Computer Vision Algorithms for Autonomous Mobile Robot Map Building and Path Planning

Stuart Meikle and Rob Yates
MIEE *MIEEE, SPIE*
s.meikle@shef.ac.uk , r.yates@shef.ac.uk

Abstract

This paper describes a new algorithm for autonomous mobile robot (AMR) map building and path planning in an unknown environment The robot learns its environment using a view-based probabilistic interpretation of the visible corner and edge features. No attempt is made to locate physical world features in a traditional Euclidean coordinate system, rather the algorithm produces map building through measures of visual similarity alone. A robust map building algorithm is developed, whose robustness is notably, independent of the size of the learnt environment. This enables large environments to be learnt reliably. To date no other AMR map building algorithm is capable of robustly learning an arbitrary sized unknown environment. The robot performs robust intra-location localization and inter-location navigation using an internal representation of the spatial arrangements of nearby known locations.

1. Introduction

A basic requirement for any autonomous entity is the ability to explore and learn an unknown environment, so as to be able to find those things required to sustain the existence of the entity. A truly autonomous mobile robot (AMR) needs to be able to refuel or recharge itself as required, and to be able to stay out of danger or harmful situations.

Many researchers have attempted to solve the map building problem through the use of model-based techniques. With these techniques robots attempt to construct Euclidean 2 or 3 dimensional models of their environments using position measurements of the physical features they observes. This is an extensively used technique [1][2][3], which works well for small or previously known environments. However the method is unreliable and inflexible for larger environments[4][5]. All features observed have to be reconciled with the whole of the robot's learnt map. Therefore feature positioning errors are cumulative, making the maps constructed less and less reliable as the size of the environment increases. Model based maps have the advantage that they can be compared directly to the real world in order to verify their accuracy.

More robust techniques involve topological maps rather than topographical maps[6][7], where the interconnections between various places in the environment are learnt , rather than the exact topography of the environment. View-based visual feature and object recognition strategies are also becoming favorable[8]. It is interesting to note that there is biological evidence for the use of both topological place maps and view-based recognition in mammalian brains. Rats have been shown to have cells in their hippocampi which respond to particular locations in their environments[9][5], and there is much evidence to suggest that object recognition in humans and primates is view-based [10][11].

Notable amongst AMR solutions is the MIT robot MAVIN which is capable of learning an unknown environment using a view-based topological technique, aimed at emulating the response of the hippocampal place cells in rats[12][5]. It succeeds in doing this, however it uses complex algorithm and carefully constructed visual features rather than ordinary physical world views. In contrast, the solution described here is simple, flexible and extensible. We aim towards a general solution, and do not construct objects by which to navigate by, rather we use whatever is visible in the environment. Furthermore our work has been tested on significantly more complex environments!

This work discusses the proposed algorithm with particular attention given to the strategies used to tackle the complex problems involved. A methodology for building extensibility and robustness into complex sensor dependent algorithms is given, along with experimental results and conclusions.

2. Building a Probabilistic Framework

Central to the methodology of this work is the recognition that when designing large, complex systems with a need for robustness, *sensor errors must be known and this knowledge must be propagated through all levels of the system*. This is crucial! If input system errors are modeled incorrectly or are ignored then the system produced will not be robust - subsequent processes will not be capable of responding correctly to the inescapable fluctuations in the input data. What is needed is a

probabilistic interpretation of the input system measurements.

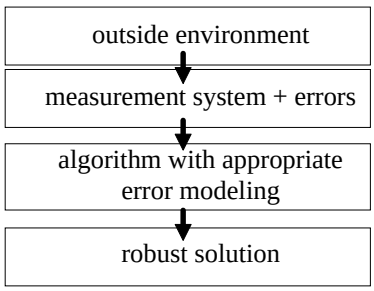

Figure 1. An idealistic process for producing robust algorithms. All physical measurements have inherent errors. These must be taken into account in subsequent processes.

Figure 1 shows a simplified model of the stages involved in robust algorithm development. This is the ideal case. In reality it may not be possible to model the errors of the input signal; it may be difficult to propagate these errors through any calculations, or the input signal may have unpredictable components, such as random impulse noise (also called salt-and-pepper noise) which can be present on video signals. This can change the input in a fashion which is difficult to model. Regardless of the input errors, the more accurately the errors in the input system are modeled the more robust subsequent algorithms can be.

For this research, the input measurement system takes the form of a CCD camera image, from which corner and edge information is extracted. (This is image gradient information). This is the only information which is used in all the map building processes. Corners and edges are visual features which are robust to changes in lighting. In other words, they can be located repeatedly in an image regardless of changes in the scene lighting. The most significant errors associated with these features is the error in their position. For edge detection the Canny's algorithm was used and for corner identification, the Harris & Stephens algorithm was used. Both of these algorithms provide information about how well their features are localized. With Canny, practical results suggest a position error of $\pm$ 0.1 pixels for each edge, and with the Harris & Stevens a corner position error of $\pm$ 0.3 is given. (See [13][14]). Effects due to speckle noise are small and will be ignored.

