles.

5.3   Regular Coverage

Our ant robots cover closed terrain repeatedly. All of the following cover times for TeamBots simulations refer to single ant robots that cover an obstacle-free terrain of size 10 by 10 meters for the first time, unless stated otherwise. The graph labeled "TeamBots Simulation of Pebbles" in Figure 7 shows how the covered area increases with time until the first coverage is complete. The ant robot covers terrain very quickly at the beginning (when it is easy to find uncovered areas since there are so many of them) but needs more time to find uncovered areas towards the end of the first coverage. It turns out that this property holds for the subsequent coverages as well. In that case, the cover time of the first coverage is 18.9 minutes. An ant robot that moves on an optimal trajectory can cover the terrain in about 3 minutes and 20 seconds if it moves at its maximum speed of one meter per second, although this ideal case cannot be achieved in practice since ant robots cannot follow trajectories precisely and at full speed. (For example, the average speed of our ant robots was less than 50 centimeters per second in our experiments because the obstacle-avoidance and trail-avoidance behavior often counteract each other.) Therefore, we might be able to decrease the cover time of our ant robots with more complex ant-coverage hardware and software. However, our ant robots do well given their small trail-sensor field. For example, the graph labeled "TeamBots Simulation of Pebbles" in Figure 8 shows that they have initially a much smaller cover time than ant robots that
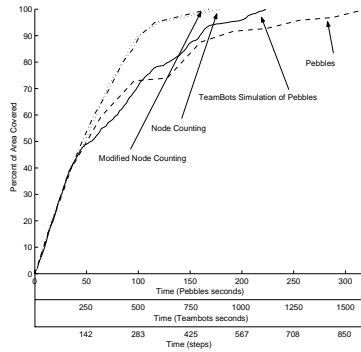
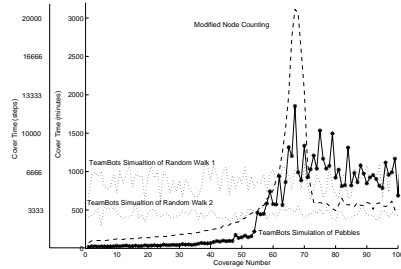**Fig. 7.** Covered Area as a Function of Time.

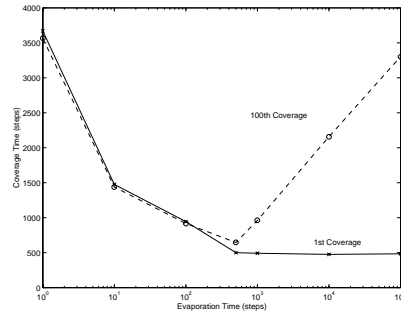**Fig. 8.** Number of Coverages.

Fig. 9. Evaporation Time.

use random walks, for two different implementations of random walks that are similar to Probabilistic Covering [41]. The graph labeled "TeamBots Simulation of Random Walk 1" refers to ant robots which were identical to our ant robots but did not drop markings. With this change, the ant robots had difficulties with traversing small passages. They continued to cover the terrain but their cover time is about 800 minutes, much larger than the cover time of our ant robots. The graph labeled "TeamBots Simulation of Random Walk 2" refers to ant robots which travel the length of their body between sensor readings and are therefore much less affected by the random sensor noise. This implementation reduces the cover time to about 450 minutes but the cover time remains much larger than the one of our ant robots.

A problem of our ant robots is that their cover time increases with the number of coverages. For example, the cover time of the first coverage is 18.9 minutes. It is 36.6 minutes for the tenth coverage and larger than two hours for the 50th coverage already. The top row of Figure 6 shows that the terrain gets saturated with markings over time. (The square in the lower left corner corresponds to the simulated Pebbles.) This makes it harder to place new markings and decreases

the influence of each newly placed marking on the navigation behavior of the ant robots which increases the cover time. Eventually, the terrain is completely saturated with markings and the coverage behavior of our ant robots then resembles random walks, which we verified by saturating the terrain with markings. Thus, the cover time of our ant robots approaches the cover time of ant robots that use random walks, which is about 40 times larger than the initial cover time. We show later how this problem can be solved by letting our ant robots remove their markings.

5.4   Computational Models of Ant Robots

We have described node counting as the inspiration behind our ant robots. Consequently, it is important to study whether it is a good computational model of them, for example, given that node counting operates on discrete graphs but our ant robots operate in continuous space. The graphs labeled "Node Counting" and "TeamBots Simulation of Pebbles" in Figure 7 show that the area that ant robots that use node counting and our ant robots cover increases in a similar way over time until the first coverage is complete. However, the graph labeled "Node Counting" in Figure 3 and the graph labeled "TeamBots Simulation of Pebbles" in Figure 8 show that the cover time increases differently with the number of coverages. The cover time of ant robots that use node counting remains small and much below the cover time of ant robots that use random walks even after a large number of coverages, whereas the cover time of our ant robots approaches the cover time of ant robots that use random walks. This difference is due to the fact that node counting does not model that the terrain gets saturated with markings over time. We therefore modify node counting to provide a better computational model of our ant robots. Figure 2 gives pseudo code for the modified version of node counting. Remember that ant robots that use node counting increase the value of their current grid cell by one and then move to the neighboring grid cell with the smallest value, breaking ties randomly. Ant robots that use the modified version of node counting, on the other hand, increase the value of their current grid cell by one only with probability $(k - x)/k$, where x is the current value of the grid cell and k is a constant (we use $k = 170$). Otherwise they leave the value of their current grid cell unchanged. Then they move to the neighboring grid cell with the smallest value, breaking ties randomly. Thus, the probability with which ant robots that use the modified version of node counting increase the value of a given grid cell by one gets smaller and smaller over time, until the value of the grid cell is k and then does not change any longer. To understand the idea behind the modified version of node counting, assume that each grid cell is divided into k small areas that are initially unmarked. Let x be the number of marked areas. If the ant robots randomly select one area of their current grid cell and mark it, then x increases by one with probability $(k - x)/k$, the probability that the chosen area was unmarked. Otherwise, x remains unchanged. The modified version of node counting is a somewhat simplistic computational model of our ant robots but does model that it becomes more and more difficult to add markings to terrain that already contains a large number of markings,

---

Algorithm 2 Modified Version of Node Counting.

