BUILDING A MAP FOR ROBOT NAVIGATION USING A THEORY OF COGNITIVE MAPS

M.E. JEFFERIES\textsuperscript{1}, W.-K. YEAP\textsuperscript{2}, L. SMITH\textsuperscript{3}, D. FERGUSON\textsuperscript{3}

\textsuperscript{1}Department of Computer Science
University of Waikato, New Zealand

\textsuperscript{2}Artificial Intelligence Technology Centre
Auckland University of Technology, New Zealand

\textsuperscript{3}Department of Computer Science
University of Otago, New Zealand

ABSTRACT

In this paper we describe the application of our theory of cognitive maps \cite{1} to the problem of building a map for a navigating robot equipped with sonar sensors. At the core of this theory is the notion that a representation is computed for each local space the robot visits. These representations, connected in the way they are experienced, form the robot’s cognitive map. Real world implementations which compute representations of a robot’s environment face the problem of errors due to inaccuracies in the sensory data and errors in the robot’s location due to wheel slippage. Sonar sensing devices have the added problem that there is not much sensory data to begin with. We show that using our theory of cognitive maps it is possible to build a reasonable representation of the robot’s environment without complex error correction procedures.

KEY WORDS

cognitive maps, robot navigation, spatial reasoning

1 INTRODUCTION

We have developed a theory of cognitive maps which has at its core the notion that an autonomous agent building a memory, i.e. a “map in the head” termed a “cognitive map”, for the places it visits must first compute a representation for each individual space visited \cite{1, 2}. The space the agent occupies, termed the “local space”, is defined as the bounded region which the agent perceives as enclosing it. The agent’s cognitive map develops as the representation for each local space visited is added to a topological network of such representations (see Figure 5). We initially tested this theory in a simulation of a robot traversing a complex 2D representation of a hospital environment. The results of this simulation and a comprehensive description of the theory and the algorithms employed in its implementation can be found in \cite{1}. Here, we show how the theory can be applied to the problem of an autonomous mobile robot, equipped with sonar sensors, building a map from its own experience of the places it has visited.

Real world implementations of algorithms which compute representations of a robot’s experience of its spatial environment face a major hurdle. Odometric and sensory errors quickly accumulate in the representation as the robot moves around its environment and the current sensory information is integrated with what has gone before. The representation of the environment soon becomes misaligned with the physical environment it represents. These errors are catastrophic for robots which rely on precise positioning to determine if they are visiting a part of the environment which has been encountered previously. This is particularly so for grid-based methods which have become a popular paradigm for representing the robot’s environment because they work well with data captured from sonar \cite{3, 4}. The sensory data is captured in a fine grained grid which if one imagines it laid over the physical environment then each cell records if the corresponding space is occupied or not. Other problems with occupancy grids include their computational complexity and not being able to determine if they are occupied or not.

Other methods for representing the robot’s environment, termed “feature-based”, extract important geometric information (e.g. edges and corners) directly from the sensory data \cite{5, 6}. However extracting this information from sonar is difficult due to the spurious readings which result from specular reflection. Recent efforts have been directed at finding methods which give better estimates of the actual surfaces in the physical environment \cite{7}.

Our approach to computing a representation of the environment is feature-based, however it differs from other methods using this paradigm in that we believe that it is not necessary to compute an accurate representation from the robot’s early experiences of its environment. We have noted from studies of psychologists and geographers on cognitive maps for human and animal navigation \cite{8, 9} that the most important information a navigating agent needs to compute is exits because they tell the agent how it can leave the space it is currently in. From a computational point of
view it is much simpler to obtain surface information from exits than it is to obtain exits from surface information. More importantly, we can acquire a reasonable and useful boundary without detailed surface information. In the next section we briefly outline the exit-based algorithm for constructing a representation of an agent’s local space. In Section 3 we show that the implementation of this algorithm on a robot is the foundation for building a cognitive map of the places the robot visits. We present our conclusions in Section 4.

2 BUILDING A COGNITIVE MAP OF LOCAL SPACE REPRESENTATIONS

The cognitive map comprises a local space representation for each local space visited with a connections to others which have been experienced as neighbours. Each local space representation has its own local coordinate system and is thus independent of all the others. We therefore term the local space representation the Absolute Space Representation (ASR) and will use this term throughout the paper.

