Efficient Appearance-Based Topological Mapping and Navigation with Omnidirectional Vision

Paper-ID: 131

Abstract—Because of a mobile robot’s ability to move in its environment, one of the most important and common tasks for mobile robots is, arguably, the task of navigation. This has to be carried out in practically every mobile robot, and it has to be carried out quickly (real-time constraints) and cheaply (computational constraints). SLAM [7] is a reliable and very accurate method which is widely used now in mobile robotics.

However, these algorithms are not computationally cheap. In this paper we therefore investigate a computationally efficient algorithm for simultaneous mapping and navigation that is suitable for application in simple mobile robots. We apply self-organising maps in a novel way to image data compression and indexing of topological map nodes at the same time.

Our proposed system builds a dense topological map using only the visual appearance of the environment, with no need for any feature extraction or matching. This is made possible through the novel use of a self-organising map and a relaxed attitude towards loop closure and metric consistency. The inconsistencies and uncertainties within the map are not considered during mapping, but rather only during navigation at which point a Bayesian approach is taken to allow accurate navigation. The paper concludes by presenting mobile robot experiments to demonstrate how the algorithm performs in practice.

I. INTRODUCTION

The ability of a robot to navigate from the current location to a specified location in an environment is of fundamental importance in all applications of autonomous mobile robots. As robots become more ubiquitous in everyday life, efficient methods of navigation that can work on consumer level hardware with low cost sensors will need to be developed. Algorithms that only work on high-end research robots with powerful on-board computers will result in the cost of robots being too great for widespread deployment.

The simplest form of navigation is dead-reckoning, in which the previous motion of the robot is integrated over time to calculate the current location. Wheel encoders are the usual choice of sensor for calculating the previous motions of the robot, but inaccuracies, for example caused by wheel slippage, result in incorrigible error.

This has lead many researches to develop alternatives that rely on the use of a map of the environment, which can either be supplied to the robot in advance (“map-based” navigation [5]) or constructed autonomously as the robot moves (“map-building based” navigation [5] or “concurrent localisation and mapping” [23]). The map of the environment can be either metric, containing the positions of landmarks and robot positions, or topological, representing not metric information, but rather the connectivity and neighbourhood relationships of places.

A. Related Work

a) Metric Maps: Metric maps can be either sparse, such as those created in visual slam (simultaneous localisation and mapping) algorithms (e.g., [4]), or dense such as those created in occupancy grid based algorithms ([21]). Algorithms for 2D navigation using a metric map are relatively simple [17]. A coordinate system is available and the problem becomes one of path-planning, where the dynamics of the robot are known and the co-ordinates of ‘way-points’ must be calculated. However, metric maps are computationally expensive to create and maintain, and often contain information that is not actually needed by the robot. They are therefore not in line with the idea of “cheap navigation” proposed in this paper.

b) Topological Maps: Although optimum routes through an environment are not typically as achievable with topological maps as metric ones, topological maps are cheaper to create and maintain [11]. They are usually sparse, containing few nodes, with each node representing a “different” location — such as a different room [3] or sufficiently different perceptions [25]. Navigation using a topological map consists of finding a route through nodes in the map that gets to the desired node, and requires the ability to navigate from node to node.

The two main approaches to achieve this are to rely on odometry as in [19], or to perform “homing” though some form of servoing based on the current perception and the desired perception, commonly known as the snapshot model [2].

c) Odometry: Over short distances odometry is reliable and cheap, but in sparse topological maps such as those used by [10] odometry may fail. The solution the authors present relies on the tracking of features with an extended Kalman filter combined with 3D reconstruction from wide-baseline stereo. Navigation between the nodes is then carried out using visual servoing. This is a computationally expensive approach — again not in the spirit of this paper — and furthermore relies on the presence of detectable features.

d) Biologically-Inspired Approaches: The biologically inspired snapshot model of Cartwright and Collett [2] also makes use of features in the environment, producing a homing vector based on feature displacement. Similar approaches have been implemented in [22] and [9]. In [22] a self-organising map is used to compute the optic flow between images which is then used to compute the homing vector, whereas in [9] features are extracted from a one-dimensional panoramic image and shifted to calculate the displacement between views.