Initially, the values $u(s)$ are zero for all $s \in S$.

1: $s := s_{start}$.
2: $a :=$ one-of $\arg\min_{a \in A(s)} u(succ(s, a))$.
3: With probability $(k - u(s))/k$ do: $u(s) := 1 + u(s)$.
4: Traverse edge a.
5: $s := succ(s, a)$.
6: Go to 2.

---

and that the terrain gets saturated with markings over time and the coverage behavior of our ant robots then resembles random walks. The modified version of node counting also models an effect that we did not anticipate. The graphs labeled "Modified Node Counting" and "TeamBots Simulation of Pebbles" in Figure 8 show that the cover times of both ant robots that use the modified version of node counting and our ant robots peak and only then reduce to the cover time of ant robots which were identical to our ant robots but did not drop markings. The modified version of node counting predicted the peak first. We then ran experiments with our ant robots to verify it. It turns out that the peak is due to local minima in the marking density. Consider, for example, a part of the terrain that is not completely saturated with markings but is enclosed by terrain that is completely saturated. It then takes both ant robots that use the modified version of node counting and our ant robots a long time to leave this part of the terrain, longer than ant robots that use random walks, since they first need to increase the trail density in their area to that of the surrounding ones, which takes time. This explains the peak. We assumed that the ant robots that use the modified version of node counting increase the value of their current grid cell by one with probability $(k - x)/k$. The integer k is a parameter that determines how quickly the terrain gets saturated with markings. We determined empirically that $k = 170$ makes the peaks of the cover times coincide in our experiment. To summarize, the modified version of node counting, similar to node counting itself, is a reasonably good computational model for predicting how the area covered by our ant robots increases over time until the first coverage is complete, as the graph labeled "Modified Node Counting" in Figure 7 shows. However, the modified version of node counting, different from node counting itself, is also a reasonably good computational model for predicting how the cover time of our ant robots increases with the number of coverages, as the graph labeled "Modified Node Counting" in Figure 8 shows.

5.5   Scaling Up: Removing Trails of Markings

It is undesirable that the terrain gets saturated with markings over time since this increases the cover time. There have been several ideas for how to map terrain with ant robots that use evaporating trails (similar to pheromone trails of ants) and can sense the strength of the trails [37, 39–41]. An obvious example
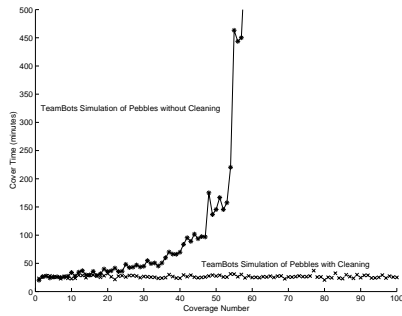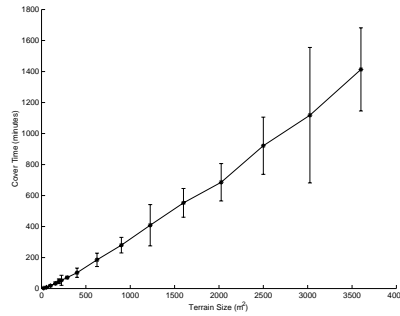
Fig. 10. Number of Coverages.

Fig. 11. Terrain Size.

is evaporating liquids. Another example is vacuum cleaning where the absence of dirt forms trails and the accumulation of dirt makes them evaporate [36]. However, evaporating markings are problematic for our ant robots because the evaporation rate needs to get optimized for each application, for example, the size of the terrain and the speed of the ant robots. Carefully tuned evaporation rates result in small cover times. However, if the evaporation rate is too high, then the markings do not last sufficiently long to guide the ant robots away from terrain that they have already covered recently, which increases the cover time. In this case, the cover time of the ant robots remains constant over time, similar to the one of ant robots that do not drop markings, and Figure 8 has already shown that this cover time is large. On the other hand, if the evaporation rate is too low, then the terrain gets close to saturated with markings over time, which increases the cover time as well. In this case, the cover time of the ant robots increases over time and eventually becomes similar to the one of ant robots that use markings that do not evaporate. Figure 8 has already shown that this cover time is large in the long run.

Figure 9 shows how the cover time varies with the evaporation rate for the modified version of node counting, both for the first coverage and the one hundredth coverage. The x-axis shows how many steps it takes for markings to disappear. For example, if the evaporation time is 1000, then the value of a cell automatically decreases by one exactly 1000 steps after it was increased by one. The figure shows that the cover time is the same as that of a random walk if the evaporation time is small or, equivalently, the evaporation rate is high. The figure also shows that the first cover time is small if the evaporation time is large. However, the one hundredth cover time is already large. To summarize, the cover time is very sensitive to the evaporation rate.

Since it is often difficult in practice to tune the evaporation rate and we want our ant robots to be able to operate in terrain of initially unknown size, we propose to use long-lasting markings that the ant robots remove themselves by removing all markings in two cleaning areas although we could not find any references in the literature to ant robots that remove their trails. Depending on

the trail material, the trails could for example be removed with brushes, vacuum cleaners, heat (for alcohol trails), and light (for some photo chemicals) but it is future work for us to build such hardware. It is an important design decision where to place the cleaning areas to avoid that ant robots remove markings immediately after they placed them or more subtle problems that result from them perceiving terrain with removed markings as not having been covered yet. This can result in ant robots following other ant robots or ant robots moving back to terrain that they just covered. We therefore let our ant robots add markings in the center below their bodies but remove them on either side below their bodies. Figure 5 shows the cleaning areas in gray.

