A Dynamic Territorial Robotic System

Toby J. Richer
Advanced Computing Research Centre, University of South Australia
Mawson Lakes Boulevard, Mawson Lakes, SA, Australia
richer@cs.unisa.edu.au
Dan R. Corbett
Science Applications International Corporation
McLean, Virginia
Daniel.r.corbett@saic.com

Abstract

This paper outlines the implementation of an algorithm that divides a robot's topological map into a series of territories. It partitions this map using an algorithm based on clustering behavior in ant colonies. To perform this partitioning, the algorithm requires a specific topological map of the environment; the vertices are important features of the environment, and each vertex is connected to all other vertices within its line-of-sight. This algorithm is implemented through a robotic system that can traverse an unknown environment, generate a topological map suitable for use by the algorithm, and then partition the space by moving from vertex to vertex. The robotic system uses a behaviour-based system to navigate from vertex to vertex. The mapping system is topological, but uses some metric information on the distance between features and the shape of features to identify the feature currently visited by a robot. The robot and mapping systems have been tested on a real robot; it has implemented the territorial algorithm in a real environment in spite of uncertainty in sensing and odometry.

1 Introduction

A team of robots working together can perform some tasks faster than a single robot. However, a robot team becomes inefficient if robots interfere with each other's performance of the task [Mataric 1997]. Different ways of reducing these inefficiencies have been investigated. One method is territoriality – defining separate areas for each robot to perform the task. Dividing an area into territories is similar to graph partitioning, which is dividing a large, well-connected graph into a number of small clusters [Hendrickson and Leland 1995]. Graph partitioning algorithms aim to produce clusters of similar area and minimal connections between clusters.

This paper extends previous work in [Richer and Corbett 2004], which presented an emergent algorithm for partitioning a robot's environment into territories. This paper demonstrates that the dynamic territorial algorithm described in [Richer and Corbett 2004] can be implemented by a robot team in a previously unknown environment. Implementing this algorithm required the development of a new topological mapping system. This mapping system is based on existing topological mapping systems, but extended to connect each environmental feature with any other feature within its line of sight. The topological mapping system is tested on a real robot, and is shown to generate reliable topological maps with unreliable sensors and odometry. The novel contributions of this paper are the implementation of the territorial algorithm in a real-world environment, and the topological mapping system used in this implementation.

2 Territoriality in Robotics

Researchers have already implemented territorial multi-robot systems that forage for objects [Schneider-Fontan and Mataric 1998]. [Balch 1999] compared a number of approaches to multi-robot foraging, including a territorial approach. Both systems were shown to decrease inter-robot interference, but Balch's system performed worse overall. In both cases, the shape of the territories was set by the designer (though the system in [Schneider-Fontan and Mataric 1998] could adjust the size of the territories if a robot was lost).

A possible advantage of a territorial system is that it allows the map to be split between robots, provided each robot has strict boundaries on its movement. As a result, the robots do not need to store and update a map of the entire search area. Furthermore, territorial systems that repeatedly cover an area could do so more evenly than a non-territorial system. Consider an area that is surveyed by four robots. Without territories, areas could be visited four times in quick succession then left unvisited for a long time. If the area is divided into four territories, all areas are visited by a robot at more regular intervals. For tasks such as fire detection (where the time since the last visit must be minimized), territorial systems will be more effective. Finally, territories could provide a simple mechanism for robots to cooperate in detecting and pursuing intruders. Robots could move to territorial boundaries to block intruders coming from other territories.

For a territorial multi-robot system, we require an algorithm to divide the space quickly into near-equal areas for each robot. Territories of differing area can result in differing workloads for each robot. Minimizing contact between territories is desirable, as it reduces the amount of information required to store the territory boundaries. It also minimizes the paths by which an intruder could move between territories and so elude capture.

3 The Ant Clustering Algorithm

The ant clustering algorithm is based on observed behavior within the colonies of certain species of ant, where food, larvae, and dead ants are sorted into distinct piles inside the colony [Bonabeau, Theraulaz et al. 1998]. This occurs without any evidence of explicit collaboration or planning.

A mathematical model for ant clustering behavior was proposed in [Deneubourg, Goss et al. 1991]. Each ant is more likely to pick up a given resource in an area where that resource is relatively sparse. They are then inclined to drop this resource in an area where that resource is relatively common. Over time, a group of ants acting in this manner build up distinct piles of each resource.

This model has been used in robots to collect and cluster objects [Martinoli, Ijspeert et al. 1999]. It has also been adapted to perform Graph Coloring [Comellas and Ozon 1995] and Graph Partitioning [Kuntz, Layzell et al. 1997]. For Graph Partitioning, the vertices of the graph are projected on to a 2D space. A team of ant agents move through this 2-D space, picking up vertices that are highly dissimilar to their neighbors. In this case, the similarity of two vertices is defined as the number of other vertices that are connected to both of the vertices. These highly dissimilar vertices are then dropped in an area where they are highly similar to their neighbors. Over time, vertices with a high degree of connectivity are grouped into distinct clusters in the 2D space. This approach to graph partitioning has been applied to VLSI circuit design [Kuntz, Layzell et al. 1997] and visualization of genetic search spaces [Layzell 2002]. It has been adapted for document classification [Ramos and Merelo 2002]. The ant clustering model has been shown to be an effective approach to problems where multiple items must be arranged according to their relative similarity.