3. C.L.A.M.

The Contextual Layered Associative Memory is an extensible, multi-layered, neural network classification scheme[15]. It is based on the precept that its inputs are probabilistic representations of sensor measurements which can be interpreted as frequency coded distributions. (This is consequently a necessary requirement for this work). Provided this is the case, inputs can be compared using known statistical methods. These can be used to calculate the probability that any two input representations are derived from the same real world phenomenon after sensor errors are taken into account.

In a real application these continuous distributions are represented using an array of quantized bins. These arrays can then be thought of as n-dimensional vectors where n is the number of bins used. If features can be represented using such a distribution, then each feature can be also described as a vector, **v,** which lies within the n dimensional feature pattern space. (With CLAM a normalized distribution is used, making **v** a unit vector). With any frequency coded distribution, the errors associated with a given element or bin contents will be given by Poisson errors, i.e. $\pm\sqrt{n}$, where n is, in this case the relative frequency of that measurement (see below). In fact, this becomes $\pm k\sqrt{n}$ after normalization. Similarly each neuron in the input layer represents a vector **w** which defines a location in the feature pattern space corresponding to a known feature distribution. The dot product of these vectors, **v.w** , yields the probability that both are derived from the same real-world features. The responses from all neurons in the first layer are absolute probability measures, and thus can be collectively considered as another frequency coded distribution, or vector. This distribution can then be processed by the second layer in the same manner that the input vector was processed by the first layer. Feed forward and feed backward weights between the layers ensure that vectors remain correctly normalized - all probability measurements must sum to 1.0! In this way information can be abstracted to any level, and the need for complex error propagation is avoided.

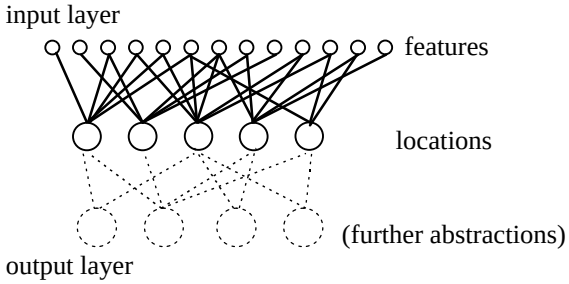

Figure 2. A symbolic representation of the CLAM architecture used in this work. Distinguishable real-world corner features are learnt by the network. Responses from all features visible from specific locations are further grouped into visually disseminable locations.

In this work we use a more simple form of CLAM where only the output of the winning neuron from a given layer is passed down to the subsequent layers. For map building and path planning the network is constructed and maintained as follows: each feature representation is given to each of the nodes in the first layer. Each node in this layer will produce a probability of overlap. If this exceeds a fixed threshold then the feature is deemed to lie within the classification space of the node, causing the node to 'fire' and the result be passed down to the second layer. If no nodes fire then the representation is deemed to be novel, and a new node is created to represent that feature.

The winning node's response from the first layer is then passed to a buffer where it is stored along with the responses from other features observed by the robot from this particular location. Once the location has been examined these stored results are passed to each node in the second layer, which produces an output value by the same method as for the nodes in the first layer. Similarly the buffered results can be added to the second layer as a novel location node.

4. Constructing Frequency Coded Distributions

In order to use CLAM it was necessary to construct a probabilistic interpretation of the corners and edges features extracted from the AMR's input image which would be suitable for view-based feature recognition. Specifically, it was desirable to produce a representation which would facilitate the recognition of features from different viewing positions and distances. This was needed in order to reduce the number of feature representations stored and to improve to ability of the robot to recognize previously visited locations. However increasing the invariance of features with respect to viewing conditions reduces the number of representations which can be stored [16].

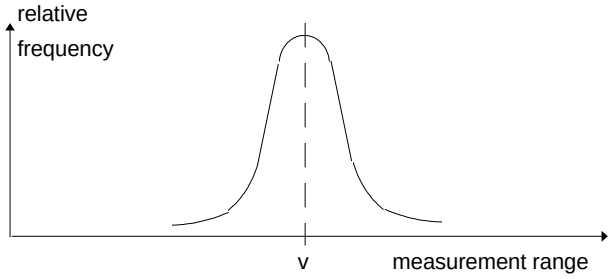

Figure 3. Measurement of a quantity v with a real sensor system produces uncertainty. Repeated measurements produce specific measured values each with a specific relative measurement frequency.

The feature representations used in this work are two dimensional Probabilistic Hough Transforms (PHTs) [17]. In effect these are matrices of normalized floating point numbers which encode the measurement errors produced by the input sensors. For example a simple 1D measurement of an erroneous signal could produce a distribution as shown in figure 3. This distribution can be represented as a PHT with a finite 1D array of numbers.