The cover time of our ant robots without cleaning is often smaller during the first five coverages than the one of our ant robots with cleaning because removing markings deletes information that is especially important when markings are still sparse. For example, the first cover time of ant robots without cleaning is about 16 percent smaller than the one of ant robots with cleaning. After ten coverages, however, the cover time of ant robots with cleaning is about 23 percent smaller than the one of ant robots without cleaning because removing markings prevents the terrain from getting saturated with them, as Figure 6 shows. The cover time of our ant robots with cleaning is only 26.4 minutes in the long run and thus much smaller than the cover time of our ant robots without cleaning. The graph labeled "TeamBots Simulation of Pebbles with Cleaning" in Figure 10 shows that the cover time remains a small constant even after a large number of coverages, similar to the cover time of ant robots that use node counting, as the graph labeled "Node Counting" in Figure 3 shows. Furthermore, the deviation from the average cover time is small. We therefore use ant robots with cleaning in all experiments with multiple terrain coverages.

5.6   Simulation Experiments

Based on the experimental results reported in the previous sections, we decided to use a trail-sensor field of the size of the simulated Pebbles, markings of size 1 by 1 centimeters, a marking drop frequency of 6.6 markings per second, and (in all experiments with multiple terrain coverages) cleaning. The line in Figure 16 shows the beginning of the resulting trajectory of an ant robot that conforms to these design decisions. We now report experiments that we performed to understand the coverage behavior of our ant robots better.

Scaling with Terrain Size Figure 11 shows how the cover time increases with the size of the terrain. The cover time increases linearly with the size of the terrain, and the variance of the cover time increases with the size of the terrain. This is an important result because Theorem 1 suggested the possibility that the cover time could have increased super-linearly with the size of the terrain, which would have implied that very large teams of ant robots are needed to cover large terrains. Our experiment shows that the cover time scales well with the size of the terrain. This result is important because it establishes that it is feasible to cover large terrains with our ant robots.
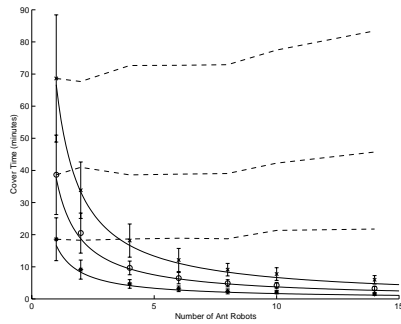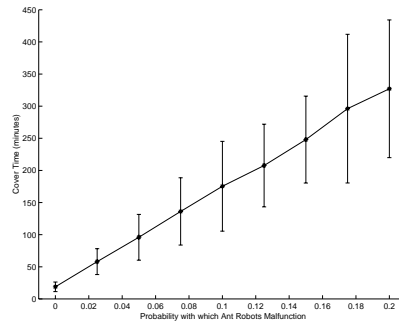
Fig. 12. Number of Ant Robots.   **Fig. 13.** Probability of Malfunction.

Scaling with the Number of Ant Robots Figure 12 shows how the cover time decreases with the number of ant robots for different terrain sizes, averaged over 25 runs each. The three solid graphs correspond to the cover times, from top to bottom, for terrains of size 20×20, 15×15 and 10×10 meters. The three dashed lines correspond to the product of the cover time and the number of ant robots for the same terrain sizes. The cover time is inversely proportional to the number of ant robots, and the product of the cover time and the number of ant robots increases only very slowly. Thus, the cover time can be roughly calculated by dividing the area in square meters by six times the number of robots. This implies that the cover time can be decreased by increasing the number of ant robots and that there is not much overhead when increasing the number of ant robots. These results are similar to our earlier theoretical results for gross robot simplifications [23] except that increasing the number of ant robots has an additional advantage here. Trails of markings can become barriers that are time consuming to cross, which can increase the cover time. The more ant robots there are, the less this is an issue since it becomes more likely that there are ant robots on both sides of the barrier. Furthermore, the ant robots are able to cross barriers more easily since they get repelled from each other (since they get repelled from obstacles) which might push them across barriers. This result is important because teams of ant robots cover closed terrain faster and are more fault tolerant than single ant robots.

Coping with Error Conditions One of the attractive properties of ant robots is that they cover closed terrain robustly even in situations where some ant robots fail, they are moved without realizing this, and some markings are destroyed. We demonstrated these properties earlier for gross robot simplifications [23] and show in the following that our ant robots share these advantages. We use a single ant robot in the experiments.

Failing Ant Robots We first measure the cover time when ant robots fail. This is important because ant robots can malfunction. Every 1.6 seconds in the experi-
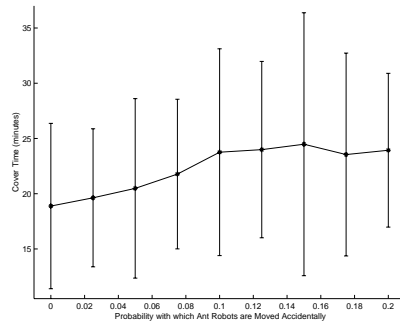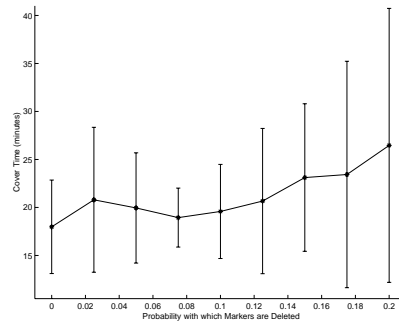
Fig. 14. Probability of Displacement.

