Proceedings of the 2006 IEEE/RSJ
International Conference on Intelligent Robots and Systems
October 9 - 15, 2006, Beijing, China

Topological Mapping Inspired by Techniques in DNA Sequence Alignment

Alan M. Zhang, Lindsay Kleeman and R. Andrew Russell
Intelligent Robotics Research Centre
Department of Electrical and Computer Systems Engineering
Monash University, Clayton, Victoria 3800, Australia
{alan.zhang, lindsay.kleeman, andy.russell}@eng.monash.edu.au

***Abstract*— This paper introduces a method of building topological maps using sequences of images and the approximate string matching algorithm, which is commonly used in DNA sequence alignment applications. Contrary to many existing dense image based topological localisation techniques that operate in known maps, our method builds topological maps in unknown environments. And unlike traditional topological mapping methods that require the robot to explicitly recognise topological path junctions during exploration, our method requires no such explicit "junction detectors". It receives as input only a sequence of unlabeled images. The validity of the approach has been demonstrated in both indoor and outdoor environments. The largest outdoor map created measures 70 by 40 meters with 3 nested loops. The system has shown robustness towards large amounts of sensor aliasing and noise caused by errors in the path following behaviours.**

I. Introduction

Existing mobile robot mapping paradigms can be divided into metric, topological or a combination of the two. The topological paradigm has the advantage that it only needs to recover the topology, not to reconstruct the geometry of the environment. Therefore, it has the potential to be the more flexible paradigm. This paper presents a topological mapping approach that takes as input a sequence of unlabeled images captured at regular intervals as the robot moves, and recovers the topology of the environment. To understand the innovations of the presented approach over existing ones, we first examine the common assumptions made by topological mapping systems. The novelty of our approach comes from the *relaxation* or *removal* of some of these assumptions. A general condition that is typically assumed for topological mapping is the existence of a set of behaviours that constrain the movement of the robot to only one degree of freedom. For example, a road following behaviour when there is a footpath and a wall following behaviour in the corridors or in an open room. This assumption is also made in this paper. When a robot executes such behaviours, the trajectory it follows is referred to as a path. These behaviours can be collectively called path following behaviours. Paths intersect each other at path junctions. Existing topological mapping systems typically require some mechanism that is functionally equivalent to a "junction detector" which signals when the robot has reached a path junction [1]. However, these path junctions can be difficult to define, especially in outdoor environments. In our approach, no such "junction detector" is assumed. As far as the mapping system is concerned, the experience of the environment consists entirely of a temporal sequence of images. The state of the art in image matching has shown some quite robust results. However, no method is completely reliable. So depending on the environment, a certain amount of sensor aliasing will occur. In this paper, a pessimistic view is adopted where a large amount of aliasing is assumed. The assumptions are summarised below:

Assumptions:

- Existence of movement behaviours that constrain the location of the robot onto paths.

Common assumptions that are relaxed or removed:

- No "junction detector" is required. Path junctions will be inferred from the observations.
- A large amount of sensor aliasing is allowed.

Under these assumptions the problem of building a topological map can be illustrated using Fig. 1. The robot starts off at position 1, executes its path following behaviour, visits positions 2-5 and stopping at 6. It captures images at regular intervals along the way. The mapping system is then presented with this sequence of images. During sections 2-3 and 4-5 the robot goes through the same part of the environment thus having the same sequence of perceptions. Positions 2, 3, 4 and 5 are path junctions but are not indicated as such to the mapping system. The mapping algorithm needs to discover that the subsequence of images captured in section 2-3 is very similar to the subsequence captured in section 4-5. If the subsequences are long enough and similar enough, then there is a high confidence that they are the same part of the environment. Positions 2 and 4 can then be inferred to be a path junction without the need to be explicitly detected using a "junction detector". The same applies to positions 3 and 5. The method used in this paper to discover the matching subsequences is the well known local alignment approximate string matching algorithm. It has been used extensively for DNA/RNA sequence alignment analysis. It is so widely used in the field of computational molecular biology that it is usually simply referred to as "the dynamic programming algorithm". Here we adapt it to operate on a sequence of images.