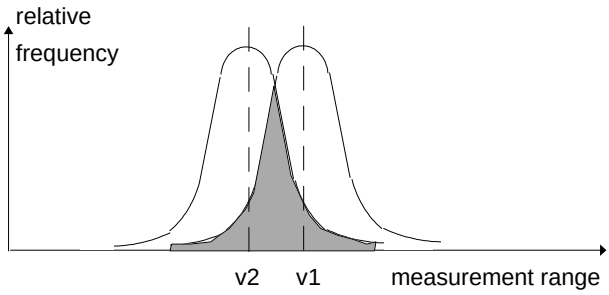

Figure 4. Two overlapping 1D PHTs from different measured values, v1 and v2. In reality these continuous distributions are represented using finite element arrays.

If another value were measured, and a similar PHT constructed, then the probability that both measurements were in fact due to the same physical feature would be given by the overlap of the two curves (see figure 4), as has been mentioned. More complex PHTs can be compared using known statistical distance metrics which provide a similar overlap function. We chose to use a specific one, the Bhattacharyya distance metric, which has been shown to be the correct method to use for frequency coded data[18]. This is directly equivalent to the dot product vector overlap mentioned in section 3, however the vectors are square rooted beforehand in order to transform them from the distorted error space of Poisson measurements to a flat error space in which a true measure of the distance between the distributions can be found. (Again, see [18]).

$$I_b = \sum_i \sqrt{p_{1i}} \cdot \sqrt{p_{2i}}$$

Equation 1. The Bhattacharyya distance metric can be used to compare two (normalized) matrices. Here p1 and p2 are quantized probability distributions such as those in figure 4.

For this work we use a corner-based feature representation. Corners provide well located features, and are less numerous than edges. The representations formed are similar to the log-polar representations used by other

researchers[8], and in the human retina. We call these Cornerwise Geometric Histograms, following on from previous work done at the University of Sheffield on probabilistic edge-based representations [19]. Figure 5 below shows the information used in encoding this information.

The cornerwise histograms encode the observed edge strings surrounding a given corner; they are 2D PHTs which represent the edge string in polar coordinate/ edge orientation space (see figure 6). As the robot explores it creates a single histogram for each corner viewed, which encodes geometric information about the edges surrounding that corner. Entries are weighted with inverse distance from the corner feature, in order to localize the information plotted.

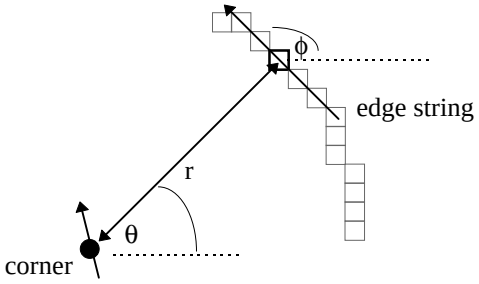

Figure 5. Geometric Histograms store information about the orientation and polar coordinate of each edgel surrounding an observed corner feature. Entries are weighted so that information near to the corner becomes more significant.

In essence, for each edge element in the image an entry is made into the histogram at a location determined by the edgels polar coordinate with respect to the feature (θ of fig. 5) and the relative orientation of the edgel (ϕ of fig. 5). The entry is weighted with an inverse distance measurement to ensure that information local to the corner feature dominates the plots. *All* edge strings in the image are used for each feature representation.

$$w_{\theta,\phi} = \sum_{N} \frac{g_{\theta,\phi}}{r_n + k}$$

Equation 2. The weighting function applied to each element of a geometric histogram. N is the number of edgels surrounding the corner, r is the distance from the corner to the edgel. 'g' is a Gaussian kernel used to represent known errors in the edgel orientation and polar position and k is a small constant added to remove infinities.

This process is repeated for all corners in the central region of a given image. Only corners near the center of the image are used as they are well described by the surrounding edgels. Corner features near the edges are less well represented as they are not wholly surrounded by the edgels which would be observed local to them.

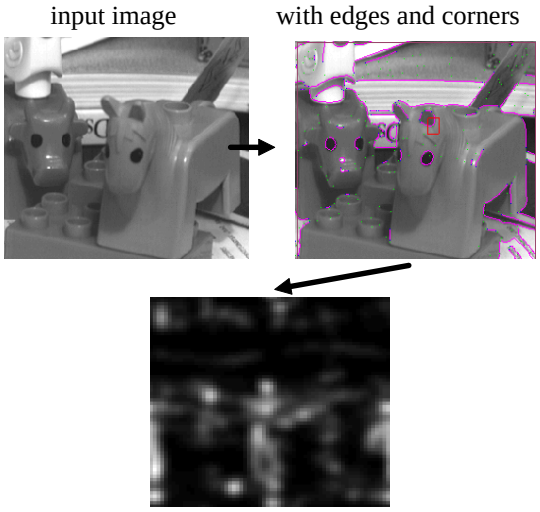

Figure 6. (a) An instantaneous visual scene from an AMR camera is captured. (b) corners and edges are extracted from this image. (c) a cornerwise histogram is formed for each corner near the center of the image. This encodes the geometric properties of the edgels surrounding the corner.

The known position errors of the edgels and corner features are taken into account and modeled on the Hough Transform as Gaussian distributions for each measurement. The plots were carefully constructed so that the edge information is well represented in the transforms.