The qualities of an effective graph partitioning – near equal areas with minimal inter-area contact – are valuable qualities in a group of territories for surveillance robots, for the reasons described in the last paragraph of Section 2.

4 The Algorithm

This algorithm is a development of a graph partitioning algorithm developed in [Kuntz, Layzell et al. 1997]. This system has been altered to quickly create a number of pre-specified, explicit territories. Agents traverse a graph of the environment. Each vertex of the graph has a color (initially chosen at random) representing its territory. Agents move to vertices that are highly dissimilar to their territorial 'neighbors' (other vertices of the same color) and move them to a color where they are similar to their territorial neighbors.

The graph used for this algorithm is a topological map of the space. A number of robotic mapping and navigation systems already generate topological maps of a space; Section 6 discusses this further. The algorithm assumes that places where paths of motion end or begin are represented by vertices. Each vertex is connected to all other vertices that can be reached by a robot following a straight path, without encountering another vertex or an obstacle.

The relative similarity of two vertices is determined by the number of common neighbors. The dissimilarity function $d(v_i, v_j)$ is defined in Equation (4.1). In this case, V_i and V_j are the set of all vertices connected to v_i and v_j respectively, plus (in each case) v_i and v_j. The dissimilarity function is defined as the symmetric difference between V_i and V_j (the number of vertices present in the V_i and V_j 's union but not in their intersection) divided by the summed cardinality of V_i and V_j. The minimum value of $d(v_i, v_j)$ is 0. In this case, v_i and v_j are both connected to the same set of vertices (including each other). The maximum value of $d(v_i, v_j)$ is 1. In this case, there is no vertex v_k such that both v_i and v_j are connected to v_k.

$$d(v_1, v_2) = \frac{|V_1 \Delta V_2|}{|V_1| + |V_2|} \quad (4.1)$$

The similarity of a vertex is found by summing (1-$d(v_i, v_j)$) for all other vertices in its territory. This is adapted from the definition of similarity given in [Kuntz, Layzell et al. 1997]. The formula for this is given in Equation (4.2). The a value in (4.2) is a parameter that affects the degree of penalty for a vertex having no common neighbors with another vertex in its territory. $\Gamma(v)$ is a function that maps a vertex v to a particular color and territory.

$$f(v_i) = \begin{cases} \sum_{v_j;\Gamma(v_j)=\Gamma(v_i)} \left(1 - \frac{d(v_i, v_j)}{a}\right) & if\ f(v_i) > 0, \\ 0 & otherwise \end{cases} \quad (4.2)$$

The probability of a vertex being selected for change is given in Equation (4.3). The probability that a vertex is assigned to a particular territory is given in Equation (4.4). These are based on the original formulae used for picking up an object and dropping an object respectively [Deneubourg, Goss et al. 1991]. A range of values of k_p, a and k_d were tested. Setting k_p to 0.8, a to 0.8 and k_d to 0.3 produced stable territories quickly, without causing one territory to be overwhelmed by others.

$$p_p(v_i) = \left(\frac{f(v_i)}{k_p + f(v_i)}\right)^2 \quad (4.3)$$

$$p_d(v_i) = \left(\frac{k_d}{k_d + f(v_i)}\right)^2 \quad (4.4)$$

Pseudo code for the algorithm is given in Figure 1. The initial positions of agents are randomly selected such that no two agents start on connected vertices, if sufficient space exists. For each timestep, each agent picks a new vertex to move to using a weighted roulette wheel selection. The weighting of each vertex equals p_p plus a base probability value. The new color for this vertex is then also selected using a weighted roulette wheel selection. In this case, the weighting of each color is given by p_d (calculated assuming the vertex is that color) plus a base probability value.

To make sure that territories are not eliminated from the graph with this system, all ants have access to a shared set of vertices that are not connected with the graph. The shared set initially contains two vertices of each territory. The agent can only change the territory of a vertex if it can swap this territory with a vertex of the desired territory in this shared set.

```
/*Initialization*/
FOR all vertices in graph
  Choose a territory for this vertex
END FOR
FOR all ants
  Choose starting position
END FOR
/*Main procedure*/
FOR all timesteps
   FOR all ants
      /*Choose the next vertex to move to*/
      FOR all adjacent vertices n
         Probability(n) = $p_p$(n) + $p_{base}$
      END FOR
       Choose best vertex using weighted
       roulette wheel selection
       (each vertices' weight equal to
       Probability(n))
       IF (best vertex is unoccupied) THEN
       move to best vertex
       ELSE move to adjacent vertex
       END IF
       /* Now select new color */
       FOR all colors c
         Probability(c) = $p_d$(n) (assuming
         color = c) + $p_{base}$
      END FOR
      /* Select a color randomly */
       Choose best color using weighted
       roulette wheel selection
       (each color's weight equal to
       Probability(c))
      /* Decide whether to change color */
       Calculate $p_d$(n) with current color
       Choose a real number R between 0and1
       IF (R < probability of change) AND
          (best color is-element-of color
        list) THEN
              Change color of this vertex to
              best color
              Replace one instance of best
              color in color list with
              previous color of this vertex
      END IF
   END FOR
END FOR
```

Figure 1

5 Simulated Testing

To determine if the algorithm could efficiently divide an area into territories, it was tested on several hand-designed environments. The environments used to test the partitioning algorithm were four connected rooms, as shown in Figure 2, a series of rooms connected by a corridor (an office environment), shown in Figure 3, and a single room with walls and obstacles, as shown in Figure 4.

