This is the published version:


Available from Deakin Research Online:

http://hdl.handle.net/10536/DRO/DU:30044772

Reproduced with the kind permissions of the copyright owner.

Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Copyright: 1996, IEEE
Abstract

For a mobile robot to perform some sort of useful function it usually must have some sort of global understanding of its environment. This is usually expressed in the form of map.

Through real-world experiments, using a mobile robot inspired by insect visual guidance, we present results showing the performance of a mobile robot in recognising a previously encountered corridor environment and discriminating between various corridors. This is achieved through the building and refinement of maps based on the observation of simple landmarks en route.

1 Introduction

For a mobile robot to do something useful usually implies knowing, to some degree, its location. Being able to move from location to location, purposefully, requires some knowledge of the world and one’s place within it.

The basic function of our mobile robot is to autonomously navigate along corridor-type environments. We, therefore, have explored the possibility of building some form of representation to adequately describe specific sections of corridor that will allow further recognition and refinement at a later date.

Given the inherently structured nature of corridors, corridor recognition could be quite efficient in an environment such as a complex complex. Assuming corridor-type structures connect various parts of the complex, it could be easier to recognise a connecting corridor, and hence one’s location, than to try to recognise some unstructured area at the end of a corridor. However, due to the structured nature of corridors, it is also quite difficult to try to recognise a specific piece of corridor by shape alone. Some other discriminating factor must be included. Simple landmarks seem very useful for this purpose.

2 Background

Inspired by the way flying insects navigate through the real world, with seemingly little effort, we have equipped a mobile robot with the ability to navigate along corridor-type environments using only the apparent motion observed, while in motion [4]. The inspiration for using apparent motion information to maintain a centring posture within a corridor came from the visual-guidance behaviour of honeybees [3].

Dead reckoning is used by several insects to aid in navigation. The desert ants of the genus *Cataglyphis*, for instance, have been shown [5] to use a vector navigation strategy to keep a bearing and distance of their nest, when out foraging, to allow an efficient return when finished.

Landmarks themselves are also used by insects. Many insects, especially foraging insects such as bees [1, 2], wasps and ants [6], exploit landmark cues to pilot their way back home.

3 Robot Setup and Optics

The vision for the mobile robot is provided by a miniature CCD video camera, which is placed looking upwards at a mirror assembly (figure 1). The mirror
assembly directs two lateral and two straight-ahead views onto the imaging plane of the camera.

![Mobile robot](image1.jpg)

**Figure 1: Mobile robot**

### 4 Map Building

In order to assemble a set of landmarks and put them into a useful form, the robot endeavours to build and refine a map. The map consists of a consecutive list of dead-reckoned landmarks as observed en route. Each landmark node records the position at which the landmark, which in this case is a very dark or very light region of the environment, was observed.

Each landmark node also records the direction in which the dark/light region was observed. This is done by recording the current robot orientation together with the side of the robot on which the landmark was observed. The cumulative odometry is recorded, allowing analysis of distances travelled between specific landmarks. We have also included a “reliability” counter with each landmark node. This count serves as an estimate on the importance of the landmark when matching and refining a map.

Thus each landmark node consists of (i) the \((x, y)\) position, (ii) the side of the robot the landmark was observed from, (iii) the cumulative robot orientation relative to the starting position, (iv) odometry, and (v) the reliability count for the node.

Thus a typical node \(n_i\) is given as:

\[
   m_i = \text{node}(x, y, \text{side}, \text{orientation}, s, \text{count})
\]

A map \(M\) consists of a sequence of nodes \(m_i\):

\[
   M = \{m_1, m_2, \ldots, m_n\}
\]

#### 4.1 Matching

For the purpose of matching the current map (pattern of landmark nodes observed en route) to previous “learned” maps, we have implemented a depth-first tree searching algorithm. Utilising a divide-and-conquer strategy we strive to find the best match between pairs of landmark nodes. The matching can therefore be done incrementally, as landmarks are encountered, making it more amenable to a real-time implementation.