Results from experiments with real images show that the representations formed are invariant to changes in rotation about the line of sight and are reasonably invariant to changes in scale. Only three representations were needed to encode a simple shape over a scale change ratio of 1:4. It is likely that more representations would be needed for encoding typical real-world features, however.

5. Storing Features in the Clam Network

A list of all known (visually different) features can be formed by using known statistical distance metrics to partition the probabilistic representations based on similarity. As mentioned, if two normalized representations are compared by calculating the overlap integral, the result will be the probability that both representations are derived from the same set of physical

features. Thresholding this value allows a set of visually different features to be formed. (Care must be taken in choosing an appropriate threshold value so that as many different features as possible can be learnt whilst staying above the background noise level.) Novel features are features which fall outside of all known feature representations by some distance greater than the threshold value. This list of all known features forms the first layer of the CLAM network. Re-observing a known feature will produce a representation (*in all probability*) which lies very close to the previous feature 'exemplar'. A winner-takes-all strategy is used in comparing a single representation to all those known.

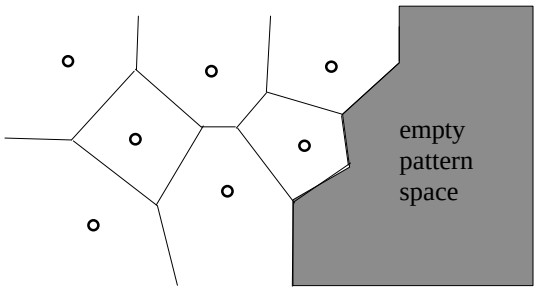

Figure 7. Each neuron in the first layer of the network responds to features in a specific volume of the feature pattern space. The density of these nodes is set by an arbitrary distance threshold. Input representations falling in empty pattern space are deemed to be novel and are added to the network as new neurons.

The second layer of the network stores lists of responses from the features - these are the responses from all known feature representations in comparison to features observed from a specific location in the AMR's environment. This list will have *variable length* as the number of known features will increase continuously as the AMR explores. This makes comparison between lists difficult. A tailor-made distance measure has been developed as part of this work, in order to partition these variable-length (or equivalently, variable dimension) response vectors based on similarity. This will be referred to as the 'Similarity Measure'. Using this measure allows a set of different response lists to be maintained, in the same way that the set of feature representations was built for the first layer (see figure 8). These response lists represent *visually distinct* locations in the AMR's environment. For an AMR which has learnt a static, closed environment in its entirety there will be a single second layer neuron which will fire in response to the features observed from each location in the environment.

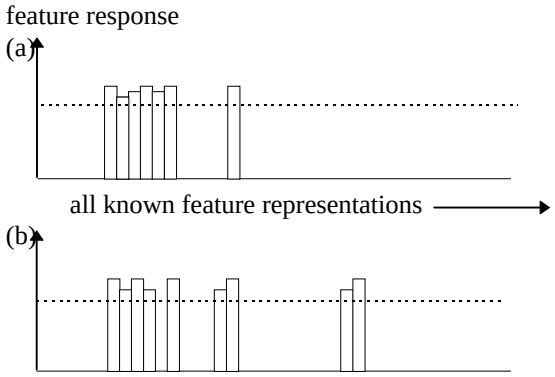

Figure 8. The Similarity Measure can be used to provide a distance measurement between two different feature response lists, such as those formed for two different locations. As a consequence of the first layer output all feature responses will lie above a threshold value.

5.1. The Similarity Measure

The Similarity Measure is a piece-wise function combining a known statistical comparison metric with a function which causes the comparison result to drop off as the number of zero-value feature responses increase. In designing a robust, probabilistic distance measure for the feature response lists, there is a problem: because there is no a-priori information about the structure of the AMR's environment , it is impossible to say what the probability is that a given feature will be observed from a given location. This probability will be feature and environment dependent, so no single value can be given to it. This poses a problem as it means that the probability of mis-identification of locations can not be calculated; the best that can be done is to provide a distance measure which tails off smoothly as feature responses drop and disappear below the feature threshold.

$$r = \frac{\sum (\sqrt{a_i} - \sqrt{b_i})^2}{\sqrt{\sum a_i + b_i}} . (\delta + k_1 (f - k_2)^{k_3})$$

Equation 3. The Similarity Measure for comparing two distributions, a and b. 'f' here is the fraction of non-zero data, and delta is a binary function for non-zero data summation.

6. Map Building

How is this methodology of use in the context of AMR map building? Using this network an AMR can explore an unknown environment, and can segment it into different locations based on visual similarity. (Assume for now that

the environment is static). Thus an environment can be learnt as a series of visually different locations. Note that no topological information is stored with the map, it is purely a list of feature representations and a set of response lists from each location. The topology of the environment is inferred by the robot at the time of global path planning, by examination of which known locations share common features. (This enables paths to be found along previously untraversed routes.) We are not limited to the number of locations which can be learnt, *except* by the errors present in the sensor system. Ultimately this will limit the number of different histograms which can be formed, which will limit the number of locations which can be learnt. The architecture scales well with the size of the environment, growing roughly linearly with the number of novel features observed.