This paper is organised as follows: Section II discusses related research in the area; Section III describes the hardware

1-4244-0259-X/06/$20.00 ©2006 IEEE

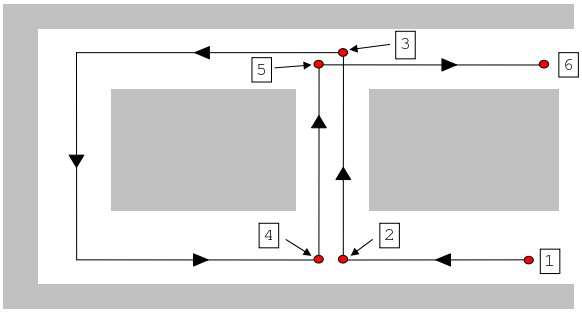


Fig. 1. Topological mapping scenario considered in this paper. The movement of the robot is constrained by path following behaviours. The robot moves from position 1 through 6. Images are captured at regular intervals.

setup; the components of the map building system is presented in Section IV; experimental results demonstrating the validity of the approach are presented in Section V; and conclusions are drawn in Section VI.

II. Related Research

Our approach can be regarded as topological map building using the dense image matching paradigm. The majority of literature in this area are concerned with topological *localisation*. Some of the more recent works include [2], [3], [4], [5], just to name a few. These works are aimed at improving the robustness of the image matching methods for localisation in known maps. Quite often the ground truth locations of where the training images were taken is known. During operation, the robot is localised to the training image position that most closely matches the current perception. On the other hand, our approach is about *building* a topological map in an *unknown* environment.

The system in [6] closes loops in topological maps using localisation. The idea is to continually try to localise the robot in the part of the map that has been built. If the localisation confidence is low then new observations are added as new parts of the map. If the confidence is high then a loop is detected. A procedure then finds where the loop was first closed by back tracking to the position when the confidence first started to increase. This heuristic is not guaranteed to produce the most likely map.

The map building technique in [7] is to modeled the map as a Partially Observable Markov Decision Process (POMDP) and trained using the expectation-maximization technique. The difficulty with map representation using Hidden Markov Model (HMM) or POMDP is in deciding the topology and number of states. Generally these can not be determined *a priori* for an unknown environment. The typical work around is to discretise the environment with a grid, such that both the topology and number of states can be specified *a priori*. This limits the flexibility of the HMM/POMDP model and does not scale well to large environments. The approach in this paper *determines* the topology of the map by looking for sufficiently long sequences of matching images.

The approaches of [8] and [9] are closest to ours. However, both perform localisation inside a known map instead of mapping the unknown environment.

III. Hardware

The hardware setup consists of a Pioneer 3DX robot for mobility, a webcam/panoramic-mirror assembly mounted atop a mast at a height of 1.5m for panoramic imaging, and a laptop for robot control and data recording. The panoramic-mirror is of the type as described in [10] with a gain of 7. It is suspended above the camera using a perspex cylinder. Wheel encoders provide odometry measurements.

IV. Topological Mapping Algorithm

During the data collection phase, the robot follows paths and captures a sequence of images at regular intervals. These images are then used as input to the mapping system. The sequence of images is treated as a string where each image is a character. A string matching algorithm finds matching subsequences in the string. To facilitate string matching, the images are first compared pairwise to generate a similarity matrix. After matching subsequences are found, a labeling process completes the building of a topologically correct map. But to more easily visualise the topological map, odometry is incorporated as a last step. Image comparison and similarity matrix generation is presented in Section IV-A. Section IV-B describes the string matching algorithm. Label assignment and incorporation of odometry for visualisation are presented in Sections IV-C and IV-D.

A. Similarity Distance and Similarity Matrix

In this work images are compared using joint histograms [11], [12]. This is a very poor image comparison method. Many other methods achieve better discrimination and are more tolerance to illumination variations and partial occlusions, eg. [13], [14]. But the histogram method was deliberately chosen *because* it is poor, in order to demonstrate the ability of the system in tolerating large amount of sensor aliasing.