Specifically the degree of match between two pairs of landmark nodes is calculated as a “probability”, using the differences in displacement between the landmark nodes and the relative angles of the landmarks, assuming a gaussian distribution. The algorithm \(\text{node\_pair\_prob}\) is shown below.

\[
   \text{Learned Map: } \{m_1, m_2, \ldots, m_n\}
   \text{Traversed Pattern: } \{p_1, p_2, \ldots, p_m\}
\]

\[
   \text{node\_pair\_prob}(m_1, m_2, p_1, p_2)
\]

node pairs are first translated so that both \(m_1\) and \(p_1\) are at origin \((0, 0)\)

\[
   D = (x, y) \text{ positional displacement between } m_2 \text{ and } p_2
\]

\[
   A_m = m_2\text{orientation} - m_1\text{orientation}
\]

\[
   A_p = p_2\text{orientation} - p_1\text{orientation}
\]

if \((m_1\text{side} \neq m_2\text{side})\) then \(A_m = A_m + 180\)

if \((p_1\text{side} \neq p_2\text{side})\) then \(A_p = A_p + 180\)

\[
   A = A_m - A_p
\]

if \(|A| > 180\) then \\
\pm \text{ multiples of 360, until } |A| \leq 180

\[
   P_1 = \text{probability of match based on } D,
   \text{ assuming a gaussian error distribution}
   (\mu = 0, \sigma = 20cm)
\]

\[
   P_2 = \text{probability of match based on } A,
   \text{ assuming a gaussian error distribution}
   (\mu = 0, \sigma = 30°)
\]

\[
   \text{node\_pair\_prob} = P_1 \times P_2
\]

The matching algorithm is essentially a recursive process. At every level of the recursive matching procedure (each level of the search tree) there are several matching possibilities which are examined:

(i) identical \((M_1)\) - A one-to-one correspondence between nodes.

(ii) transposed landmarks \((M_2)\) - Between two valid landmark nodes there are two landmarks which are transposed with one another. If two landmarks are observed very close to each other on opposite sides of the corridor then, depending on the particular position and motion of the robot, either landmark can be observed first.

(iii) combination of missing and additional landmarks \((M_3)\) - Between two valid landmark nodes there are assumed to be \(k\) missing landmarks as well as \(j\) additional landmarks; \(0 \leq k \leq C\) and \(0 \leq j \leq
match is basically defined by the “average” match of the solution. If the average match is greater than some threshold (0.5 in our case) then the solution is considered to be a valid match. However, when an incremental matching algorithm is used here, one must also take into account the number of landmark nodes. Clearly, the fewer landmarks actually observed, the less likely the solution (currently attained) is reliably valid.

Specifically, the average match for a solution is calculated as:

$$average\ match = \frac{best\ match}{solution\ size - 1 + penalty}$$

$$penalty = \sum_{i=1}^{m} \frac{m_{count} \cdot \#missing}{\#count} + \sum_{i=1}^{0.2}$$

where solution size is the number of pairs of matched nodes in the solution; $M$ is the set of map nodes not mentioned in the solution; $\#missing$ is the size of $M$; $\#added$ is the number of pattern nodes which are missing from within the solution; tcount is again the sum of all landmark-node counts within the learned map.

Given a valid solution, the best matched of the previously learned maps can now be augmented and improved. If additional landmarks are contained in the pattern, they are added to the matched map. On the other hand, if there are missing landmarks in the pattern then the corresponding landmarks in the matched map are augmented to indicate the lesser certainty pertaining to these. This is accomplished through maintaining a simple “reliability” counter with each landmark node. Each missing landmark node has its corresponding counter decremented by one. However, if the count associated with a landmark is decremented to zero it is removed from the map. Each successful matching of landmarks also increases this count by two. In this way landmarks which are only detected between one third and one half of the time may still provide useful guidance information.

4.2 Map Augmentation

If a successful match has been found the matched map is augmented with any additional information observed in the currently traversed map. Currently a successful match is basically defined by the “average” match of the solution. If the average match is greater than some threshold (0.5 in our case) then the solution is considered to be a valid match. However, when an incremental matching algorithm is used here, one must also take into account the number of landmark nodes. Clearly, the fewer landmarks actually observed, the less likely the solution (currently attained) is reliably valid.

Specifically, the average match for a solution is calculated as:

$$average\ match = \frac{best\ match}{solution\ size - 1 + penalty}$$

$$penalty = \sum_{i=1}^{m} \frac{m_{count} \cdot \#missing}{\#count} + \sum_{i=1}^{0.2}$$

where solution size is the number of pairs of matched nodes in the solution; $M$ is the set of map nodes not mentioned in the solution; $\#missing$ is the size of $M$; $\#added$ is the number of pattern nodes which are missing from within the solution; tcount is again the sum of all landmark-node counts within the learned map.