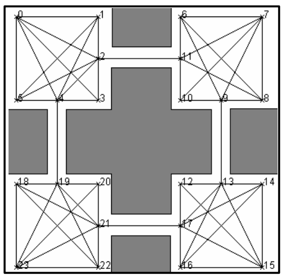


Figure 2

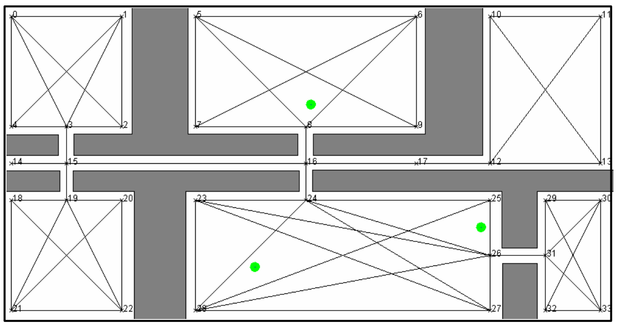


Figure 3

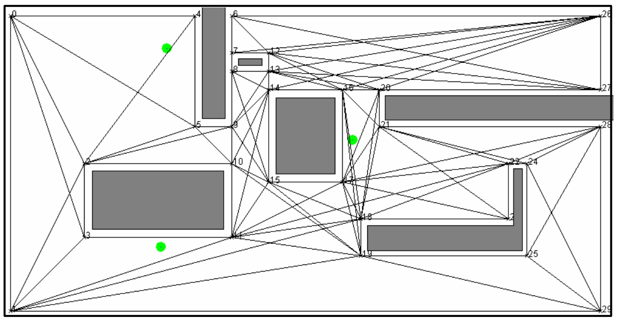


Figure 4

A graph representation of each of these environments was constructed according to the rules stated in Section 4. In each case, 10 trials were run. Equal numbers of agents and territories were used. The territories and iterations for each environment were:

- Four rooms: 2, 3, and 4 territories. 400 timesteps.
- Office: 3, 4, 5 and 6 territories. 1000 timesteps.
- Obstacles: 2, 3, and 4 territories. 1000 timesteps.

To measure the effectiveness of the algorithm for partitioning the graph into territories, the local distribution of dissimilarity values was measured. The local dissimilarity, assuming a set of vertices *V*, is given in Equation (5. 1):

$$\frac{1}{K_t}\sum_{v_i \in V}\sum_{\Gamma(v_i)=\Gamma(v_j)} d(v_i, v_j) \quad (5.1)$$

K_t is the total number of dissimilarity calculations. This measure, adapted from [Kuntz, Layzell et al. 1997], calculates the level of difference between all pairs of vertices that share the same territory, then averages this value over all calculations. This measure was averaged over all runs for a particular number of territories in a given environment.

These tests are described in full in [Richer and Corbett 2004]. They showed that this algorithm can detect clusters (groups of vertices that are highly connected) and use these clusters to partition a graph into a number of territories. It can do this using a very small number of iterations. If the number of clusters is divisible by the number of territories, the territories formed are highly stable. If not, the boundaries of the territories are more fluid. Vertices that are not highly connected (such as the corridor in the office environment) remain fluid, as they are not connected sufficiently to any one territory to resist being changed to another territory. As long as clusters are present, large contiguous blocks of territory are formed. Tests in the obstacle-strewn environment (a three territory division of this environment is shown in Figure 5) show that the algorithm can effectively locate clusters that are connected by several edges.

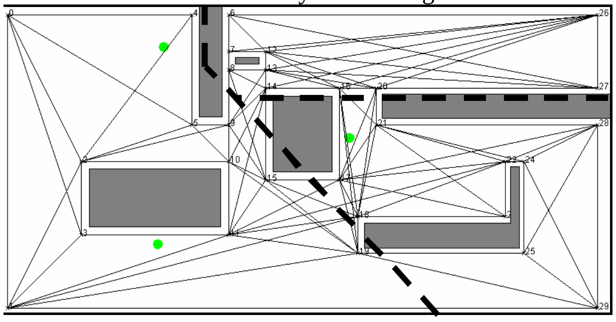


Figure 5

6 Mapping System

The intended application of this algorithm is in multi-robot systems that must repeatedly cover the same space, such as multi-robot surveillance systems. To implement this algorithm, the space must be represented as a well-connected topological map as described in Section 4.

A range of topological mapping systems have been developed for robots. Topological maps require less memory and processing than metric maps, as they represent the environment using minimal information.

The first topological mapping systems worked by identifying specific obstacles from sensor readings, then linking adjacent obstacles [Mataric 1992]. These systems have been extended by introducing metric or visual information to identify particular obstacles [Goel, Roumeliotis et al. 1999; Zanichelli 1999; Yairi, Hirama et al. 2001], or making assumptions about the environment [Dedeoglu, Mataric et al. 1999] to aid in identification. Other systems generate graphs from the boundaries between obstacles – points where a robot can choose to move in several directions. The points are identified through changes in the robot's sensor values. Two examples of such systems are the Spatial Semantic Hierarchy [Kuipers 1996; Lee 1996; Kuipers 2000] and the Generalized Voronoi Graph [Choset and Burdick 1995; Choset and Burdick 1995; Choset, Konukseven et al. 1996; Choset and Nagatani 2001].