The comparison method calculates a number of local features for each pixel: *edge density*, *gradient magnitude*, *texturedness* and *rank*, together with the colour channels R, G and B to form a 7 dimensional feature vector. Refer to [11] for details on the local features. A 7D histogram is then constructed for each image. The number of bins in each dimension are as suggested in [11]: 4 for edge density, 5 for edge magnitude, 4 for texturedness, 4 for rank and 4 for each of the colour channels. The χ^2 bin-by-bin distance measure shown below is used to compare image histograms:

$$D(I,J) = \sum \frac{(f_I(k) - f_J(k))^2}{f_I(k) + f_J(k)} \qquad (1)$$

where $D(I,J)$ is the distance measure between images I and J, and $f_I(k)$ and $f_J(k)$ are the corresponding histogram entries. This distance measure is then converted in to a *similarity* measure via a simple linear transformation:

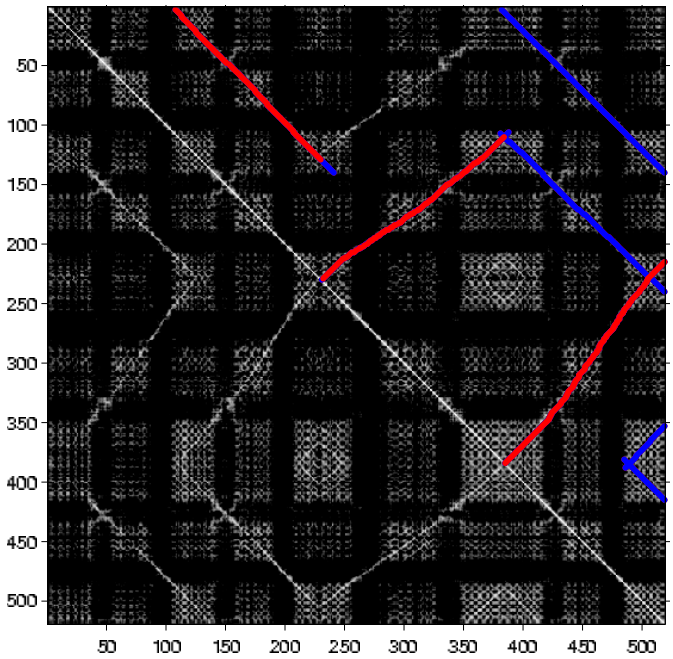


Fig. 2. Similarity matrix, with detected and pruned matching subsequences superimposed. Matching pairs of images are shown as white pixels. The red and blue lines are the detected matching subsequences. Only the red lines remain after pruning and are subsequently used to construct the topological map.

$S(I,J) = 1 - \kappa * D(I,J)$, where κ is set experimentally. A similarity matrix is then constructed by calculating $S(I,J)$ for every pair of images. Because (1) is symmetric, i.e. $D(I,J) = D(J,I)$, the similarity matrix is also symmetric. A similarity matrix calculated from real data is shown in Fig. 2, where brighter cells represent pairs of images with higher similarity. This dataset was gathered in an office environment. Note the large amount of sensor aliasing.

Using this measure simulates the worst case performance. Any improvement in image comparison can only increase the performance of the system. Although in this paper vision is used as the only sensor, any other type of sensor can be used to generate the similarity matrix, so long as there exists a method to compare pairs of readings.

B. Local Alignment Approximate String Matching

As mentioned previously the sequence of images can be treated as a string. When the robot revisits the same path the sequence of perceptions will be repeated. This means there will be repeated substrings if the robot revisits the same path. But because of noise, the repeat will not be exact. Such approximately matching substrings can be discovered robustly using the well known and widely applied local alignment approximate string matching algorithm [15], [16]. In the subsequent sections "sequences of images" and "strings" are used interchangably depending on context. By the same token characters refer to individual images. The string matching algorithm is described below.

The notation used here follows the convention in [15].