An alternative homing strategy similar to that in [9] was proposed by Labrosse in [16]. In his work the entire omni-
rectional image is used “as is”, in a purely appearance-based manner similar to the method presented in this paper, with no costly feature extraction or tracking required. However, as with all the methods derived from that of [2], strong assumptions have to be made about the distance of objects around the robot.

e) Dense Topological Maps: Little work has been done creating dense topological maps. Using a dense topological map has the advantage that odometry can more reliably be used for navigation between nodes, and routes closer to the optimum can be taken through the environment than with a sparse map. The main disadvantages of dense topological maps are the increased number of nodes to be analysed when localising, and the high computational cost of finding the best route between nodes. However, even with these added complexities the overall efficiency can still be higher than using a metric map.

f) Sensors: How the robot senses the environment is an important aspect of many navigation methods in robotics, with laser range finders being amongst the most frequently used sensors. Although high levels of accuracy can be obtained with such sensors, their prohibitively high cost and limited applicability has led many researches to consider the use of vision as a cheap and versatile alternative.

Of all vision sensors, omnidirectional vision is the most appealing for robotic applications, as it provides information about the entire surroundings of the robot in one image. In the past omnidirectional vision sensors were typically as expensive as laser range finders, but recent advances in technology have changed this and now a cheap webcam can be used with a miniature fish-eye lens to obtain panoramic vision at consumer level prices.

g) Our Approach: In this paper we propose an approach to learning the topology of the environment that uses only a cheap sensor and a computationally cheap algorithm. We demonstrate a system that learns how to navigate in an unknown environment using a single omnidirectional camera. This is achieved by combining local odometry information with a minimally processed representation of the appearance of locations. A dense topological map is created and efficiently stored and searched using a self-organising map (SOM, [13]), overcoming the additional complexity of localisation caused by the dense mapping.

II. TOPOLOGICAL APPEARANCE BASED MAPPING AND NAVIGATION

The navigation method proposed in this paper consists of two distinct learning phases:

- **Phase I**: The robot learns what kind of perception to expect. In this phase, a self-organising map is trained to cluster the perceptual space into regions.
- **Phase II**: The robot learns how the perception will change when a certain motion is taken. A dense topological map is created in association to the self-organising map learned in the first phase, exploiting the topology preserving property of the SOM as a “key” to the robot’s perceptual space. Every node in the topological map corresponds to a physical location, which is represented by the SOM classified perception at that location.

A. Visual Appearance as Perception

In this work a webcam is coupled with a 190° field of view miniature fish-eye lens, with a total cost of less than £100. The camera is positioned on a Scitos G5 mobile robot so that it is approximately at the robot’s centre of rotation and facing the ceiling, as shown in figure 1. An example distorted wide field of view image captured by the camera is shown in figure 2.
As in [16], a purely appearance based perception is used in the form of a minimally processed raw image. Using this approach, image features need not be detected, matched or tracked, and no assumptions need to be made about the presence of specific types of feature. This yields a faster, more efficient algorithm, and as shown in section IV, accurate navigation is still achievable. The relationship between image distance and spatial distance was shown to be approximately linear over short distances in [9]. This property is fully exploited in our approach.

The minimal image processing performed aims to achieve rotation and illumination invariance.

a) Rotation Invariance: Our navigation method works with a robot heading invariant perception. This can be achieved in a number of ways, for example magnetic compasses, visual compasses such as [15] and [24], or Fourier space representations such as [20]. As this is not the focus of this work, we developed a simple colour blob detection based algorithm for finding and removing the rotation from our images as follows.

Two square tiles are placed on the ceiling of the laboratory, one red and one blue. These tiles are detected as the largest red and blue regions in the $640 \times 480$ RGB image using opposing colour channel coding as in [18]. First, the blue channel of the RGB image is emphasised by creating a greyscale image $B'$ where the value of each is computed by equation 1 and the red channel is emphasised by creating a second greyscale image $R'$ where the value of each pixel is computed using equation 2.

$$B'_{u,v} = b_{u,v} - \frac{(r_{u,v} + g_{u,v})}{2} \quad (1)$$

$$R'_{u,v} = r_{u,v} - \frac{(b_{u,v} + g_{u,v})}{2}, \quad (2)$$

where $u, v$ is the pixel location, and $r, g$ and $b$ are the red, green and blue channels of the original image.