7. Software Simulation

In order to test the network strategy a simple software simulation was devised to determine whether or not an AMR could in fact partition an unknown environment into different locations. Sample results are shown in figure 8, below. The network was seen to behave as expected, responding to known locations in a similar fashion to biological systems: individual neurons produce place-field like responses. In this particular experiment features were known a priori; the environment was segmented by considering which features were visible from each location, and forming response lists appropriately. The environment was planar the segmentation process can be thought of as either 2 or 3 dimensional, as the robot only needs to know which features are visible at this stage, and not what the relative positions of the features are.

For this particular experiment the simulated AMR needed no odometry, or similar internal measurements, whatsoever. Only visual information was used. With a self-constructed network for a given environment, the AMR is able to localize itself to a specific known location by looking around and comparing the list of feature responses to all those known.

8. Path Planning

8.1. Local Navigation

Another criterion that must be fulfilled is the ability of the AMR to navigate from known location to known location. I.e. the AMR must be able to (1) infer the topology of the environment from the stored network and (2) to be able to use visual information in order to navigate between neighboring locations. The information within the map is useless unless it can be used for navigation as well as recognition. With the CLAM network navigation between neighboring locations is possible because these locations share a certain number of covisible features. (Hence the need for some invariance in feature recognition - the AMR needs to be able to recognize features over a range of viewing distances and orientations, in order that it can recognize that two neighboring locations both share the same features).

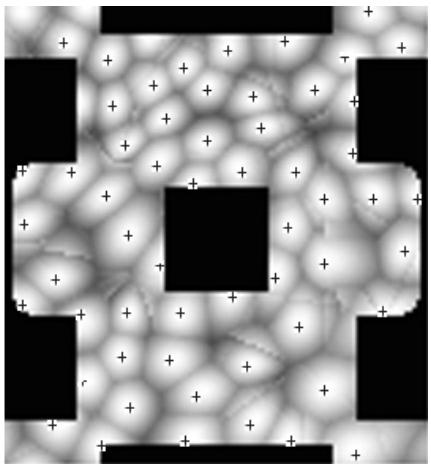

Figure 9. Place field like response of the Similarity Measure for a simple planar environment. Responses tail off in a Gaussian fashion with distance from the most similar known location. Crosses mark the positions of learnt locations.

9. Further Software Simulation

A more sophisticated software simulation was produced in order to examine practical implementation considerations. Unfortunately results from the cornerwise histogram experiments suggested that these would be unsuitable to use in such simulation: although we could have used a robot with real input images, or we could have simulated a full 3D environment, it would have been impractical to use the cornerwise features and histograms, due to time and memory limitations. In later experiments the response of the network to cornerwise features will be simulated for an AMR in an unknown environment, but for now an alternative feature representation was used.

An AMR was simulated in a 2D planar environment, with a pair of stereo line scan cameras with which to view its environment. It was our aim to keep the visual input as simple as possible, whilst still providing enough visual information for the AMR to be able to recognize its location and move from place to place. To do this the robot was given a (simulated) pair of line scan camera's with which to view its world. These had a pixel resolution of 512 pixels, over an viewing angle of about 90'. This produces an input image similar to that available with

commercial mini-robots such as the Kephera[1]. At all stages the robot was simulated to be as realistic as possible, such that the same software could be used later to control a real-world robot. For example the software was carefully constructed so that the AMR had no access to information that it would not ordinarily have: all of the information about its environment it obtained from its visual inputs. Furthermore these inputs were carefully modeled so as to include Gaussian noise and random impulse noise, similar to that which would be produced by a real camera system. Similarly any odometry measurements made by the simulated AMR were subject to error, as they would be in a real robot. The robot simulated in these experiments was holonomic, which was a necessary requirement as the AMR had to stop and turn on the spot at each location in order to compare what was visible to known locations.

At this stage the robot explores using random movements! Exploration algorithms are not the subject of this work. It was only necessary that the AMR navigated somehow around its environment. The architecture is such that it can be implemented entirely separately from any navigation algorithm - in order to learn the environment the AMR simply needs access to the visual input. Navigation controls and algorithms such as wall following can be implemented in parallel to this algorithm. (See [23]).

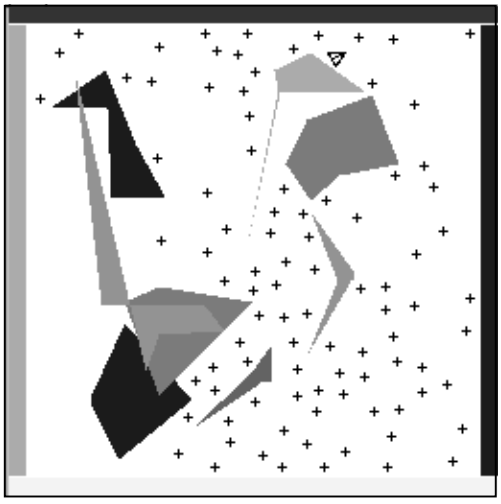

Figure 10. A s/w AMR in the process of learning a large, planar environment populated with randomly generated polygons. Crosses mark the center of visually different locations.