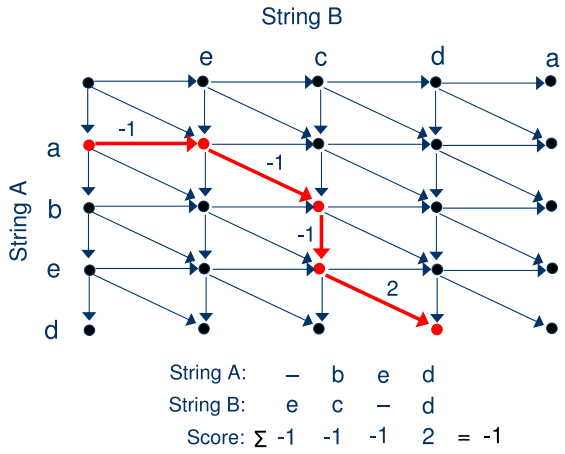


Fig. 3. Weighted grid graph for matching strings A and B. A weight is given to each edge but only relevant weights are shown. Each path in the graph encodes an alignment between strings. The alignment generated by traversing the red path is shown, along with the alignment's score.

Suppose that two strings A and B are to be matched. In the topological mapping context, A and B are both assigned the same input image sequence. But for the sake of clarity the labels A and B are kept. Their lengths are $|A| = m$ and $|B| = n$. The string matching problem is to find an optimal alignment (pairing of characters in the two strings) between a substring A' of $A = A_1 \ldots A_m$ and B' of $B = B_1 \ldots B_n$. This problem can be modeled using a weighted grid graph. An (m,n) weighted grid graph $G = (V,E)$ is a directed (acyclic) weighted graph that contains $(m+1) \times (n+1)$ vertices with rows $0 \ldots m$ and columns $0 \ldots n$. Vertex (i,j) has a directed edge to $(i+1,j)$, $(i+1,j+1)$, and $(i,j+1)$, except when the endpoints are outside the boundaries of the grid. Fig. 3 shows such a weighted grid graph for matching strings $A = \{a,b,e,d\}$ and $B = \{e,c,d,a\}$.

An alignment of $A_i^c = A_{i+1} \ldots A_c$ with $B_{i'}^{c'} = B_{i'+1} \ldots B_{c'}$ then corresponds to a path from (i,i') to (c,c') in the graph. Following a horizontal edge corresponds to skipping over a character in B and incurs a penalty that is reflected in a negative weighting given to horizontal edges. A vertical edge represents skipping over a character in A and also incurs a penalty. The diagonal edge $\langle (k,l),(k+1,l+1) \rangle$ corresponds to matching characters A_{k+1} and B_{l+1} and the edge weight is positive if the characters match, negative otherwise. Fig. 3 shows the character alignment represented by the red path, along with the score of that alignment. Skipping a character is represented in the alignment by matching that character to a "gap", denoted by a dash. Therefore a good alignment of A_i^c with $B_{i'}^{c'}$ is indicated by the high score of its corresponding path in the grid graph. Let $score(A_j^r, B_l^t)$ denote the score of the best alignment between substrings $A_j^r = A_{j+1} \ldots A_r$ and $B_l^t = B_{l+1} \ldots B_t$. The best such scores can be found using a dynamic programming approach. Each entry in the dynamic programming table T corresponds to a vertex in

the grid graph. The cell $T[r,t]$ would hold the maximum in $\{score(A_j^r, B_l^t) \mid j \in [0,r], l \in [0,t]\}$. With relation to grid graphs, $T[r,t]$ holds the highest score of all possible paths that end at the vertex (r,t). Because a maximum of only 3 edges go into a vertex, any path that ends at vertex (r,t) will have to go through one of its 3 neighbours. Therefore, $T[r,t]$ can be calculated from $T[r-1,t]$, $T[r-1,t-1]$ and $T[r,t-1]$. Hence the DP table T can be filled with a raster scan. The initialisation and recurrence relationships are shown below:

Initialisation:

$$T[r,0] = 0, \quad r \in [0,m] \tag{2}$$

$$T[0,t] = 0, \quad t \in [0,n] \tag{3}$$

Recurrence:

$$T[r,t] = max \left\{ \begin{array}{l} 0, \\ T[r-1,t-1] + S(A_r, B_t), \\ T[r-1,t] + w(A_r, -), \\ T[r,t-1] + w(-, B_t) \end{array} \right\} \begin{array}{l} r \in [1,m] \\ t \in [1,n] \end{array} \tag{4}$$

where $S(A_r, B_t)$ is the similarity measure calculated in Section IV-A that corresponds to matching characters A_r and B_t. So if the images match, $S(A_r, B_t)$ will add a positive reward to the cumulative score, otherwise, the score is added with a negative penalty; $w(A_r, -)$ and $w(-, B_t)$ are the penalties assigned to skipping a character in string A and B respectively, i.e. matching against a gap. The first argument in (4) is zero to ensure that if none of the suffixes of A_0^r and B_0^t can be aligned with a strictly positive score, $T[r,t]$ would be set to $score(A_r^r, B_t^t) = 0$. The scores in T are now a robust measure of similarity between substrings that takes into account alignment errors like skipped or mismatched characters. By placing a suitable threshold on the scores it is possible to determine whether the robot is *revisiting* a part of the environment, i.e. closing the loop.

A modification is made to the standard algorithm described above, in which an upper bound is imposed on the cumulative score. If the cumulative score is allowed to grow unbounded, then two pairs of very high scoring substrings that are separated by a wide non-matching segment would be joined into one pair of matching substrings. This occurs because even though the non-matching segment provides enough evidence to indicate that the robot is exploring a new part of the environment, the high cumulative score "spills over" into the non-matching segment and does not reduce to below 0 when it reaches the other pair of matching substring.

The next step is to determine the beginning and end positions of the matching substrings. These positions can be inferred to be path junctions, thereby removing the need for an explicit "junction detector". Bearing in mind that there are possibly more than one pair of matching substrings (i.e. more than one loop or same part of the environment traversed multiple times), the problem is one of identifying *all* substring pairs that are locally optimal. This problem has been analysed in [17], [18] and also described in [15]. The method presented in [15] has been adopted in this paper. Due to lack of space, please refer to [15] for the details of the algorithm. Gusfield [16] also presents excellent discussions on the subject of string matching.

The substrings that match in the reverse order can be discovered by appropriate modifications to the dynamic programming recurrence relationships and running the above algorithm on the strings in a second pass.

Fig. 2 illustrates the result of string matching. Each blue or red line segment is a locally optimal path through the grid graph representing a pair of locally optimal matching substrings. The lines in the direction of top-right to bottom-left are substrings matched in reverse order, i.e. the robot is traversing the same path but in the reverse direction.

C. Pruning and Dense Label Assignment

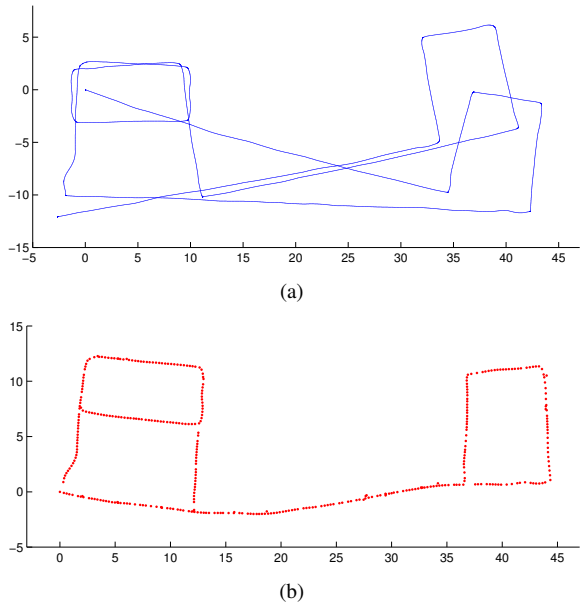
Having identified the matching subsequences, we are ready to infer the topological map. The approach taken is to assign *each* image a label. Later in Section IV-D, these labels help to better visualise the data. Given a pair of matching substrings, the unique pairing of images within the substrings is provided by the string matching algorithm. Lets denote these pairings using their image indices: $\{1,10\}$, $\{3,11\}$, $\{11,25\}$, $\{25,45\}$, etc. Where $\{1,10\}$ means the 1st image is matched with the 10th image in the sequence. We take a rather simple approach to labeling by assuming that the relationships in the aligned pairs are transitive. So in the example sequence, images 3, 11, 25 and 45 are all assigned the same label. However, the problem that arises is inconsistency caused by noise. Consider the following 3 subsequence alignments, where image indices are paired vertically:

$$\text{Alignment 1: } \left\{ \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ 11 & 12 & 13 & 14 & 15 \end{array} \right\}$$

$$\text{Alignment 2: } \left\{ \begin{array}{ccccc} 11 & 12 & 13 & 14 & 15 \\ 21 & 22 & 23 & 24 & 25 \end{array} \right\}$$

$$\text{Alignment 3: } \left\{ \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ 22 & 23 & 24 & 25 & 26 \end{array} \right\}$$

From these alignments we can infer that 5 = 15, 15 = 25, 25 = 4, 4 = 14, 14 = 24, 24 = 3 and so on. It means that eventually all images in the subsequences will be assigned the same label. The ideal solution to this problem is to perform what is known as multiple string alignment. The string matching algorithm presented in this paper is a pair-wise method. An extension of this dynamic programming method to handle multiple strings is possible, but a 3 strings alignment requires a 3 dimensional dynamic programming table, and a 4 dimensional table for 4 strings. The computational complexity explodes beyond what is reasonable very quickly. Methods based on heuristics are therefore commonly used instead for multiple alignment. However, more robust methods of resolving inconsistent alignments will be left to future work. In this paper we describe a simple heuristic that resolves these inconsistencies. The heuristic is to prune the set of alignments such that for each column in the dynamic programming table, only the alignment of the longest



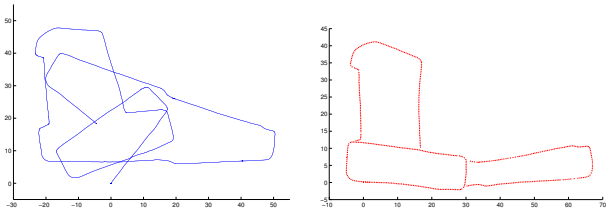
(a) (b)

Fig. 4. Experiment No. 1: Indoors. A total of 869 images captured at 25cm intervals. Units on the axes are in meters. (a) Cumulative odometry. (b) Topological map visualised by incorporating odometry. Red dots are the estimated locations where images were taken.

matching subsequence is retained. The red lines in Fig. 2 represent the matching subsequences that remain after pruning. The blue lines were pruned out. The transitive method of labeling is then applied on the pruned matching subsequences. At this stage a topologically correct map has been created, where the images having the same label come from the same part of the environment. But this purely topological map is difficult for humans to interpret. Odometry information is therefore integrated into the map for better visualisation.

D. Incorporating Odometry for Visualisation

Since each image is allocated a label, the data association step in Simultaneous Localisation and Mapping (SLAM) has already been done. An extended Kalman filter algorithm is used to build a metric representation of the environment. Predictions of the robot's pose come from odometry only. Observable features are the *locations* where images were taken. Since the histogram based image similarity measure does not provide relative bearing estimates, the features are encoded with (x, y) position information only. Because the histogram method does not provide estimates of any other kind of geometric relationship between pairs of images either, the predicted position of an observed image is simply the current estimated position of the robot. Observation errors are assumed to be symmetrical two dimensional Gaussians. The use of odometry with the Kalman filter is merely an aid for a human to check the validity of the topological map in the results and is not intended as an exact geometric representation. So the metric map created may not be geometrically correct.



(a) (b)

Fig. 5. Experiment No. 2: With indoors and outdoors sections. A total of 800 images captured at 50cm intervals. Robot traveled a total of 404 meters. Units on the axes are in meters. (a) Cumulative Odometry (b) Topological map visualised by incorporating odometry

V. Results

Experiments in both indoor and outdoor environments were carried out. Because the path following behaviours have not yet been implemented, the robot was manually controlled with a joystick in all experiments to simulate behaviours such as corridor following. Images and odometry were logged and processed offline.