Generally, these systems do not connect all features of the environment that the robot can travel between by moving through open space. There are some exceptions [Yairi, Hirama et al. 2001]. The territorial algorithm described in this paper requires all these connections to be present. The next two sections describe a topological mapping system, based on previous work in topological mapping, that can represent an environment in a form usable by the algorithm, and a robot system to create this map. The map produced consists of a set of features, representing important parts of the environment, connected by links that represent straight line paths of motion between these features.

6.1 Features

The features are defined by a rough position estimate, in x and y coordinates, and a set of paths. The paths describe the valid directions for a robot to move from this feature. They aim to represent the general shape of the feature. Each path is defined by a heading, and two boolean variables indicating if there is a wall to the left or right of this path.

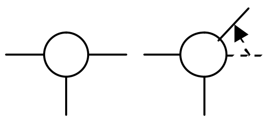
When a feature is first detected by the robot, the definition of the feature is generated from the robot's sensor and position data. The sensor readings are divided into categories of *short*, *medium*, and *long* readings according to length. A path exists in the direction of a reading if the reading is in the *long* category, and has a wall to its left or right. If, moving left or right from a sensor reading, a *short* reading is reached before a *long* reading, then there is a wall in this direction. If a *long* reading is reached first, there is no wall. The heading of a feature is determined by the sensor position and the heading of the robot. The presence of a wall on the left or right is tested using the method described above and recorded as part of the path's definition.

6.2 Comparison of Features

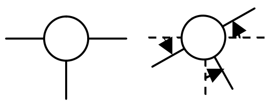
The system models the movement of the robot between features as a Partially Observable Markov Decision Process (POMDP). This technique was developed in [Zanichelli 1999]. The map is modeled as a series of states; each state corresponds with a feature. The robot's movement is modeled as transitions between these states; the position of the robot between states is not modelled.

To determine if a new feature observed by the robot is part of the current map, the new feature is compared with previous features using the POMDP. The relative locations are compared, and the relative shape of the features is compared using two variables, defined as the *orientation error* and the *landmark error*. The landmark error represents the difference between single Paths in the feature; how far individual paths must be moved for the features to then match under rotation. The orientation error represents the total rotation required; both the landmark error, then the overall rotation of the features to match. The difference is illustrated in Figure 6.

To determine the landmark error and orientation error between two features, all possible match sets are compared – all mappings of the paths in one feature to corresponding paths in the other feature. For two features of *n* paths, there are *n* possible match sets; path *n* of the first feature is matched to path 1 of the second feature, and the remaining paths are then matched in order (path $n+1$ to path 2, path $n+2$ to path 3, etc). In each case, the booleans indicating the presence of walls for each path are compared between features. If the matched paths from two features have different walls, that match set is discarded. The landmark and orientation error is calculated for the match sets where the walls match. The match set with the smallest landmark error and orientation error is used for the overall comparison, as described below; the lowest orientation error is used if two landmark error values are very similar.



Landmark Error



Orientation Error

Figure 6

Calculation of Uncertainty

The topological mapping system does not take previous movements of the robot into account, except when a match occurs and the internal position estimate of the robot is changed (see below). The belief value that a particular state (i.e. feature) *B(s)* corresponds with an observation p is:

$$\forall s \in S \quad B(s) \leftarrow P(p \mid s) \quad (6.1)$$

The value of B(s) is not normalized along s, as the probability that the observation p corresponds to a new state, not currently in S, must be considered. $P(p|s)$ represents the probability that the system receives a set of sensor values p when at feature s. The set of sensor values p contains the landmark error l_p, the position error p_p, the orientation error o_p, and the number of paths n_p. The total probability of a set of sensor values appearing when the robot is at a feature can be described by the conditioned probability distribution:

$$P(p \mid s) = P(l_p, o_p, p_p, n_p) \quad (6.2)$$

From the distribution independence hypothesis, it is assumed that:

$$P(p \mid s) = P(l_p \mid l_s) P(o_p \mid o_s) P(p_p \mid p_s) P(n_p \mid n_s) \quad (6.3)$$

This assumption is not entirely correct, but significantly simplifies the calculation of certainty. The probability of being at a feature with n_s Paths, given an observation of n_p Paths, is assumed to be zero if the number of paths is different in each case, i.e.

$$P(n_p \mid n_s) = 1 \; if \; n_p = n_s$$

$$P(n_p \mid n_s) = 0 \; if \; n_p \neq n_s \quad (6.4)$$

$P(l_p \mid l_s)$, $P(o_p \mid o_s)$, and $P(p_p \mid p_s)$ are assumed to have a Gaussian distribution. The standard deviation of landmark error is fixed, the standard deviation of orientation error is proportional to the total angle rotated by the robot, and the standard deviation of position error is proportional to the total distance travelled by the robot.

Feature Comparison

To compare the current position of the robot to the existing features on the map, the belief that the robot is at each feature, given the current surroundings of the robot, is calculated as described in the previous Sections.

If the total uncertainty for all current features is below a specified threshold, then the feature at the current position of the robot is considered to be a new feature, and is added to the map. If the total uncertainty for any of the current features is above this threshold, then the feature observed by the robot is assumed to be identical to the feature with the highest $P(p|s)$, as given in Formula 6.3.