The simplified form of the cornerwise histograms used in the simulation simply encodes 1D edge information as observed by the robot. (This produces a useful representation, but one which is much more brittle than the cornerwise representation - many more locations will be needed to encode an environment than would be needed with cornerwise histograms.)

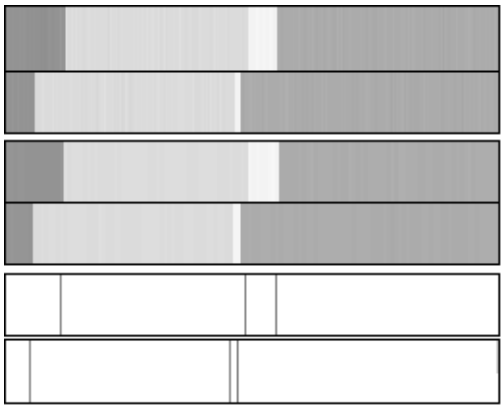

Figure 11. (Top-bottom)Line scan inputs to the AMR left and right cameras, followed by noise-reduced inputs (left then right), followed by extracted edges (left then right).

10. Results

Results from this simulation show that **t**he environment *can* be partitioned into an 'optimal' number of locations; typically a single neuron will fire for each location in the environment. This is ideal for unambiguous recognition, but is unsuitable for navigation purposes. For the AMR to be able to navigate between different locations each point in the world must be mapped by more than one location neuron. Thus the robot can navigate by moving so that the response from one neuron increases as the response from the other drops off. See figure 12 below.

In reality navigation using neuron responses turns out to be difficult. There are ambiguities in the derived navigation control directions due to the nature of the feature representations: namely, that they are symmetrical, and that whilst they store scale information they do not store distance information. A further complication is that the feature lists (the first layer in the network) are continually updated - no information is stored as to *where* the AMR was in the environment when the representations were first observed. Hence trying to navigate using the response of these representations is impossible.

[1] See the internet site at http://lamiwww.epfl.ch/lami/robots/K-family/Khepera.html

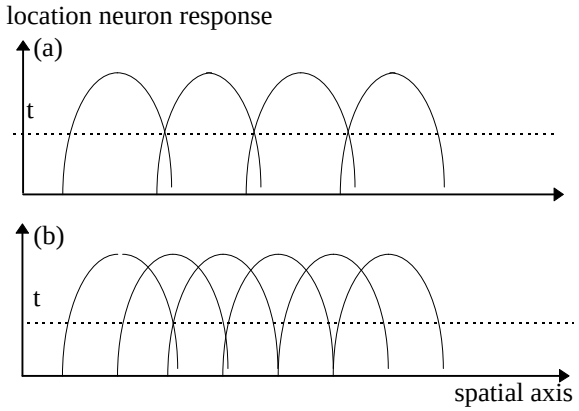

Figure 12. Learning locations using the CLAM network. (a) Optimum, minimum number of locations needed to learn the whole of the space. A neuron fires for each location in the space. (b) Learning requirement for navigation. Points in the space are covered by more than one neuron. ('t' is the recognition threshold).

11. Forming Internal Representations of Local Space

This navigation problem is overcome by storing a limited amount of distance and position information. This is does *not* mean that local structures are modeled - only a limited amount of distance information is needed to enable the AMR to navigate between neighboring locations. Indeed there is evidence that rats brains store similar information[9]. Distance information is derived from the stereo visual system. Although only a single camera is needed to learn an environment, some distance information is needed in order that the AMR is able to navigate from place to place within that environment. This distance information could be supplied from a single camera using parallax measurements etc., but for this work it was easier to use stereo.

For each novel location learnt, (approximate) distance information and angular position for each feature is stored. Here the internal odometry is used for the first time: angular rotation of the AMR needs to be known so that all the features visible for a single location can be positioned correctly with respect to each other. Angular measurements are all relative to some arbitrary coordinate system, which may be entirely misaligned between different locations - it is the angular separation between features which is needed, not their polar coordinates.

Using this extra information the robot can form a robust estimate of the position of any known location with respect to its current location and its current coordinate system, based on the features it can see now (from its current location). This enables navigation between neighboring locations, and it enables the topology of the environment to be inferred. As no direct topological information is stored when the map is constructed, this enables paths to be found through learnt locations, which have never been traveled before by the robot.

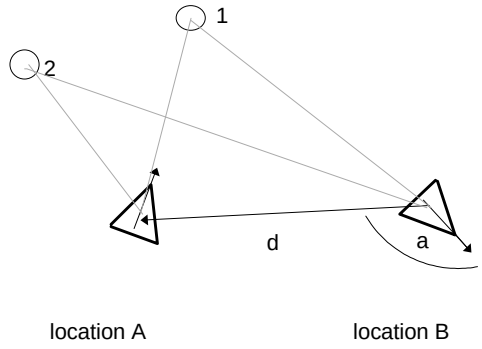

Figure 13. Internal representations of the polar coordinate and distance from the current location to a previously visited location can be calculated from distance and angle information stored for features recognized from both locations.