From the view the agent has of its environment, the ASR algorithm firstly works out where the exits are. It does this by looking for surfaces which occlude other surfaces; it is here that the gaps, i.e. exits occur. For each occlusion the algorithm determines which part of the gap associated with it actually forms the exit. For example, in Figure 1 (a) the exit could lie between the vertices s4-vertex2 and s8-vertex1, or it could be the perpendicular formed between s4-vertex2 and s5, or the perpendicular formed between s8-vertex1 and s7. The algorithm chooses the shortest of these and the exit lies between s4-vertex2 and s8-vertex1. This is exit e2 in Figure 1 (b). Once an exit is identified it becomes obvious which surfaces are viewed through the exit and as such are not part of the boundary of the ASR. For exit e2 these are surfaces s5, s6 and s7. Surfaces which form an unbroken connection between two exits, for example s3 and s4 which lie between exits e1 and e2, are included in the boundary. Exits computed as above have a dual role, in the traditional sense to indicate where one can leave the current space and to indicate parts of the current ASR which are yet to be uncovered. These two roles are distinguished by labelling the latter as unknown and the former as known. As the viewer moves around the local space parts of it that were once unknown are no longer so, and the exits covering these areas are updated. We describe the updating process further in the next section and we refer the reader to [1] for an in depth discussion on the role of exits and the updating process. Eventually the robot will leave the current local space; as it does so it will be entering either a space it has never encountered before or one that has already been visited. For the former, an explicit connection will be made in the robot’s cognitive map, linking the space the robot has just left with the one it has just entered. The link is made via the exit used to go from one local space to the other. An explicit connection will also be made for the latter if there is currently no connection in its map linking the two ASRs with the exit just crossed. ASR recognition is difficult at this stage in the cognitive mapping process due to the meagre information available. However we have had success in recognising these raw ASRs using a small global map, called a “Memory for the Immediate Surroundings”, in conjunction with the topological map. We refer the reader to [10, 1] for our work in this area. Later in the cognitive mapping process, as landmarks come to be identified with specific ASRs, more sophisticated recognition techniques can be employed.

3 A MOBILE ROBOT BUILDS A COGNITIVE MAP

Constructing the representation for the local space

The robot we have implemented the ASR algorithm on is a Real World Interface B14r robot with sixteen sonar sensors. Computing a coarse boundary for the ASR does

![Figure 1 The construction of the ASR. (a) the environment (b) The local space boundary](image-url)
not require complex surface extraction techniques. This turned out to be important for a robot using sonar to sense its environment. The first step in the algorithm constructs what we term a simple map. This part of the process finds the occlusions and skeletal surface information. To locate the occlusions we first need to know where the end points of surfaces are. There are two problems with sonar data. First, with a 360 degree coverage provided by 16 sonar sensors the resolution is not sufficient to determine where one surface ends and another begins. Each sensor detects just the surface which is closest to it. Second, the point which is detected by the sonar could lie anywhere across an approximately 15 degree cone. To overcome this difficulty we exploited the fact that, while we could not tell whereabouts in the cones the readings were coming from, we did know the approximate whereabouts of the edges of the cones. The trick is to detect when the end-point of a surface is on the edge of the cone. For example, in Figure 2(a) surface A is just within range for the sonar sensor, i.e. when a sonar sensor reading is taken our system returns the distance to surface A. We detect that this reading came from the end-point of surface A by turning the robot a small amount and taking a new sonar reading. In Figure 2(b) when the robot turns 60°, surface A falls just outside the range of the sonar sensor and the distance to surface B which is further away from the robot is detected instead. Since surface A is no longer in range the previous reading must have detected the end-point of surface A. Thus to detect the occlusions, we turn the robot through 24° in 6° intervals. An occlusion is detected when the disparity between adjacent readings for the same sonar sensor is greater than half a metre. Figure 3(a) shows the occlusions which were detected for the robot’s initial view of the environment depicted in Figure 4. The occlusions are the lines labelled occ. Note that in this environment, many of the obstacles that look as if they can be stepped around to a human, do not appear so to a robot. They block the robot’s line of sight and appear to be part of the boundary the sonar has detected in between. These rough surfaces are the dark lines not marked occ in Figure 3(a). With sonar sensing the resolution decreases rapidly as a surface’s distance from the robot increases. At far away distances the whole exit will fit within the arc of the sonar beam, making them undetectable. Therefore we discard all sonar readings returned from surfaces more than two metres from the robot. These surfaces are indicated by the light coloured lines in Figure 3(a). Gaps in the boundary which result from this elimination process are marked as unknown. They mark areas that the robot will explore next. The algorithm then proceeds as described in Section 2. From the simple map exits are constructed where there are occlusions and surfaces outside these exits are eliminated. Figure 3(b) shows the resulting ASR. From Figure 4 it can be seen that this ASR has just two exits, however, eight have been computed. The six false exits result from specular reflection. They will be eliminated as the robot moves around the room and sensor readings are collected from different locations.