Update of Previous Positions

As the robot travels further, its perceived position will drift further and further from its actual position. To counteract this problem, when the robot detects a new feature and matches this to an existing feature on the map, the robot adjusts its internal position to the position of the feature on the map. Currently, the robot updates its internal position to match the position of the feature in the map every time the feature perceived by the robot matches a pre-existing feature on the map. The heading of the robot is not updated to match the map, as the robot compass (described in Section 8) should not drift significantly during an experiment.

Links

According to the description of the graph given in Section 3, any features that are visible from one another should be linked. The behaviour of the robots (as described in Section 7) is designed such that any two features visited in succession fulfil this requirement. Because of this, the process for linking features is simple; any two features visited in succession are linked.

6.3 Summary

This section has described a topological mapping system that generates a representation of an environment that can be partitioned into territories using the Territorial algorithm described in Section 4. It represents the environment as a series of features, representing distinctive parts of the environment. These features are described by both position and general shape, in a manner that is robust to sensor error. Features are compared using a probabilistic model. The next section describes a robot control system that is designed to traverse an unknown environment and generate a map of this form.

7 Robotic System

The robot control system is designed to generate a map of an unknown environment and divide the map into partitions according to the territorial algorithm. The robot completes this objective by performing three tasks in sequence. Firstly, the robot develops a map of the external boundary of the environment, by following the left wall and transmitting the details of any features it encounters. Secondly, the robot finds all the links between these features. It alternates following the left wall, and moving across open space. When it gets close to a new wall, it moves to the wall then follows it to find new features. Finally, the robot implements the territorial algorithm by moving to features and assigning them to territories.

The robot control system is a behaviour-based system as initially developed by [Brooks 1985] and further developed by several researchers, including [Mataric 1995] and [Arkin and Balch 1997]. The robot is controlled through a series of behaviours, each designed to fulfil a particular short-term goal; moving along a wall, turning to face a wall, moving through open space, and others. The robot switches between behaviours through a finite-state-automaton, or FSA. Figure 7 is a diagram of the FSA. Each of the three tasks required to map and partition the area uses a subset of the behaviours in sequence. The behaviours are described individually overleaf.

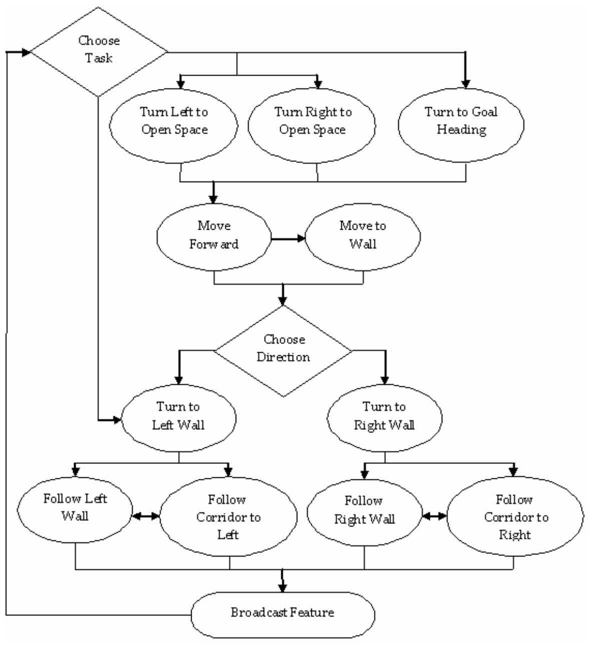


Figure 7

7.1 Behaviours

Choosing the Task to Perform

The *choose-state* behaviour does not move the robot, but determines which sequence of other behaviours the robot will use. If the robot is performing the left-hand traversal of the environment, the robot immediately switches to the *turn-left-to-wall* state. If the robot is filling in links, the robot alternates behaviour. It will move through open space, turning to face open space using either the *turn-left-to-open-space* or *turn-right-to-open-space* behaviours. Once it detects a feature using this method, it uses the *turn-left-to-wall* state to find and follow a wall to the next feature. If the robot is implementing the territorial algorithm, and is supplied with a goal heading, it switches to the *turn-to-goal-heading* state.

Turning to Open Space

Turn-left-to-open-space and *turn-right-to-open-space* turns the robot left or right respectively until it faces open space. *Turn-to-goal-heading* turns the robot until it faces the goal heading supplied by the base station. The robot then either moves into a wall-following behaviour, corridor-following behaviour, or moves forward into empty space, depending on whether any walls are observed by the robot.

Moving Forward to Open Space

The *move-forward* behaviour is used by the robot to cover large areas of open space. It moves the robot forward until an object is detected in front. Unless the object is very close to the robot, the object is ignored until the robot has travelled a significant distance from its initial position. If a very close object is discovered, the robot enters the *choose-direction* behaviour. Otherwise, the robot moves closer to the object using the *move-to-wall* behaviour.

Moving to a Wall

The *move-to-wall* behaviour improves the rate of successful feature detection by moving the robot to a wall as it nears a feature, and thus preventing the robot from moving past the feature. It is activated by the *move-forward* behaviour, when the front of the robot detects a wall within 30cm. The behaviour checks all sensors, to determine which is the closest. If the closest sensor is the front sensor, the robot moves forward, otherwise the robot turns towards the closest sensor. The behaviour ends when the closest sensor is the front sensor and the sensor reading is less than 10cm. The robot then chooses which direction to follow the wall, through the *choose-direction* behaviour.