Fig. 15. Probability of Erasure.

ment, a functional ant robot failed with a given probability and then remained motionless for an amount of time that was uniformly distributed between 0.0 and 4.6 minutes. Figure 13 shows how the cover time increases with the failure probability, averaged over 50 runs each. The cover time increases linearly with the failure probability (this can also be derived analytically), and the variance of the cover time increases with the failure probability. Thus, our ant robots continue to cover terrain robustly and with a cover time that increases gracefully with the failure probability.

Moving Ant Robots We now measure the cover time when ant robots are accidentally moved without realizing this ("kidnapped robot problem"). These problems are important because people or other ant robots can easily run into them and accidentally push them to a different location. Ant robots should be able to cope well with this problem since they do not need to know their current location. Every 1.6 seconds in the experiment, an ant robot was moved with a given probability in a random direction for a random distance. Figure 14 shows how the cover time increases with the movement probability, averaged over 50 runs each. The cover time remains small even if the probability of being moved is 0.2, which is unrealistically high. Thus, our ant robots continue to cover the terrain robustly with a cover time that increases only slightly and thus gracefully with the probability of being moved. This result is not surprising since some random displacements move the ant robots to locations that they have already visited (which can increase the cover time) while others move them into unvisited terrain (which can decrease the cover time).

Destroying Markings Finally, we measure the cover time when some markings are accidentally destroyed. This is important because markings can get destroyed due to wind, dust, rain, humans, or other ant robots. We test two different scenarios.

- In the first scenario, randomly selected individual markings are destroyed. This can happen, for example, due to wind and dust. During each time

step in the experiment, one randomly selected marking was destroyed with a given probability. Our ant robots continue to cover the terrain robustly and with a cover time that does not change significantly. This is so because the number of markings in the terrain was in the thousands. Thus, destroying a small number of markings uniformly across the terrain decreases the marking density somewhat but has no large impact otherwise.

– In the second scenario, all markings in randomly selected areas are destroyed. This more systematic destruction of markings can happen, for example, due to humans walking around. During each time step in the experiment, all markings in one randomly selected area of size 20 by 20 centimeters were destroyed with a given probability. Figure 15 shows how the cover time increases with the probability of destroying areas of markings, averaged over 10 runs each. Our ant robots continue to cover the terrain robustly and with a cover time that increases only slightly and thus gracefully with the probability of areas of markings being destroyed.

Large-Scale Experiment To demonstrate that a large team of our ant robots covers a large terrain robustly, we placed ten of our ant robots into a terrain of size 25 by 25 meters that resembled a factory floor with two production lines and a number of office rooms, as shown in Figure 17. This complex but very realistic environment is very difficult to cover due to its many narrow passages. The ten ant robots were started in three groups but quickly distributed evenly across the factory floor. They covered the factory floor 35 times during 85 hours, with an average cover time of about 146.9 minutes. This experiment shows that it takes longer to cover the factory floor than the simple terrains used so far, even if we take the size difference into account. (Figure 11, for example, shows that a single ant robot can cover a simple terrain of size 25 by 25 meters in 185.0 minutes.) Thus, the shape of terrain has an important influence on the cover time, although this influence is hard to quantify and currently not well understood (although there are first ideas for some approaches) [40].

6   Physical Robot Setup and Experiments

The simulation shows the promise of our ant-coverage approach. Although the simulation environment is realistic, it cannot model all details. Thus, we also need to demonstrate the promise of our ant-coverage approach on Pebbles itself.

6.1   Changes to the Ant-Coverage Hardware

We added both a trail-laying and a trail-sensing mechanism to Pebbles. The new components are marked in Figure 4 (right). The trail-laying mechanism uses a simple black pen with a thick tip to lay continuous trails. The pen is mounted below the front center of the body of Pebbles, as shown on Figure 4 (right) in position C. The trail-sensing mechanism uses two one-dimensional trail-sensor arrays of proximeters that are mounted to the right (position A)
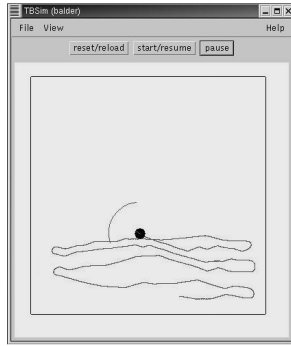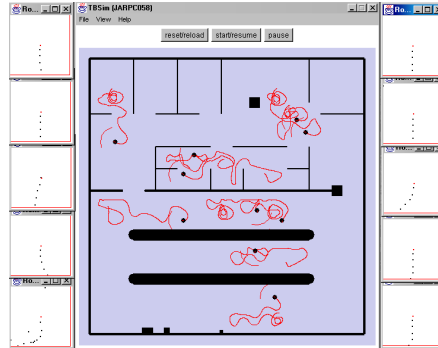
Fig. 16. Sample Coverage Behavior.

Fig. 17. Large-Scale Experiment.

and left (position B) of the pen. This allows Pebbles to sense its trails right away and avoids that it gets trapped in local minima in the trail density. Each trail-sensor array consists of four Sharp 2L01 proximeters and covers an area of about 4 by 1 centimeters. Each proximeter detects the amount of light that is reflected from an LED via the floor to it. This allows it to detect trails since there is less surface reflection in darker areas, that is, on trails. The signal from the proximeter is then sent through a multiplexer to an analog-to-digital converter integrated in an Atmel AVR Mega163 micro-controller (position D) and sampled approximately 2000 times a second. The value that corresponds to the darkest area is stored until a read command is sent from the external computer via an RS232 interface (position E) approximately every five times a second. The value is then thresholded and reported and subsequently reset to zero. This allows Pebbles to move fast without missing trails. To summarize, Pebbles differs from its simulation in that it lays continuous trails by dragging a pen rather than discrete markings by dripping chemical substances and in that it uses only a one-dimensional trail-sensor field rather than a two-dimensional one. We did this for simplicity (price and ease of implementation) since this was our first feasibility study on a physical robot. In simulation, discrete markings proved to be advantageous over continuous trails since, this way, one avoids having to refill the trail chemical frequently and at the same time saturates the terrain less quickly. Pebbles could later be equipped with a servo motor to lift the pen, which would allow it to lay discrete markings with the pen rather than continuous trails even though the marking drop frequency would be harder to control and optimize than when dripping a liquid. However, since Pebbles uses a one-dimensional trail-sensor field rather than a two-dimensional one, discrete markings have the disadvantage that Pebbles would need to use a complex history mechanism to remember where the trails are. This explains our decision to let Pebbles lay continuous trails.