The greylevel images are then thresholded to create binary images, in which the blue and red tiles are large blobs. The centre of the largest non-zero region in each image is then selected as the tile location.

Once the image locations $(u_r, v_r)$ and $(u_b, v_b)$ of the two tiles are known, the image is rotated $\theta$ degrees so that the image locations are aligned vertically, that is $u_r = u_b$ and $v_b < v_r$. The rotation is calculated as

$$\theta = \text{atan2}(v_r - v_b, u_r - u_b) \quad (3)$$

b) Illumination Invariance: The intensity of light varies over time, and a pixel-wise comparison of two images taken at exactly the same location at different times can be misleading. In order to remove some of the effect of varying illumination, we compute a gradient magnitude image by convolving with $x$ and $y$ Sobel gradient operators. The partial derivatives $f_x$ and $f_y$ for each pixel are combined to give a new pixel value

$$G = \sqrt{f_x^2 + f_y^2}, \quad (4)$$

c) Image Size: The initial image supplied by the camera is $640 \times 480$ RGB. This is reduced to $160 \times 120$ greyscale, from which a circular region of radius 50 pixels is taken from the centre. The circular region is mapped to a 7986 dimensional vector in a spiraling concentric circles pattern as shown in figure 3. Although a column-wise or row-wise mapping would also be possible, a concentric circles mapping improves the rotational invariance by considering only the circular region.

The whole process of taking an image and turning it into the appearance descriptor is summarised in figure 3.

B. Phase 1: Learning the Perceptual Space

In the first learning phase the robot discovers what kind of perceptions are typically going to be encountered while operating. To do this, a $10 \times 10$ self-organising map is trained from perceptions logged by the robot as it is manually driven around the environment. The SOM is folded at each edge to form a torus, and trained using $N = 500$ epochs. The learning rate $\alpha$ and the update neighbourhood size $R$ are reduced at each epoch according to equations 5 and 6:

$$\alpha_{t+1} = 0.6 \exp(-10t^2/N^2) \quad (5)$$

$$R_{t+1} = 4 \exp(-10t^2/N^2), \quad (6)$$

where $t$ is the current epoch number.

The weight $\vec{w}$ of nodes within the radius $R$ of the winning node are updated according to equation 7, with a learning rate that is reduced as the distance from the winning node increases.

$$\vec{w}_{t+1} = \vec{w}_t + \exp(-r^2/R^2)\alpha((\vec{x} - \vec{w}_t), \quad (7)$$

where $r$ is the topological distance from the winning node to the node being updated and $\vec{x}$ is the training vector.

After training, the SOM coarsely splits the perceptual space into 100 clusters, each cluster corresponding to a class of perceptions. Each class of perceptions equates to an area of perceptual space which, due to the nature of the appearance-based perception vector we are using, will equate to a well defined region in physical $[x, y]$ space, where $x$ and $y$ are the robot’s position, and the heading of the robot can take any value. The topology-preserving property of the SOM means that close nodes will relate to close physical locations.

C. Phase 2: Topological Mapping

In the second learning stage, the robot explores the environment using a random walk behaviour. As the robot moves around the environment, the perception is classified using the SOM obtained in phase one. Note that rather than using only the winning node as the output of the SOM classification (as is usually the case), the 7986-dimensional perception vector $\vec{P}$ is transformed to a 100-dimensional classification vector $\vec{P}'$, with each element containing the Euclidean distance between the current perception and the weights of a node as given...
in equation 8. This method of utilising the SOM provides a compression ratio of nearly 80 : 1, at the expense of being unable to reconstruct the uncompressed perception.

\[ \hat{P}_i = \sqrt{\sum_{j=1}^{7986} (P_j - w_{ij})^2}, \quad i = 1...100. \]  

Once per second, the current classified perception vector \( \hat{P} \) and the corresponding orientation \( \theta \) is stored as a subnode under the winning node in the SOM as shown in figure 4. A directed link is placed between the new perception and the previous perception, and is labelled with the difference in the robot’s odometry since the previous perception. This method of storage allows faster matching of perceptions in the navigation phase as only the subnodes underneath the nodes in a region around the winning node need to be considered as potential matches to the current perception.