Choosing a Direction to Follow

The *choose-direction* behaviour does not move the robot; it determines the direction in which to follow the wall. The method for choosing the direction varies depending on the current goal heading. If the robot does not have a goal heading, it choose the direction closest to the current direction of the robot. The front-left and front-right sensors are compared, and the robot follows the wall on the side of the closer sensor reading. If the robot has a goal heading, this heading is compared with the current heading of the robot. If the current heading is left or right of the goal heading, the robot follows the wall on the left or right respectively. The *turn-left-to-wall* or *turn-right-to-wall* behaviours are then used to align the robot with the wall.

Turning to Align With a Wall

The purpose of the *turn-left-to-wall* and *turn-right-to-wall* behaviours are to align the robot with a nearby wall so that the wall is parallel to the robot and on the left, or right, respectively. The robot uses the front sensor, combined with the front-left or front-right sensor respectively, to determine if it is aligned with the wall. The robot turns left for the *turn-left-to-wall* behaviour, or right for the *turn-right-to-wall* behaviour, until the front-left or front-right sensor detects a wall while the front sensor is detecting open space. At this point, the robot enters a wall-following behaviour if no wall is detected on the other side of the robot, or a corridor-following behaviour if walls are detected on both sides.

Following a Wall

The purpose of the *follow-left-wall* and *follow-right-wall* behaviours is to move the robot along a wall until the robot reaches a feature. The robot maintains a fixed separation from the wall using either the front-left ultrasonic sensor, for the *follow-left-wall* behaviour, or the front-right sensor for the *follow-right-wall* behaviour.

The robot determines if it is too far away or too close to the wall by comparing this sensor reading with a fixed value. It then turns towards or away from the wall to compensate. These behaviours stop when the robot detects either a new wall in front, a new wall on the opposite side to the wall being followed, or an end to the wall being followed. Provided the robot has travelled sufficient distance from the last feature, the robot broadcasts a new feature.

Following a Corridor

The *follow-corridor-on-left* and *follow-corridor-on-right* behaviours are extensions of the *follow-left-wall* and *follow-right-wall* behaviours to navigate the robot through a corridor. The robot follows the corridor by maintaining a fixed separation from one wall, as with *follow-left-wall* and *follow-right-wall*. This behaviour ends if a new wall is detected in front, or if open space is discovered on either side. The robot then broadcasts a new feature.

7.2 Implementing the Mapping System

This robot control system was implemented on the robot described in Section 8 This robot does not have the processing capability to perform mapping. The mapping system is implemented in a separate computer that communicates to the robot through a wireless link. When the robot detects a new feature, it is sent back to the computer, where it is compared to the map according to the process described in Section 6. The computer sends back the position of this feature, if it matches a previous feature in the map, and a byte giving information about the robot's new task. If the robot has to move to a particular feature, as required when implementing the territorial algorithm, the byte gives the heading of the feature, otherwise the byte gives a fixed value indicating which task the robot should perform next.

7.3 Implementing the Territorial Algorithm

Once the full map of the environment is generated, the territorial algorithm can be implemented both in software and hardware. As in Section 5, the graph can be partitioned into territories by software agents. Alternatively, the graph can be partitioned into territories physically, through the movements of the robot. This robot system is designed to partition the map, in a similar way to an ant agent, following the construction of the map. In this case, the robot does not wander the map, but moves towards a specific feature designated by a goal heading. The behaviours used for this task are described in the previous subsection.

When performing the territorial algorithm, if the robot moves to a new position, the new position is compared to the current map. If there is a match with a previous feature, the feature visited by the robot is passed to the territorial algorithm. The territorial algorithm updates the territories, and sends a new goal feature to the robot, based on the territorial algorithm. The robot attempts to move to this feature. Whether or not it reaches this particular feature, it sends the next feature it visits to the territorial algorithm, and the process continues. Over time, the robot should implement the territorial algorithm, and divide the environment into a pre-specified set of territories.

8 Physical Testing

The system's ability to map an area and divide it into territories was tested in three stages, corresponding to the three subtasks described in Section 7. The robot performed a left-hand traversal of the environment, fills in all links by crossing open space, then divides the area into territories.

8.1 Robot used for implementation

The robot Remorseful (see Figure 8) is used to test the system. Remorseful is a DescartesBot from AliveRobotics. Its processor is a Basic Stamp 2SX. It senses obstacles through a ring of eight ultrasonic sensors, and estimates its own position through optical wheel encoders and a digital compass. The robot moves using two differentially-driven wheels. The behaviour of the robot is implemented on the Basic Stamp 2SX, while the mapping system is implemented on a separate PC that communicates with the Robot through a radio transceiver.

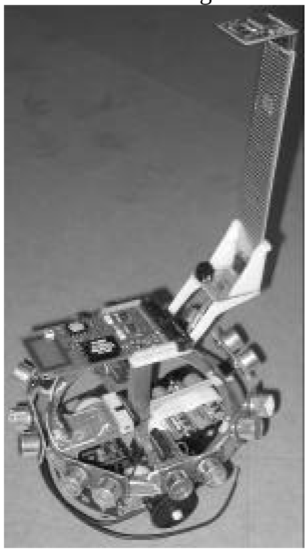


Figure 8

8.2 Testing Environment

The test of the left-hand traversal was performed in a model of a two-room environment, as shown in Figure 9. This model was used for most robot testing. It has a wooden floor, measuring 1.2 metres by 2.4 metres, and chipboard and cardboard walls. For these tests, the walls formed the shape of two rectangular rooms, separated by a short corridor.