Given a valid solution, the best matched of the previously learned maps can now be augmented and improved. If additional landmarks are contained in the pattern, they are added to the matched map. On the other hand, if there are missing landmarks in the pattern then the corresponding landmarks in the matched map are augmented to indicate the lesser certainty pertaining to these. This is accomplished through maintaining a simple “reliability” counter with each landmark node. Each missing landmark node has its corresponding counter decremented by one. However, if the count associated with a landmark is decremented to zero it is removed from the map. Each successful matching of landmarks also increases this count by two. In this way landmarks which are only detected between one third and one half of the time may still provide useful guidance information.

Successfully matched landmarks are also improved, in the sense that their position and relative landmark-direction are refined in accordance with the new data. The position of the landmark node and the relative direction of the landmark are improved using a weighted average between the current pattern of landmark nodes and the matched map. The landmark count is used as the weight, such that, as the landmark counters are incremented over time, less of an improvement is made to the map landmark nodes. In this way the learned map should tend toward a more “accurate” representation over time, such as where the side to side meandering nature of the robot is cancelled out.
5 Results

The set of results shown in figure 2 shows the process by which a specific section of corridor is "learned". Figures 2(a,c,e,g,i,k) show the initial raw landmark maps which are built from the robot's dead-reckoning en route, and concurrently matched to learned ones. Figures 2(b,d,f,h,j,l) show the evolution of the matched (learned) map, where each successive map is the updated and improved version of the previous, given the successfully matched current map in each case.

Figure 2(a) shows the landmarks observed on the initial run. Since no maps have been learned at this stage, a new map (figure 2(b)) is created from the initial pattern. Figure 2(c) shows the second run along the identical portion of corridor with the same landmark layout. This is successfully matched (0.622) with the current learned map (2(b)). The learned map is then improved (figure 2(d)) given the new information contained within 2(c). The third through to the sixth run also matched successfully (see table 1) with their current learned map. At each stage the learned map is refined.

Having successfully learned one piece of corridor we now put the robot in a new corridor and see how it performs. In fact, the new corridor has simply a different arrangement of landmarks. Figure 3 shows the maps built during this phase. Again, figures 3(a,c,e,g,i,k) show the patterns of landmarks as observed by the robot on successive runs, whereas the rest show the evolution of the learned map.

The first run through the new corridor is shown in figure 3(a). The new pattern of landmark nodes is then matched against all previously learned maps. The degree of match for the previous learned map (figure 3(l)) was 0.415 and hence failed. A new map is therefore created from the new pattern, shown in figure 3(b). The rest of the runs were again successfully matched with the refined versions of 3(b) (see table 2).

<table>
<thead>
<tr>
<th>Run</th>
<th>Match with map 2(l)</th>
<th>Match with latest map 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3(a)</td>
<td>0.415</td>
<td></td>
</tr>
<tr>
<td>3(c)</td>
<td>0.350</td>
<td>0.663</td>
</tr>
<tr>
<td>3(e)</td>
<td>0.281</td>
<td>0.863</td>
</tr>
<tr>
<td>3(g)</td>
<td>0.461</td>
<td>0.735</td>
</tr>
<tr>
<td>3(i)</td>
<td>0.479</td>
<td>0.606</td>
</tr>
<tr>
<td>3(k)</td>
<td>0.354</td>
<td>0.697</td>
</tr>
</tbody>
</table>

Table 2: Match data from figure 3

Having shown that the robot can correctly match and discriminate between a couple of straight sections of corridor, it should be noted that in some sense this can be considered a worst case scenario. A straight corridor is difficult to discriminate from another straight section, as there is very little difference in the shape of the corridor. Hence the pattern of dead-reckoned landmark nodes becomes increasingly important. The discrimination essentially comes down to the "uniqueness" of the pattern of landmarks.