As the robot explores the environment a dense topological map, in which the same physical location may potentially be represented multiple times, is formed. The nodes that represent the same or near-by physical locations also have similar classified perception vectors that are stored as subnodes of the same or near-by SOM node. Each time a new classified perception is stored in the topological map, the subnodes of the same parent and those of the adjacent parents within the SOM are compared using the Euclidean distance between the classified perception vectors. If the perceptual distance falls below a preset threshold \( \tau \) then a similarity link is placed between the subnodes, and the link is labelled with the perceptual distance between the subnodes.

D. Navigation With The Learned Map

1) Localisation: In order to localise the robot within the topological map, the current perception needs to be matched against all stored nodes. The best matching node can then be taken as the robot location. This approach is not very efficient, requiring a large number of comparisons to be made. We avoid this inefficiency by further exploiting the topological nature of the SOM just as we do when forming similarity links in phase two. Rather than comparing the current classified perception to all existing topological map nodes, it only needs to be compared to the subnodes of the same parent SOM node and those within a preset radius of the parent (in our case 1 unit). This dramatically reduces the number of comparisons that are necessary, and prevents the need for a computationally expensive Bayesian filter based approach such as that presented in [1].

2) Navigation to a Goal Node: In order to navigate from the current location to a target location, a route through the topological map starting at the node that represents the current location and ending at the node that represents the target location needs to be found. The local odometry offsets stored for each of the links in the route then need to be combined to find the total distance the robot needs to travel. A simplistic algorithm to achieve the navigation would then simply drive the calculated distance. However, this would not take into account the uncertainties involved in the current location, the errors in the odometry offsets and the strength of similarity links.

A more intelligent approach is to formulate the problem in a probabilistic way, encompassing all of these uncertainties. For this we employ the framework of a linear Kalman filter with a simplified one dimensional state space. The \( x \) and \( y \)
offsets from the current location to the goal location are filtered independently, reducing the complexity and maintaining the applicability of our algorithm to small low cost robots.

The usual predict-update cycle is followed. First the current \( x \) or \( y \) offset is predicted as

\[
\hat{x}_k = \hat{x}_{k-1} - u_{x,k} \quad (9)
\]
\[
\hat{y}_k = \hat{y}_{k-1} - u_{y,k} \quad (10)
\]
\[
\sigma_{x,k} = \sigma_{x,k-1} + q_{x,k} \quad (11)
\]
\[
\sigma_{y,k} = \sigma_{y,k-1} + q_{y,k} \quad (12)
\]

where \((\hat{x}, \hat{y})\) is the offset to the goal location, and \((u_x, u_y)\) is the odometry distance travelled since the last cycle. The subscript \( k \) is the time step, \((\sigma_x, \sigma_y)\) is variance of the estimate and \((q_x, q_y)\) is the variance of the odometry distance travelled since the last filter cycle.

Next a measurement of the offset is made using the topological map. This measurement is then used to update the offset estimate \((\hat{x}, \hat{y})\) and it’s certainty measure \((\sigma_x, \sigma_y)\):

\[
\hat{x}_k = \hat{x}_k + \frac{\sigma_{x,k}}{\sigma_{x,k} + r_{x,k}} (z_{x,k} - \hat{x}_k) \quad (13)
\]
\[
\sigma_{x,k} = \sigma_{x,k} - \frac{\sigma_{x,k}^2}{\sigma_{x,k} + r_{x,k}} \quad (14)
\]
\[
\hat{y}_k = \hat{y}_k + \frac{\sigma_{y,k}}{\sigma_{y,k} + r_{y,k}} (z_{y,k} - \hat{y}_k) \quad (15)
\]
\[
\sigma_{y,k} = \sigma_{y,k} - \frac{\sigma_{y,k}^2}{\sigma_{y,k} + r_{y,k}} \quad (16)
\]

where \((z_x, z_y)\) is the offset to the goal location as measured by re-localising within the topological map and finding a route to the goal node from the current node, and \((r_x, r_y)\) is the variance of the measurement. The variance of the measurement is computed as a combination of the distance measures of all the similarity links within the route, the total odometry distance covered by the route and the perceptual distance in the localisation described above. The route with the least variance is selected when measuring the offset \((z_y, z_y)\).