6.2   Changes to the Ant-Coverage Software

Our simulated ant robots use a schema-based navigation strategy with three behaviors, namely an obstacle-avoidance behavior, a trail-avoidance behavior, and random noise. Pebbles uses a similar schema-based navigation strategy with two behaviors, namely the obstacle-avoidance behavior and the trail-avoidance behavior. Pebbles switches the schema-based navigation strategy off only in the rare occasion when one of its bump sensors triggers, that is, if it has run into an obstacle. In this case, it moves away from the obstacle for a short period of time before it resumes its normal operation.

Obstacle-Avoidance Behavior As in simulation, the obstacle-avoidance vector is the sum of vectors, one for each obstacle sensor. This time, however, we used all six infrared proximeters of Pebbles, not just three as in simulation, and calculate the obstacle-avoidance vector as the weighted sum of the corresponding six vectors to achieve a smooth obstacle-avoidance behavior. The weight of each vector is proportional to the importance of its obstacle sensor. The weight of the front obstacle sensor is $1/10$, the weight of the rear obstacle sensor is $1/20$, the weight of the front-left and front-right obstacle sensors is $1/40$, and the weight of the left and right obstacle sensors is $1/55$. The weights reflect, for example, the fact that obstacles in front of Pebbles are more important than obstacles in its rear.

Trail-Avoidance Behavior The trail-avoidance vector has a fixed length of 0.1. Pebbles uses a one-dimensional trail-sensor field to sense trails rather than the two-dimensional one used in simulation. There are four proximeters in the left trail-sensor array and four proximeters in the right trail-sensor array. Each proximeter of the left (right) trail-sensor array that senses a trail changes the angle of the vector by 17 degrees to the right (left). Thus, if all eight proximeters sense trails, then the vector points straight ahead. If all four proximeters of the left trail-sensor array sense trails but no proximeter of the right trail-sensor array senses a trail, then the vector points 68 degrees to the right. A problem with this approach is that a one-dimensional trail-sensor field senses only a small part of the terrain. This makes it difficult for Pebbles to determine a good trail-avoidance vector based on the current information of the trail-sensor field alone. Pebbles could simulate a virtual two-dimensional trail-sensor field that starts behind the one-dimensional one by keeping track of how the trails sensed by the one-dimensional trail-sensor field move within the area of the virtual two-dimensional one. This would allow it to use the trail-avoidance behavior used in simulation. However, it is difficult for Pebbles to maintain a virtual two-dimensional trail-sensor field since it does not know about trails that enter the virtual two-dimensional trail-sensor field from the sides other than the front (which is a problem, for example, in corners) and cannot keep track of the exact movements of the trails within the virtual two-dimensional trail-sensor field. We thus implemented a different solution, where Pebbles uses only a limited amount

of history. It calculates the direction of the new trail-avoidance vector as above but then adds the direction of the old trail-avoidance vector to it, weighted with a decay factor smaller than one. (If the resulting angle is larger than 90 degrees to the left or right, it gets reduced to 90 degrees.) This way, the new trail-avoidance vector is influenced not only by the current information from the trail-sensor field but also the information in the recent past. The decay factor ensures that the influence of trails decays over time, since they move further away from Pebbles and their actual location is known less precisely. This approach preserves the idea of a virtual two-dimensional trail-sensor field but is easier to implement. If Pebbles continues to detect trails on the same side, it turns more and more sharply away from that side. If it stops detecting trails, it turns less and less sharply until it moves straight again. The decay factor cannot be set too low because otherwise Pebbles makes many small turns when it senses trails. These turns need to be sharp to turn Pebbles sufficiently. Thus, the motion of Pebbles is jerky when it senses trails. On the other hand, the decay factor cannot be set too high either because otherwise Pebbles continues to slowly turn even long after it has stopped sensing trails. We determined experimentally that a decay factor of 0.5 resulted in smooth trajectories that turn Pebbles for only a short time after it senses a trail. So, if exactly one proximeter of the left trail-sensor array of Pebbles always senses a trail, then Pebbles will eventually turn 34 degrees to the right. Usually, the trail is no longer below its body before it achieves this turn angle, resulting in turns of only approximately 30 degrees before it moves straight again. The following table gives a fictitious example of exactly how the direction of the trail-avoidance vector is computed over time, assuming for simplicity that each trail is sensed by only one proximeter at a time. All angles are relative to Pebbles and were rounded to integers.

Time	Event	Old Angle (in degrees)	New Angle (in degrees)
			0.0000
1	no trail	0.0000	$0.0000 + 0.5 \times 0.0000 = \quad 0.0000$
2	trail on the right side	0.0000	$-17.0000 + 0.5 \times 0.0000 = -17.0000$
3	no trail	-17.0000	$0.0000 - 0.5 \times 17.0000 = \quad -8.5000$
4	trail on the right side	-8.5000	$-17.0000 - 0.5 \times 8.5000 = -21.2500$
5	trail on the left side	-21.2500	$17.0000 - 0.5 \times 21.2500 = \quad 6.3750$
6	no trail	6.3750	$0.0000 + 0.5 \times 6.3750 = \quad 3.1875$