8.3 First Test

To test the first sub-task, the robot is placed next to a wall, and activated. The robot turns until aligned with the left wall, then follows the left wall around the environment, broadcasting features when it reaches them. As it reaches each feature, it will broadcast the characteristics of the feature to the base station, then continue the traversal. These features will be added to the map if new. If they match a previous feature, a new link will be added between the features. The robot runs for 3 circuits of the environment. The corridor in the environment is sufficiently narrow that it should be treated as a corridor, rather than two separate walls, by the robot.

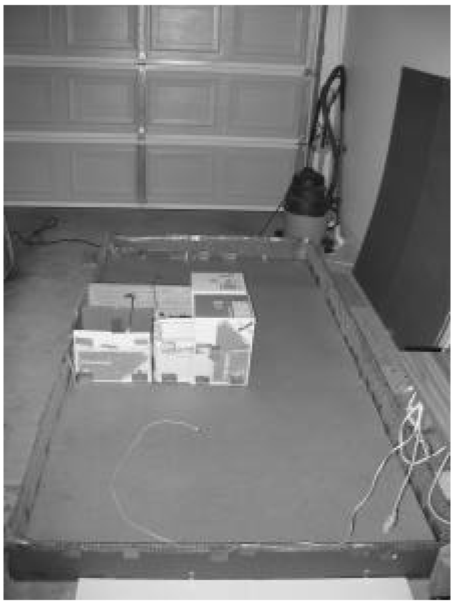


Figure 9

The map produced by the system is shown in Figure 10. The arena is shown, for comparison, in Figure 9. The robot has produced a map that accurately reflects the environment, and has matched all previously-visited features to the correct feature that has already been discovered. This shows that the feature matching system used here is effective. In generating the map, the robot has performed an effective left-hand traversal; it remains close to each wall, while not colliding with it. This shows that the behaviours used by the robot to traverse the area are reliable, both in avoiding obstacles and correctly detecting features.

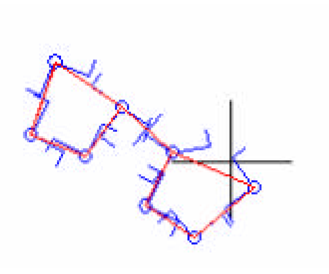


Figure 10

8.4 Second Test

The second test is performed using the map from the first test. It tests the ability of the robot to perform the second sub-task; to find all straight-line paths between features, and add these to the map. It also tests the ability of the mapping system to match features correctly when the features are approached from several directions.

The robot is placed in the same position as for the first test, and activated. The robot should alternate between following the wall and moving across open space. Over time, its moves through open space should detect any links between features that were not detected by the left-hand traversal.

The map produced by the robot is shown in Figure 11. As required, the robot has added all possible straight-line links to the existing map of features. The resulting map shows all valid connections between features. This shows that the robot mapping system developed in this thesis can effectively traverse and generate an unknown environment and generate a topological map. The system develops a topological map with sufficient connectivity for use with the dynamic territorial algorithm. The robustness of the mapping system is shown by the lack of extra features generated as a result of the wandering, and the robustness of the robot system is shown by its ability to stop at the same feature points from several different directions.

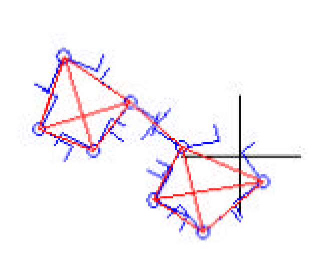


Figure 11

8.5 Third Test

The third test tests the implementation of the territorial algorithm. Using the map produced by the second test, the robot implements the territorial algorithm as described in Section 3.

The area was initially split randomly into two territories. The territorial decomposition of the map when the robot had run the territorial algorithm for 41 transitions between features is shown in Figure 12. The environment has been organised into two territories with minimum contact through the movements of the robot. This shows that the territorial algorithm can be implemented physically.

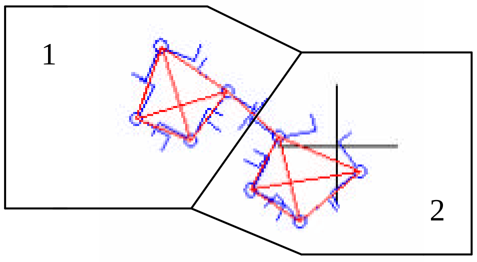


Figure 12

9 Conclusions

This paper presents general arguments for the use of territories in some multi-robot systems, and describes the algorithm, first proposed in [Richer and Corbett 2004] for partitioning a graph into a set number of territories based on the relative connectedness of vertices. The paper then presents a novel robot mapping system that can traverse an environment in a two-stage process and produce a map of the environment that can be partitioned by the algorithm. The mapping system is tested on a real-world environment, and is shown to generate a map that can then be used to physically implement the territorial algorithm. It is intended to use this territorial algorithm in several multi-robot tasks, in particular multi-robot surveillance.