Of course, introducing a new source of information - a new set of measurements, introduces a new set of errors and these have to be taken into account when forming the internal representation. This error propagation proves to be difficult for the calculation involved; (see figure 13) there are non-Gaussian errors in the distance measurements to recognized features, and Gaussian errors in the angular measurements. Transforming distances to a scale invariant domain and using Monte Carlo simulations shows that the size of the errors in this calculation depend critically on the geometrical positioning of the observed features. It is a non-trivial matter to calculate these errors analytically, and in the s/w simulations an 'on-the-fly' Monte Carlo calculation was performed to determine the errors for every feature pair.

An important transferable result from this calculation is that errors *must* be modeled correctly, or noisy results will *dominate* the plots. Measurements must be either modeled well or a Gaussian distribution should be used which encompasses any real error margins under all conceivable circumstances. This latter prevents subsequent processes from making full use of all the information available.

Due to the PHT summation used, the internal representation formed is a more robust form of the Similarity Measure. It could, in theory could be used in place of the existing measure, though it is a great deal more computationally intense. Maintaining a continuous internal representation of how the robots current location

relates to similar known (typically neighboring) locations would be extremely advantageous for exploration and navigation as it would allow the AMR to explore outside of 'known space' and to explore such that it could always find its way back to places it knows. It is unfeasibly slow to do this with a general purpose machine. (It takes several minutes to produce a single plot such as the one shown in figure 14 on a Sun Sparc).

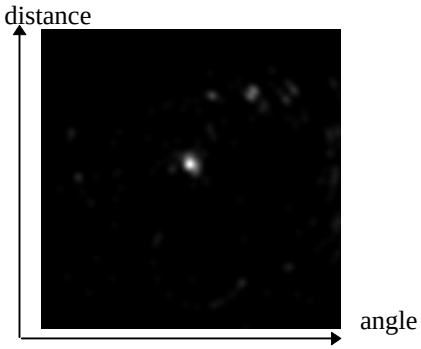

Figure 14. a robust internal representation of the position of a nearby location can be formed from all covisible feature pairs.

12. Global Path Planning

It is a reasonably trivial matter to calculate longer paths between known locations; it is possible to calculate which locations are neighboring using either the Similarity Measure or the internal representation method. In the software simulation 'distances' between all locations were calculated using a robust measure of the feature overlaps between locations. This was not a measure of Euclidean distance, but can be considered as a measure of visual similarity. Neighboring locations share a higher degree of visual similarity than more distant locations. These 'distance' measurements were used in a simple recursive path planning algorithm to calculate navigable paths between distance known locations. See figure 15. Furthermore paths were weighted so as to pass through known locations in preference to passing between locations, to improve the ability of the AMR to follow these paths.

Paths formed by this method are navigable, however they are non-optimal because they are based upon visual similarity, and not physical location. A better result would be obtained by forming internal representations of the positions of each location with respect to every other location, from which relative (Euclidean) distance measurements could be made.

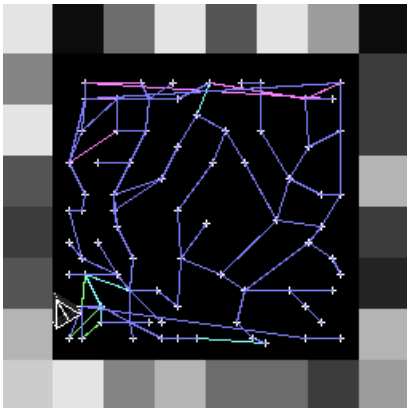

Figure 15. Paths planned through visually different locations in a simple simulated environment.

15.Conclusions

This paper provides initial results from research into to a previously unsolved problem: that of autonomous, robust navigation through an unknown arbitrary environment. A probabilistic framework is developed and used to ensure robustness in the final solution. In particular a geometric representation of corner and edge features is formed which is found to provide the required discriminability and invariance needed by a solution of the problem. These cornerwise histograms were carefully designed to conform to the requirements of the network used and to provide invariances to feature viewing conditions.

Software prototypes were developed to test the algorithms and it was found that realistic simulated robots could reliably learn arbitrary environments, robust to noise on their input senses. The environments could be successfully segmented into visually different locations. The robot was then able to plan a path between any known locations and to navigate along that path successfully. Minimum use was made of internal odometry measurements to further improve the robustness. The robot was able to locate itself both with respect to the known locations using the CLAM network, and within locations by forming a probabilistic interpretation of its position with respect to known features.

It should be noted that because the CLAM network requires only a suitably encoded input, it is be possible to add further supplementary information from any sensor system to the network to back up visual information used. *Any* information which suitably modeled can be used. (e.g. sonar, color information etc.)