Combining the Behaviors As in simulation, Pebbles always calculates the weighted average of the obstacle-avoidance vector and the trail-avoidance vector, and moves in the direction of the resulting vector with a speed that is proportional to the length of this vector. As in simulation, the coverage behavior of Pebbles is sensitive to the choice of the weights. The last turn in Figure 18 (left), for example, shows a situation where the trail became a barrier for Pebbles. This problem occurs frequently when the weight of the trail-avoidance behavior is set too high. Figure 18 (right), on the other hand, shows a situation where Pebbles crossed orthogonal trails. Pebbles came from the left and turned right to
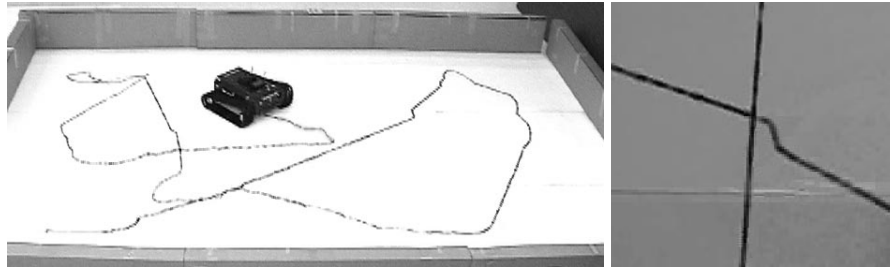
Fig. 18. Trail-Avoidance Behavior.

turn away from the trail but continued to move forward and thus crossed the vertical trail before turning. It then turned left, again to turn away from the vertical trail, and eventually continued on its old trajectory. If the weight of the trail-avoidance behavior is tuned carefully, this behavior clearly dominates.

6.3   Physical Robot Experiments

At this point, we have only one physical ant robot available and it does not have a cleaning mechanism, which limits the experiments we can perform with physical ant robots. We conducted experiments that are possible with this setup to check whether the results on Pebbles are similar to those in simulation. We used terrains of sizes ranging from 2 by 2.5 meters to 3 by 4.5 meters. Their floor was covered with white paper, and they were surrounded with brown cardboard walls. Pebbles is only a first crude realization of the simulated ant robots and thus has weaker capabilities. For example, Pebbles uses only a very small one-dimensional trail-sensor field rather than a much larger and two-dimensional one and thus has a lot less information available about the trails in its vicinity, lays continuous trails rather than discrete markings, cannot sense trails nor move without noise, and cannot lay trails of consistently high quality. Furthermore, we used only crude estimates for the parameter values of the ant-coverage software on Pebbles but performed extensive experiments to tune them in simulation. Similarly, the environment that Pebbles operates in is more challenging than the simulated one since simulations are always idealized to some degree even if they are sophisticated. Despite the more challenging conditions of the experiments with Pebbles, the outcomes are similar to those in simulation and thus confirm the results of our simulation experiments. For example, we noticed the same problems on Pebbles that we already noticed in simulation, including that trails can become barriers that are time consuming to cross, which can increase the cover time. No barrier built up between the two rooms in Figure 19 (right) during the first coverage and Pebbles switched from one room to the other six times. However, a slight barrier built up during the second coverage. Pebbles tried to switch five times between the two rooms but was successful only three of the five times.

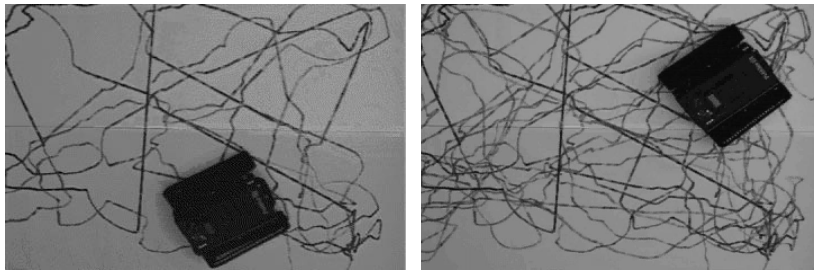**Fig. 19.** Test Terrain (left: 2 by 2.5 meters; right: 2.5 by 4 meters).

Fig. 20. End of First and Third Coverage.

Regular Coverage We first verify that Pebbles covers terrain of different shapes multiple times without getting stuck, including the ones shown in Figure 19 (left and right). This is indeed the case. We then measured the cover time. All of the following cover times for Pebbles refer to it covering an obstacle-free terrain of size 2 by 2.5 meters for the first time, unless stated otherwise. Pebbles needed 320 seconds for the first coverage. The graph labeled "Pebbles" in Figure 7 shows that the covered area increases over time in a similar way to how the covered area increases in simulation until the first coverage is complete. Pebbles needed 329 seconds for the second coverage and 489 seconds for the third coverage. As in simulation, the cover time increases because the terrain gets saturated with trails. For example, Figure 20 (left and right) shows the trails after the first and third coverage. However, as in simulation, Pebbles has initially a much smaller cover time than when it uses random walks. When the random walks were implemented by making Pebbles not lay trails but keeping everything else unchanged, it moved along the walls and covered about 50-60 percent of the terrain in 451 seconds but never covered the terrain completely in a reasonable amount of time. When the random walks were implemented by moving Pebbles for about one second each into a random direction, it only moved back and forth within a small area. It looks like the noise we added in TeamBots is beneficial for ant robots that use random walks and explains why the ant robots that used random walks in simulation performed better than Pebbles when it used random walks.
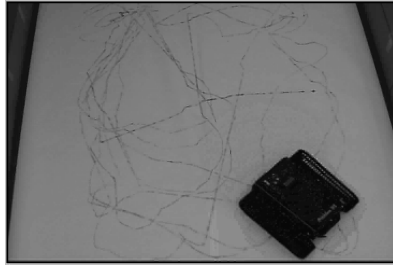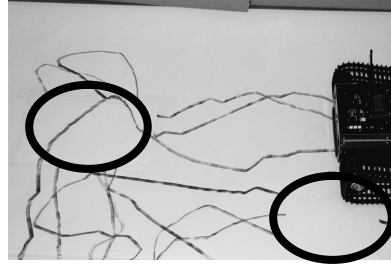
Fig. 21. Low-Intensity Trails.   **Fig. 22.** Destroyed Areas of Trails.