Experiment No. 1 was performed in an office/corridor environment. The area covered was approximately 45 meters by 12 meters. Images were captures at 25cm intervals. Cumulative odometry is shown in Fig. 4(a). The resulting topological map with odometry incorporated for visualisation is shown in Fig. 4(b). Each red dot represents the estimated location where an image was taken. The robot started from position (0, 0) then moved towards the right of the map along a corridor, looped around a stair well, then traveled back along the same corridor in the opposite direction. In doing so, it experiences the sequence of images in reverse order while traveling back along the corridor. The algorithm successfully identified these reversely matched subsequences thereby closing the loop.

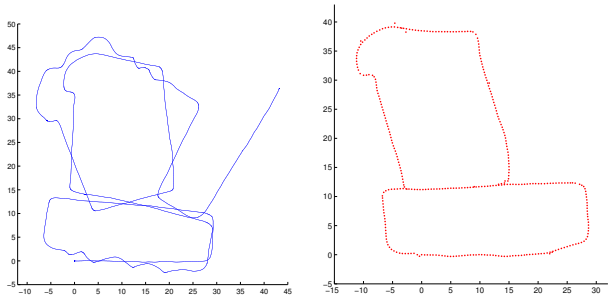
Fig. 5 shows results of the second experiment consisting of both indoor and outdoor sections with 3 nested loops. The top half of the loop on the right side of the map was inside a corridor while the rest of the map were outdoors. Images were taken at 50cm intervals. The map is approximately 70 by 40 meters. The total distance traveled by the robot was 404 meters. All loops were closed successfully.

Experiment No. 3 shown in Fig. 6 demonstrates the system's robustness towards errors in the path following behaviour. In order to simulate poor path following behaviour, the robot was deliberately made to follow a zig-zag pattern during a second traversal of the same part of the environment. The zig-zag pattern meant that the images were captured at different locations to that of the first traversal, hence different in appearance. There were also more images in the zig-zag sequence because images were captured at fixed distance intervals and the zig-zag pattern covers more distance. Nevertheless, the string matching algorithm was robust enough to find matching subsequences. The two loops in this experiment are the same two loops on the left of Fig. 5(b). As shown in Fig. 6(b), the topology of the environment was correctly deduced. Note that the poor odometry information in these experiments resulted in

TABLE I

EXECUTION TIME OF EXPERIMENTS

Experiment No.	No. of Images	Histogram Calculation	Similarity Matrix	Map Construction
1	869	25.2 sec	13.1 sec	105 ms
2	800	23.2 sec	9.7 sec	81 ms
3	787	22.8 sec	10.0 sec	100 ms



(a) (b)

Fig. 6. Experiment No. 3: Outdoor experiment demonstrating tolerance to errors in path following behaviour. The robot was deliberately driven in a zig-zag patter when it traverses the same part of the environment a second time. A total of 787 images were captured at 50cm intervals. Units on the axes are in meters. (a) Cumulative Odometry (b) Topological map visualised by incorporating odometry

deformed metric maps. However, we stress that the aim of the current work is to build topologically correct maps. Odometry is used only for a human to check the validity of the results.

Table I shows the execution time for the experiments. The platform was a PC with a 2.6GHz Pentium 4 CPU and 512MB of RAM. All images were 320 by 240 pixels. The fourth column shows the time it took to generate the similarity matrix. The 'Map Construction' column is the total combined time of string matching, pruning and dense label assignment. It is evident from the table that image comparison accounted for most of the computation. The mapping algorithm itself is very efficient.

VI. CONCLUSION

A novel approach to building topological maps in unknown environments has been proposed. The approach requires only an unlabeled sequence of images taken at regular intervals as the robot travels. To achieve loop closing it uses an approximate string matching algorithm to robustly discover repeated subsequences in the entire sequence of images. Our method has the potential of facilitating the fusion of an arbituary set of sensors of very different sensing abilities, as long as there exists a similarity measure between sets of sensor readings. Its feasibility has been demonstrated with data gathered by a mobile robot with a panoramic vision system. The largest outdoor map created was 70 by 40 meters with 3 nested loops. The system has shown robustness towards large amounts of sensor aliasing and noise due to errors in path following behaviours.