The Kalman filtered \((\hat{x}, \hat{y})\) offset is then used as the goal position in a robot-centred co-ordinate frame, and a controller based on the dynamic window approach [8] is used in order to drive the robot towards the goal. When either the offset lies within one standard deviation of 0, or is less than 0.15m then the navigation process is terminated.

This method of navigation allows odometry drift errors and localisation errors to be smoothed and ignored without having to employ computationally costly topology checks such as those carried out in [6]. As the robot approaches a target, the errors in the map will get smaller. Provided that the robot continually re-evaluates the necessary route, then these errors will not prevent navigation.

III. COMPUTATIONAL AND PHYSICAL COST

The computational cost of the algorithm is very low, as shown in table I, with the most demanding component being the initial training of the self-organising map. This is a task that only needs to be carried out one time, and has no real-time constraints placed on it. On the whole, the algorithm is highly parallelisable, and as such is well suited for hardware level implementation, with hardware implementations of SOMs already existing [14].

The use of a self-organising map to compress the perceptions results in a low storage cost. For each node in the topological map, a vector of size equal to the number of nodes in the SOM is stored. For a \(10 \times 10\) SOM, this results in a memory requirement of approximately 390Kb per 1000 nodes. The storage of the SOM requires storing a 7986d floating point vector for each node. In the case of a \(10 \times 10\) SOM this requires approximately 3Mb.

The physical cost of the hardware necessary to run the algorithm is low as the only sensors required are a simple webcam and wheel odometers. An estimate of the cost of the hardware necessary to run the algorithm is shown in table II.

IV. EXPERIMENTS AND RESULTS

In this section we present the results of experiments testing the navigation performance with the method presented in the previous section. The MetraLabs Scitos G5 mobile robot shown in figure 1 was used in all experiments, and a Vicon tracking system was used to track the robot in order to assess the accuracy of the algorithm.

First the robot was driven randomly around an area of \(4 \times 4\) metres, and 1000 images were logged at a rate of 2 Hz. These images were then turned into 7986-dimensional perception vectors as described in section II-A and used to train a SOM as described in section II-B.

Next the robot was left to autonomously learn the environment. This was achieved by programming the robot to randomly move around within the \(4 \times 4\) metre area, creating new topological map nodes once per second. In total 4000 nodes were created. For each node created in the map, the...
ground truth world co-ordinates were recorded in a separate file to allow later evaluation.

A. Navigation to a Goal

In order to assess the navigation method, the robot was placed at a goal location and the perception at that location was stored. The robot was then moved to a different location and the navigation algorithm described in section II-D was applied. This was repeated with 15 different starting positions.

The results are shown in figure 5. The goal location is shown as a large diamond, and the start and end positions of each run are shown with open circles and stars respectively. Figure 6 shows the filtering of the offset estimation during one of the navigations.

The results show that in all tests the robot was able to move closer to the target perception. The absolute mean error in final position of the robot was $0.283 \pm 0.027$ metres. The maximum and minimum position errors were 0.456 m and 0.137 m respectively.

The route taken to the goal by the robot is often not completely direct, as can be seen in run 13. This is because the offset to the goal location, as measured using the topological map, fluctuates around the true offset, and a combination of the Kalman filter and the physical robot causes a smooth trajectory to be taken. In run 1 the robot can be seen to initially be driving away from the target. However, this is not a result of errors in the topological map, but is rather caused by our use of the dynamic window approach to controlling the robot. In this case the robot was initially facing away from the goal, and in order to achieve a continuous smooth trajectory the linear velocity was kept at a minimum of 0.2m/s.

B. Random Goal Navigation

The navigation algorithm was further tested by performing 300 navigations to different map nodes. The destination map nodes were automatically selected at random, and each trial took place sequentially. The difference between the finishing location of the robot and the target node location in external world coordinates was recorded. A two dimensional scatter of these errors is shown in figure 7.

As can be seen in figure 7, the errors in the ending $x$ and $y$ position each approximately follow a normal distribution. For the error in the $y$ direction, the mean error is 0.045 metres with a standard deviation of 0.6012 metres. For the $x$ direction the mean error was $-0.0186$ metres with standard deviation 0.5313 metres.