Coping with Error Conditions We have already demonstrated in simulation that our ant robots cover closed terrain robustly even in situations where they are moved without realizing this and some trails are destroyed. We now demonstrate that these properties continue to hold on Pebbles. In addition, we also demonstrate that Pebbles covers closed terrain robustly even if its trails are of uneven quality. It is future work to study the effect of smearing or moving markings (for example, due to wind or ant robots driving over them) since the trail-laying mechanism used by Pebbles made it difficult to create this condition. In this case, markings can be moved to areas that have not or not recently been covered, which can delay the coverage of these areas. One way to address this issue is to use patterns of markings since they get destroyed when markings are smeared or moved.

Moving Pebbles We first measure the cover time when Pebbles is accidentally moved without realizing this. In the middle of a run, we moved Pebbles twice by a couple of meters in a random direction. The resulting cover time of 385 seconds is larger than its regular cover time of 320 seconds but the difference is not significant. This result is similar to the one in simulation.

Trails of Uneven Quality We now measure the cover time when the pen of Pebbles was nearly exhausted. This is important because its pen is not constantly refilled with ink and its trails thus get lighter over time and harder to detect, as shown in Figure 21. This makes it more likely that Pebbles misses trails. Since Pebbles moves at different speeds the pressure on the pen changes and the trails are not only faint but also of uneven quality. Since the intensity of the trails adds up over time, Pebbles continued to cover terrain robustly although the resulting cover time of 573 seconds is larger than the regular cover time of 320 seconds. This difference is significant. The coverage became also more uneven since some trails were stronger than others and became barriers that were difficult to cross, which can increase the cover time as well.

Destroying Areas of Trails Finally, we measure the cover time when some trails are destroyed. After Pebbles had covered about 90 percent of the terrain, we

randomly placed three sheets of paper of size 15 by 20 centimeters on terrain that it had already covered and that thus contained trails. Pebbles covered the three areas again only 103 seconds later since it was drawn to each area once it had sensed part of it and thus noticed that it did not contain trails. This result is similar to the one in simulation. Figure 22 shows a snapshot of the experiment. The two bold ovals mark erased patches of terrain, one of which has already been covered again.

7   Conclusions

In this article, we have described how to build ant robots that leave trails in the terrain to cover closed terrain repeatedly. The trails convey information to our ant robots, which allows them to cover terrain more systematically and faster than with random walks. Our ant robots do not need to be localized, which completely eliminates solving difficult and time-consuming localization tasks. Furthermore, our ant robots do not need to know a map of the terrain and the terrain can change over time, for example, because people move furniture around. We showed that a modified version of node counting, a real-time search method, can model their coverage behavior and and thus provides a solid theoretical foundation for them. We also discussed how our ant robots continue to cover terrain fast in the long run if they remove their trails with a cleaning method. To validate our claims, we performed experiments both in simulation and on a physical robot. For example, we reported the results of a large-scale experiment in a realistic simulation environment where ten ant robots covered a factory floor of 25 by 25 meters repeatedly over 85 hours to demonstrate that they indeed cover closed terrain repeatedly over long periods of time without getting stuck. We also showed experimentally that our ant robots robustly cover terrain even if they are moved without realizing this (say, by people running into them), some ant robots fail, the trails are of uneven quality, and some trails are destroyed. These properties were confirmed, where possible, on a Pebbles III robot from IS robotics. We are now working on a better trail-laying mechanism for Pebbles and a cleaning mechanism. We are also working on building several ant robots so that we can demonstrate the advantages of teams of ant robots not only in simulation but also on physical ant robots. Finally, the purpose of this article was to demonstrate the robustness of our minimalistic ant robots despite their limited ant-coverage hardware and simplistic ant-coverage software. We are now working on more sophisticated ant-coverage hardware and software that decreases the cover time of our ant robots even more while continuing to let them cover closed terrain robustly without knowing where they are. Finally, we are working on comparing our ant robots to other coverage algorithms, including more traditional ones, even if they make less realistic assumptions or are less robust than our ant robots. In a first simulation study on grids, we have already compared ant robots that use node counting with ant robots that use other real-time search methods, such as Learning Real-Time A* [24], Wagner's Value-Update Rule [38], and Thrun's Value-Update Rule [33]. Our results show

that all of these real-time search methods provide a good foundation for implementing ant robots [23]. Which real-time search method has slight advantages over the others depends on the performance measure but node counting is a good choice since ant robots that use node counting are very easy to implement.

Acknowledgments

References

1. F. Adler and D. Gordon. Information collection and spread by networks of patrolling ants. *The American Naturalist*, 140(3):373–400, 1992.
2. R. Aleliunas, R. Karp, R. Lipton, L. Lovasz, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 218–223, 1979.
3. R. Arkin. Motor-schema based mobile robot navigation. *International Journal of Robotics Research*, 8(4):92–112, 1989.
4. R. Arkin. *Behavior-Based Robotics*. MIT Press, 1998.
5. T. Balch and R. Arkin. Avoiding the past: A simple, but effective strategy for reactive navigation. In *International Conference on Robotics and Automation*, pages 678–685, 1993.
6. T. Balch, Z. Khan, and M. Veloso. Automatically tracking and analyzing the behavior of live insect colonies. In *Proceedings of the International Conference on Autonomous Agents*, pages 521–528, 2001.
7. T. Balch and A. Ram. Integrating robotics research with JavaBots. In *Proceedings of the AAAI Spring Symposium*, 1998.
8. M. Batalin and G. Sukhatme. Efficient exploration without localization. In *Proceedings of the International Conference on Robotics and Automation*, page (in press), 2003.
9. M. Blum and D. Kozen. On the power of the compass, or, why mazes are easier to search than graphs. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 132–142, 1978.
10. M. Blum and W. Sakoda. On the capability of finite automata in 2 and 3 dimensional space. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 147–161, 1977.
11. J. Borenstein, B. Everett, and L. Feng. *Navigating Mobile Robots: Systems and Techniques*. Peters, 1996.
12. V. Braitenberg. *Vehicles*. MIT Press, 1984.
13. H. Choset. Coverage for robotics - a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31:113–126, 2001.