Current calculations indicate that the cornerwise representation has the capacity to store around 20,000 distinguishable, real-world features. As between 30-50 corner features are typically observable in a given

location, this suggests that a large number of locations can be learnt. More recent experiments indicate that around 10^5 individually disciminable locations can be learnt. For a physical robot, however, it is likely that the environment size will be limited more by hardware constraints than by the representations used, as corner and edge extraction and representation comparison are computationally intensive processes.

Specialist currently available hardware of interest includes the matrix accumulator chip of C.Brown[24]. This is capable of comparing 175,000 feature histograms per second. In the software simulations it was the continual comparison of observed features to all known features which was the bottleneck. So whilst the algorithm is computationally intensive it is not outside the bounds of current h/w solutions.

16. References

[1] David J. Braunegg "MARVEL: A System That Recognises World Locations with Stereo-Vision". IEEE Transactions on Robotics and Automation, vol9, No.3 June 1994.

[2] Alberto Elfes "Using Occupancy Grids for Mobile Robot Perception and Navigation". Computer, June 1989.

[3] T.C.Hu, Andrew B. Khang, Gabrial Robins "Optimal Robust Path Planning in General Environments" IEEE Transactions on Robotics and Automation, Vol9. No.6, Dec. 1993.

[4] Ulrich Nehmzow "Animal and Robot Navigation" Department of Computer Science, Manchester University, UK. Contact u.nehmzow@cs.man.ac.uk.

[5] Ivan A. Bachelder, Allen M. Waxman, Michael Seibert and Alan N. Gove "From Learning Objects to Learning Environments: Biological and Computational Neural Systems". MIT Lincoln Laboratory , Lexingtion, MA.

[6] R.A.Brooks "Visual Map-Making for a Mobile Robot". In 'Readings in Computer Vision', M.A.Fischer and O. Firschein, pp 438-443, Los Altos, CA.

[7] David Kortenkamp, Terry Weymouth, Eric Chown and Stephen Kaplan " A Scene-Based, Multilevel Representation for Mobile Robot Spatial Mapping and Navigation". IEEE Transactions on Robotics and Automation Nov 1991.

[8] Michael Seibert, Allan M. Waxman "Adaptive 3D Object Recognition from Multiple Views" IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.14, No.2, Feb 1992.

[9] J.O'Keefe, L.Nadel "The Hippocampus As A Cognitive Map", Clarendon Press, Oxford, 1978.

[10] Thomas M.Breuel "View-Based Recognition" IDIAP Memo #93-09 July 1993. Available from http://eee.idiap.ch/idiap-trs.html. IDIAP, CP 609, 1920 Martigny, Switzerland.

[11] N.K.Logothetis, J.Pauls, T.Poggio "View-Centred Object Recognition in Monkeys" MIT AIL & CBCL Memo # 1473, April 1994.

[12] Ivan A. Bachelder , Allen M.Waxman "Mobile Robot Visual Mapping and Localisation: A View-Based NeuroComputational Architecture That Emulates Hippocampal Place Learning". Neural Networks, Vol.7, Nos 6/7, pp 1083-1099, 1994.

[13] John Canny "A Computational Approach to Edge Detection". IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI1-8, No. 6, November 1986.

[14]Chris Harris, Mike Stephens "A Combined Corner and Edge Detector". (c) The Plessey Company plc., Plessey Research, Roke Manor, United Kingdom. 1988.

[15] Neil Thacker, J.E.W. Mayhew, "Designing A Layered Network For Context Sensitive Pattern Classification". Neural Networks, Vol 3, pp291-299, 1990.

[16] Michael Tarr, Heinrich H.Bulthoff "Is Human Object Recognition Better Described By Geon-Structural-Descriptions of by Multiple Views?". Available from ftp://ftp.mpik-tueb.mpg.de/pub/mpi-memos/cogsci-3.ps.Z

[17] P.V.C. Hough "Method and Means for Recognizing Complex Patterns", US Patent 3069654.

[18] Neil Tacker, Frank Aherne, P.I.Rockett "The Bhattacharyya Metric As An Absolute Similarity Measure For Frequency Coded Data". submitted to Pattern Recognition.

[19] Anthony Ashbrook " Pairwise Geometric Histograms for Object Recognition: Developments and Analysis", Ph.D. Thesis, University of Sheffield 1996.

[20] Neil A. Thacker , J.E.W.Mayhew "Designing A Layered Network For Context Sensitive Pattern Classification", Neural Networks, Vol 3, pp291-299, 1990.

[21] A. Ashbrook, P.I.Rockett "Robust Recognition of Scaled Shape Using Pairwise Geometric Histograms". Proceedings of the BMVC 95 Conference. Vol 2., pp 503-512.

[22] A. Bhattacharyya "On a Measure of Divergence Between Two Statistical Populations Defined by Their Probability Distributions". Bulletin of the Calcutta Mathematical Society, #35. pp 99-110 1943.

[23] R.A.Brooks "A Robust Layered Control System For A Mobile Robot". IEEE Journal of Robotics and Automation, Vol RA-2, No.1, March 1986.

[24] C. Brown "A VLSI Device for Multiplication of High Order Sparse Matrices" Ph.D. Thesis, University of Sheffield 1996.