ACKNOWLEDGMENT

The authors would like to thank the ARC Centre for Perceptive and Intelligent Machines in Complex Environments (pimce.edu.au) for their financial support.

REFERENCES

[1] B. Kuipers and Y.-T. Byun, "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations," *Journal of Robotics and Autonomous Systems*, vol. 8, pp. 47–63, 1991.

[2] M. Artac, M. Jogan, and A. Leonardis, "Mobile robot localization using an incremental eigenspace model," in *2002 IEEE International Conference on Robotics and Automation*, Washington, DC, May, 2002.

[3] H. Andreasson and T. Duckett, "Topological localization for mobile robots using omni-directional vision and local features," in *5th IFAC Symposium on Intelligent Autonomous Vehicles*, Lisbon, Portugal, 2004.

[4] J. Wolf, W. Burgard, and H. Burkhardt, "Robust vision-based localization for mobile robots using an image retrieval system based on invariant features," in *2002 IEEE International Conference on Robotics and Automation*, Washington, DC, May, 2002.

[5] D. M. Bradley, R. Patel, N. Vandapel, and S. Thayer, "Real-time image-based topological localization in large outdoor environments," in *Proceedings of the International Conference on Intelligent Robots and Systems*, 2005.

[6] N. Tomatis, I. Nourbakhsh, and R. Siegwart, "Hybrid simultaneous localization and map building: a natural integration of topological and metric," in *IEEE International Conference on Robotics and Automation ICRA'02*, 2002.

[7] S. Koenig, R. Goodwin, and R. Simmons, "Robot navigation with markov models: A framework for path planning and learning with limited computational resources," in *International Workshop, Reasoning with Uncertainty in Robotics*, Amsterdam, Netherlands, 1995.

[8] T. Nishimura, S. Nozaki, and R. Oka, "Spotting-based global positioning with non-monotonic continuous DP for mobile robots using image sequences," in *Proc. IEEE International Conference on Intelligent Robots and Systems*, 1999.

[9] J. Y. Zheng and S. Tsuji, "Panoramic representation of scenes for route understanding," in *Proceesings of the 10th International Conference on Pattern Recognition*, vol. 1, 1990, pp. 161–167.

[10] J. S. Chahl and M. V. Srinivasan, "Panoramic vision system for imaging, ranging and navigation in three dimensions," in *Proceedings of the International Conference on Field and Service Robotics FSA'99*, Pennsylvania, USA, 1999.

[11] G. Pass and R. Zabih, "Comparing images using joint histograms," *ACM Journal of Multimedia Systems*, vol. 7, no. 3, pp. 234–240, 1999.

[12] C. Zhou, Y. Wei, and T. Tan, "Mobile robot self-localization based on global visual appearance features," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'03)*, Taipei, Taiwan, 2003.

[13] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision*, September, 1999.

[14] C. Schmid and R. Mohr, "Local grayvalue invariants for image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp. 530–535, 1997. [Online]. Available: citeseer.ist.psu.edu/schmid97local.html

[15] J. P. Schmidt, "All highest scoring paths in weighted grid graphs and their application to finding all approximate repeats in strings," *SIAM J. Comput.*, vol. 27, no. 4, pp. 972–992, 1998.

[16] D. Gusfield, *Algorithms on Strings, Trees, and Sequences. Computer Science and Computational Biology*. New York, NY, USA: Cambridge University Press, 1997.

[17] B. W. Erickson and P. H. Sellers, "Recognition of patterns in genetic sequences," in *Time Warps, String Edits, and Macromolecules: The theory and Practice of Sequence Comparison*, D. Sankoff and J. B. Kruskal, Eds. Reading, MA: Addison-Wesley, 1983, pp. 55–91.

[18] M. S. Waterman and M. Eggert, "A new algorithm for best subsequence alignment with application to tRNA-rRNA comparisons," *Journal of Molecular Biology*, vol. 197, pp. 723–728, 1987.