Extending the local space representation

This initial description of the room roughly describes the space surrounding the robot and identifies some exits the robot could try to leave the room by. But more importantly this initial representation forms the basis for further exploration; the robot moves to “unknown” areas of the room so these parts can be captured in the representation; it moves close to exits to verify that they are indeed exits. In this section we describe this “filling in” of the initial ASR with a “better” description as the robot moves around the local space.

The robot visits each of the unknowns and exits in turn, moving to a position where it can get a better view of the region of the ASR they encompass. Figure 3 (b) shows the robot using the ASR to plot a path to the point marked x in front of the unknown, U1. It moves to this position and takes a new set of readings which are used to create a new simple map. Figure 3 (c) shows the new simple map for the point x in Figure 3 (b). Points from this map in the region of the unknown U1 are incorporated into the previously
Figure 3  (a) The simple map constructed for the robot’s first view of the local space. (b) The ASR constructed from the simple map in (a). (c) The simple map constructed when the robot moves to position x in (b). (d) The simple map resulting from incorporating the exposed unknown region in the simple map in (c) into the simple map in (a). (e) The ASR reconstructed from the simple map in (d). (f) The ASR constructed after all the unknowns and exits have been explored.
constructed simple map (Figure 3 (a)). The ASR is then re-
constructed from the updated simple map (Figure 3 (d)). In
this way the simple map serves to collect the data needed to
construct an ever better ASR. However it is the ASR itself
which is used to decide its own completeness. The algo-
rithm repeatedly visits unknown regions until they are all
completely exposed in the ASR.

The process for checking exits is similar to that for up-
dating unknowns. The robot moves close to an exit to get a
better view. A new simple map is constructed. Points
around the region of the exit in the new simple map are in-
corporated into the previously constructed simple map. The
ASR is reconstructed from the updated simple map. If the
exit remains in the reconstructed ASR then it is deemed that
the exit is not a product of specular reflection and that it re-
ally does exist. The process continues until all exits have
been checked. The ASR which results from the completed
validation process is shown in Figure 3 (e). Figure 4 shows
the ASR overlaid on the actual environment. It can be seen
that the false exits E2 and E4 were not able to be eliminat-
ed. The success of the process relies on the robot at some
stage positioning itself so that the sonar beam is reflected at
an angle perpendicular to the specular surface. However,
because the robot does not distinguish specular surfaces
from non-specular surfaces it may not always achieve this.

Finally, Figure 5 shows the ASR of Figure 4 along with
some other ASRs in its cognitive map. The arrows indicate
the exits which have been used to cross into ASRs. For ex-
ample exit E3 in ASR2 and E5 in ASR3 identify a gap in
the boundary of their respective ASRs which can be used to
cross from one of these ASRs into the other. By keeping
track of the exits in each ASR the robot can use this map to
navigate its environment.

4 CONCLUSION

We have shown how a useful representation of a ro-
bot’s local environment can be computed from sparse, in-
accurate sonar sensory data. The robot has a rough idea of
the extent of its local space and there are some exits which
can be used to get out of the current local space. The exits
are not exact but they do not need to be. When the robot ini-
tially heads towards an exit, it need only be pointing in the
right direction at first. As it gets closer to the exit and re-
ceives feedback from its sensors it can make adjustments
for any discrepancies it detects. We can eliminate most of
the false exits. Of those that remain, the robot would even-
tually get close enough to detect that they too were false.
Our algorithm does not require any sophisticated error cor-
rection techniques. At each step in the process it does no
more than incorporate the current raw sensory information
into the robot’s current representation of its local environment. The resulting ASR is the appropriate unit of representation for a topological map, i.e. the cognitive map, of the robot’s environment.

REFERENCES