On some of the navigations the finishing position error was large, and indeed the navigation was manually terminated outside of the mapped area. These are examples of complete failure of the system, and occur whenever the robot drives outside of the mapped region. Once outside of the mapped region the localisation can not possibly succeed, and getting back to the known area is only possible if the mapping phase is still active.

Overall, of the 300 navigations 88% finished within 1 metre of the target and 67% finished within 0.5 metres.
<table>
<thead>
<tr>
<th>Run</th>
<th>Start Position</th>
<th>End Position</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X [m]</td>
<td>Y [m]</td>
<td>X [m]</td>
</tr>
<tr>
<td>1</td>
<td>0.4286</td>
<td>1.4557</td>
<td>-0.7138</td>
</tr>
<tr>
<td>2</td>
<td>1.7701</td>
<td>-0.5042</td>
<td>-0.3315</td>
</tr>
<tr>
<td>3</td>
<td>0.7873</td>
<td>0.8325</td>
<td>-0.2777</td>
</tr>
<tr>
<td>4</td>
<td>1.8285</td>
<td>-1.5733</td>
<td>-0.173</td>
</tr>
<tr>
<td>5</td>
<td>1.59</td>
<td>1.3649</td>
<td>-0.3707</td>
</tr>
<tr>
<td>6</td>
<td>0.1633</td>
<td>1.251</td>
<td>-0.4596</td>
</tr>
<tr>
<td>7</td>
<td>2.1041</td>
<td>-0.1425</td>
<td>-0.3585</td>
</tr>
<tr>
<td>8</td>
<td>1.292</td>
<td>-1.925</td>
<td>-0.2104</td>
</tr>
<tr>
<td>9</td>
<td>2.3676</td>
<td>1.0939</td>
<td>-0.2114</td>
</tr>
<tr>
<td>10</td>
<td>0.4392</td>
<td>2.0428</td>
<td>-0.3492</td>
</tr>
<tr>
<td>11</td>
<td>-1.2498</td>
<td>0.0935</td>
<td>-0.2495</td>
</tr>
<tr>
<td>12</td>
<td>2.4885</td>
<td>1.53065</td>
<td>-0.5892</td>
</tr>
<tr>
<td>13</td>
<td>0.4317</td>
<td>1.0652</td>
<td>-0.2106</td>
</tr>
<tr>
<td>14</td>
<td>-0.2226</td>
<td>0.53</td>
<td>-0.4398</td>
</tr>
</tbody>
</table>

**TABLE III**

The navigation results for fifteen runs to a goal location of \((-0.487, -1.329)\).

---

**Fig. 7. A scatter plot of the errors for 300 navigations.**

**Fig. 8. Fifteen navigations of a route described by five waypoints indicated by 5 crossed circles.**

---

**C. Route Following**

As a demonstration of the algorithm, a route following experiment was performed. The robot was manually driven to five waypoint locations, and the topological map node was recorded. The robot then autonomously navigated along the route through these points using the navigation algorithm described in section II-D. Fifteen loops of the route are shown in figure 8.

**V. CONCLUSION AND FUTURE WORK**

In this paper a new environment learning and navigation strategy has been demonstrated that is both effective and computationally cheap. The method makes use of dense topological mapping to allow odometry information to be successfully used for navigation, while at the same time maintaining efficiency through the use of a SOM to index the perceptual space. The combination of a cheap sensor and a computationally simple approach will pave the way to the more widespread deployment of mobile robots.

The self-organising map has been shown to be a suitable tool for clustering the perceptual space, and our use of the SOM within the algorithm not only allowed us to improve the efficiency by compressing the vectors, it also allowed us to increase the speed by reducing the number of comparisons that were required during localisation. However, it is the only aspect of the method preventing entirely on-line operation. If an alternative clustering method capable of on-line operation were to be used then an entirely on-line learning approach would be achievable.

The method presented in this paper assumes the orientation of the robot is known. Although the simple method of finding this described in section II-A was utilised here, a more generic method will need to be used in order to allow more widespread applicability. Future work includes allowing arbitrarily orientated images, perhaps utilising methods similar to those in [12].

**REFERENCES**