References

Arkin, R. C. and T. Balch (1997). "AuRA: Principles and Practice in Review." Journal of Experimental and Theoretical Artificial Intelligence **9**(1997): 175-189.

Balch, T. (1999). The Impact of Diversity on Performance in Multi-Robot Foraging. Agents '99, Seattle, WA.

Bonabeau, E., G. Theraulaz, et al. (1998). "The phase-ordering kinetics of cemetery organization in ants." Physical Review E **57**: 4568-4571.

Brooks, R. A. (1985). A Robust Layered Control System for a Mobile Robot. Cambridge, MA, Massachusetts Institute of Technology.

Choset, H. and J. Burdick (1995). Sensor based planning, Part I: the generalized Voronoi graph. Proceedings of the 1995 IEEE International Conference on Robotics and Automation. Part 2 (of 3), May 21-27 1995, Nagoya, Jpn, IEEE, Piscataway, NJ, USA.

Choset, H. and J. Burdick (1995). Sensor based planning, Part II: incremental construction of the generalized Voronoi graph. Proceedings of the 1995 IEEE International Conference on Robotics and Automation. Part 2 (of 3), May 21-27 1995, Nagoya, Jpn, IEEE, Piscataway, NJ, USA.

Choset, H., I. Konukseven, et al. (1996). Mobile robot navigation: Issues in implementing the generalized Voronoi graph in the plane. Proceedings of the 1996 IEEE/SICE/RSJ International Conference on Multisensor Fusio n and Integration for Intelligent Systems, Dec 8-11 1996, Washington, DC, USA, IEEE, Piscataway, NJ, USA.

Choset, H. and K. Nagatani (2001). "Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization." IEEE Transactions on Robotics & Automation **17**(2): 125-37.

Comellas, F. and J. Ozon (1995). "Graph Coloring Algorithms for Assignment Problems in Radio Networks." Applications of Neural Networks to Telecommunications **2**.

Dedeoglu, G., M. J. Mataric, et al. (1999). Incremental, on-line topological map building with a mobile robot. Mobile Robots XIV - SPIE 99, Boston, MA.

Deneubourg, L., S. Goss, et al. (1991). The Dynamics of Collective Sorting: Robot-like Ants and Ant-like Robots. the First International Conference on Simulation of Adaptive Behavior: From Animals to Animats, Cambridge, MA, MIT Press/Bradford Books.

Goel, P., S. I. Roumeliotis, et al. (1999). Robust localization using relative and absolute position estimates. Intelligent Robots and Systems, 1999. IROS '99. Proceedings. 1999 IEEE/RSJ International Conference on.

Hendrickson, B. and R. Leland (1995). A multilevel algorithm for partitioning graphs. the 1995 International Conference on Supercomputing, San Diego, California, ACM.

Kuipers, B. (1996). A hierarchy of qualitative representations for space. Proceedings, Qualitative Reasoning: The Tenth International Workshop, 21-24 May 1996, Fallen Leaf Lake, CA, USA, AAAI Press.

Kuipers, B. (2000). "The spatial semantic hierarchy." Artificial Intelligence **119**(1-2): 191-233.

Kuntz, P., P. Layzell, et al. (1997). An Ant Clustering Algorithm Applied to Partitioning in VLSI Technology. The 4th European Conference on Artificial Life (ECAL97).

Layzell, P. (2002). Visualising Evolutionary Pathways in Real-World Search Spaces. 8th Int. Conf. on Artificial Life, Visualisation Workshop, Sydney.

Lee, W. Y. (1996). Spatial Semantic Hierarchy for a Physical Mobile Robot. Department of Computer Sciences. Austin, University of Texas at Austin.

Martinoli, A., A. J . Ijspeert, et al. (1999). "Understanding Collective Aggregation Mechanisms: From Probabilistic Modelling to Experiments with Real Robots." Robotics & Autonomous Systems(29): 51-63.

Mataric, M. J. (1992). "Integration of Representation Into Goal-Driven Behavior-Based Robots." IEEE Transactions on Robotics & Automation **8**(3): 304-312.

Mataric, M. J. (1995). "Designing and understanding adaptive group behavior." Adaptive Behavior **4**(1): 51-80.

Mataric, M. J. (1997). "Learning social behavior." Robotics & Autonomous Systems **20**(2-4): 191-204.

Ramos, V. and J. J. Merelo (2002). Self-Organized Stigmergic Document Maps: Environment as a Mechanism for Context Learning. MAEB 2002 - 1st Spanish Conference on Evolutionary and Bio-Inspired Algorithms, Merida, Spain.

Richer, T. J. and D. R. Corbett (2004). Using an Ant Clustering Algorithm to Create Partitions for a Territorial Robotic System. The 8th Conference on Intelligent Autonomous Systems, Amsterdam, IOS Press.

Schneider-Fontan, M. and M. J. Mataric (1998). "Territorial multi-robot task division." IEEE Transactions on Robotics & Automation **14**(5): 815-22.

Yairi, T., K. Hirama, et al. (2001). Fast and simple topological map construction based on cooccurrence frequency of landmark observation. Proceedings of RSJ/IEEE International Conference on Intelligent Robots and Systems, 29 Oct.-3 Nov. 2001, Maui, HI, USA, IEEE.

Zanichelli, F. (1999). Topological Maps and Robust Localization for Autonomous Navigation. IJCAI workshop on Adaptive spatial representations of dynamic environments.