Table 1: Match data from figure 2

<table>
<thead>
<tr>
<th>Run</th>
<th>Match with latest map 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2(a)</td>
<td></td>
</tr>
<tr>
<td>2(c)</td>
<td>0.622</td>
</tr>
<tr>
<td>2(e)</td>
<td>0.693</td>
</tr>
<tr>
<td>2(g)</td>
<td>0.808</td>
</tr>
<tr>
<td>2(i)</td>
<td>0.509</td>
</tr>
<tr>
<td>2(k)</td>
<td>0.720</td>
</tr>
</tbody>
</table>

Having successfully learned one piece of corridor we now put the robot in a new corridor and see how it performs. In fact, the new corridor has simply a different arrangement of landmarks. Figure 3 shows the maps built during this phase. Again, figures 3(a,c,e,g,i,k) show the patterns of landmarks as observed by the robot on successive runs, whereas the rest show the evolution of the learned map.

The first run through the new corridor is shown in figure 3(a). The new pattern of landmark nodes is then matched against all previously learned maps. The degree of match for the previous learned map (figure 3(l)) was 0.415 and hence failed. A new map is therefore created from the new pattern, shown in figure 3(b). The rest of the runs were again successfully matched with the refined versions of 3(b) (see table 2).

<table>
<thead>
<tr>
<th>Run</th>
<th>Match with map 2(l)</th>
<th>Match with latest map 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3(a)</td>
<td>0.415</td>
<td></td>
</tr>
<tr>
<td>3(c)</td>
<td>0.350</td>
<td>0.663</td>
</tr>
<tr>
<td>3(e)</td>
<td>0.281</td>
<td>0.863</td>
</tr>
<tr>
<td>3(g)</td>
<td>0.461</td>
<td>0.735</td>
</tr>
<tr>
<td>3(i)</td>
<td>0.479</td>
<td>0.606</td>
</tr>
<tr>
<td>3(k)</td>
<td>0.354</td>
<td>0.697</td>
</tr>
</tbody>
</table>

Table 2: Match data from figure 3

Having shown that the robot can correctly match and discriminate between a couple of straight sections of corridor, it should be noted that in some sense this can be considered a worst case scenario. A straight corridor is difficult to discriminate from another straight section, as there is very little difference in the shape of the corridor. Hence the pattern of dead-reckoned landmark nodes becomes increasingly important. The discrimination essentially comes down to the "uniqueness" of the pattern of landmarks.
This can be quite a limiting factor on the number of corridors which can be successfully discriminated, given a maximum map size. If, however, a straight corridor were to be matched against a curved one, or even simply one with turn or kink in it, the shape differences are clearly going to increase the chances of discrimination.

Having successfully learned two straight sections of corridor we now examine the performance within a curved section of corridor. Once again, figures 4(a,c,e,g,i,k) show the sequence of dead-reckoned landmark maps generated en route. The alternate maps in figures 4 again show the evolution of the learned map.

The initial run through the new corridor (figure 4(a)) failed to match either of the two previously learned maps and hence a new map is generated. As shown in table 3, each successive run was successfully matched with the current learned map and failed to match the two previously learned ones. As expected, the curved corridor was well distinguished from the straight sections.

<table>
<thead>
<tr>
<th>Run</th>
<th>Match with map 2(l)</th>
<th>Match with map 3(l)</th>
<th>Match with latest map 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>4(a)</td>
<td>0.175</td>
<td>0.159</td>
<td></td>
</tr>
<tr>
<td>4(c)</td>
<td>0.204</td>
<td>0.225</td>
<td>0.661</td>
</tr>
<tr>
<td>4(e)</td>
<td>0.263</td>
<td>0.330</td>
<td>0.567</td>
</tr>
<tr>
<td>4(g)</td>
<td>0.281</td>
<td>0.242</td>
<td>0.746</td>
</tr>
<tr>
<td>4(i)</td>
<td>0.256</td>
<td>0.208</td>
<td>0.522</td>
</tr>
<tr>
<td>4(k)</td>
<td>0.268</td>
<td>0.348</td>
<td>0.790</td>
</tr>
</tbody>
</table>

Table 3: Match data from figure 4

6 Conclusions

The map building scheme shown is clearly not an optimal solution. However, given the computation restrictions of working in real-time and the simple entomological nature of the mobile robot, the performance was quite successful. If computational overhead was not a consideration, a brute force matching regime, examining complete patterns and maps as a whole, could be imposed. This would surely produce superior results but the costs would be prohibitive, especially in a real-time application.

References