14. X. Deng and A. Mirzaian. Competitive robot mapping with homogeneous markers. *IEEE Transactions on Robotics and Automation*, 12(4):532–542, 1996.

15. G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *IEEE Transactions on Robotics and Automation*, 7(6):859–865, 1991.

16. S. Hedberg. Robots cleaning up hazardous waste. *AI Expert*, pages 20–24, 5 1995.

17. W. Huang. Optimal line-sweep-based decompositions for coverage algorithms. In *Proceedings of the International Conference on Robotics and Automation*, pages 27–32, 2001.

18. T. Ishida. *Real-Time Search for Learning Autonomous Agents*. Kluwer Academic Publishers, 1997.

19. S. Koenig. Agent-centered search. *Artificial Intelligence Magazine*, 22(4):109–131, 2001.

20. S. Koenig, A. Blum, T. Ishida, and R. Korf, editors. *Proceedings of the AAAI-97 Workshop on On-Line Search*. AAAI Press, 1997. Available as AAAI Technical Report WS-97-10.

21. S. Koenig and R.G. Simmons. Xavier: A robot navigation architecture based on partially observable Markov decision process models. In D. Kortenkamp, R. Bonasso, and R. Murphy, editors, *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*, pages 91–122. MIT Press, 1998.

22. S. Koenig and B. Szymanski. Value-update rules for real-time search. In *Proceedings of the National Conference on Artificial Intelligence*, pages 718–724, 1999.

23. S. Koenig, B. Szymanski, and Y. Liu. Efficient and inefficient ant coverage methods. *Annals of Mathematics and Artificial Intelligence*, 31:41–76, 2001.

24. R. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211, 1990.

25. D. Lambrinos, R. Möller, R. Labhart, R. Pfeifer, and R. Wehner. A mobile robot employing insect strategies for navigation. *Robotics and Autonomous Systems*, 30:39–64, 2000.

26. D. Payton, M. Daily, R. Estowski, M. Howard, and C. Lee. Pheromone robotics. *Autonomous Robots*, 11(3):319–324, 2001.

27. A. Pirzadeh and W. Snyder. A unified solution to coverage and search in explored and unexplored terrains using indirect control. In *Proceedings of the International Conference on Robotics and Automation*, pages 2113–2119, 1990.

28. R. Russell. Heat trails as short-lived navigational markers for mobile robots. In *Proceedings of the International Conference on Robotics and Automation*, pages 3534–3539, 1997.

29. R. Russell. Ant trails – an example for robots to follow? In *Proceedings of the International Conference on Robotics and Automation*, pages 2698–2703, 1999.

30. R. Russell. *Odour Sensing for Mobile Robots*. World Scientific, 1999.

31. R. Russell, D. Thiel, and A. Mackay-Sim. Sensing odour trails for mobile robot navigation. In *Proceedings of the International Conference on Robotics and Automation*, pages 2672–2677, 1994.

32. R. Sharpe and B. Webb. Simulated and situated models of chemical trail following in ants. In *Proceedings of the International Conference on Simulation of Adaptive Behavior*, pages 195–204, 1998.

33. S. Thrun. Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, School of Computer Science, Carnegie Mellon University, Pittsburgh (Pennsylvania), 1992.

34. S. Thrun. Probabilistic algorithms in robotics. *Artificial Intelligence Magazine*, 21(4):93–109, 2000.

35. R. Vaughan, K. Stoy, G. Sukhatme, and M. Mataric. LOST: Localization-space trails for robot teams. *IEEE Transactions on Robotics and Automation*, 18(5):796–812, 2002.

36. I. Wagner and A. Bruckstein. Cooperative cleaners – a study in ant robotics. In A. Paulraj, V. Roychowdhury, and C. Schaper, editors, *Communications, Computation, Control, and Signal Processing: A Tribute to Thomas Kailath*, pages 289–308. Kluwer Academic Publishers, 1997.

37. I. Wagner, M. Lindenbaum, and A. Bruckstein. Smell as a computational resource – a lesson we can learn from the ant. In *Proceedings of the Israeli Symposium on Theory of Computing and Systems*, 1996.

38. I. Wagner, M. Lindenbaum, and A. Bruckstein. On-line graph searching by a smell-oriented vertex process. In S. Koenig, A. Blum, T. Ishida, and R. Korf, editors, *Proceedings of the AAAI Workshop on On-Line Search*, pages 122–125, 1997. Available as AAAI Technical Report WS-97-10.

39. I. Wagner, M. Lindenbaum, and A. Bruckstein. Efficiently searching a dynamic graph by a smell-oriented vertex process. *Annals of Mathematics and Artificial Intelligence*, 24:211–223, 1998.

40. I. Wagner, M. Lindenbaum, and A. Bruckstein. Distributed covering by ant-robots using evaporating traces. *IEEE Transactions on Robotics and Automation*, 15(5):918–933, 1999.

41. I. Wagner, M. Lindenbaum, and A. Bruckstein. MAC vs. PC: Determinism and randomness as complementary approaches to robotic exploration of continuous unknown domains. *International Journal of Robotics Research*, 19(1):12–31, 2000.

42. H. Yaguchi. Robots introduction to cleaning work in the east japan railway company. *Advanced Robotics*, 10(4):403–414, 1996.