Mapping Repetitive Structural Tunnel Environments for a Biologically Inspired Climbing Robot

Gavin Paul[1], Shuyuan Mao[2], Liyang Liu[3] and Rong Xiong[4]
Authors [1,3] are with the CAS, University of Technology Sydney and Author[2,4] are with CSC at Zhejiang University{[1]*gavin.paul*, [3]*liyang.liu*} *at uts.edu.au*, {[1]*maoshuyuan*, [4]*rxiong*} *at iipc.zju.edu.cn*

This paper presents an approach to using noisy and incomplete depth-camera datasets to detect reliable surface features for use in map construction for a caterpillar-inspired climbing robot. The approach uses a combination of plane extraction, clustering and template matching techniques to infer from the restricted dataset a usable map. This approach has been tested in both laboratory and real-world steel bridge tunnel datasets generated by a climbing robot, with the results showing that the generated maps are accurate enough for use in localisation and step trajectory planning.

Keywords: 3D Mapping; Depth cameras; Tunnel Environments; Biologically Inspired; Climbing Robot

1. INTRODUCTION

Recent advances in robotics and sensor technology increasingly enable evermore complicated tasks to be automated. When a climbing robot[1] must perform tasks in a three dimensional(3D) environment, for example inspecting the health of structures[1–3] a geometric map is crucial for online localisation, motion planning and accurately determining foot placements that avoid certain areas and collisions. Building an accurate map in repetitive structured environment such as a tunnel is challenging due to noisy and incomplete sensor data and the lack of visual features.

Many mobile robot systems that perform 3D geometric mapping and SLAM[4] have been developed using laser range finders and/or depth cameras, combined with scan-matching algorithms to construct 3D volumetric maps. To overcome occlusions, a 3D map can be generated from multiple locations using manipulator-mounted sensor exploration approaches.[5–7] In these cases, data is registered and overlapping parts of several views are fused together. However, a robot manipulator with the flexibility to traverse complicated surfaces, and carry its own adhesion mechanism, has been found to be subject to sag in its links,[1] making sensor data taken from multiple poses more complicated to fuse than for a rigid industrial manipulator. Noisy and incomplete sensor data, combined with inaccuracies in the manipulator model, or flexible joints/links, often mean that even after exploring an environment the geometry cannot be determined with sufficiently high accuracy.

Prior knowledge about structural elements in a workspace has previously been used for an eye-in-hand sensor configuration mounted on a static base to identify and localise these elements.[8] However, our prior work made several assumptions that simplified the problem: a) the exact shape of the templates was known *a priori* so fused point to template plane ICP could be performed, and b) a static base meant template locations could be confined in a limited bounding box. For a mobile climbing robot in a tunnel, where scans can only happen at certain locations, planes must be detected from the noisy data necessarily gathered by looking at surfaces at near-parallel angles (i.e. into the tunnel), and templates need to be created online since tunnel sizes vary at all locations.

This paper presents an approach that combines plane extraction, plane clustering, and template matching techniques so as to infer a smooth tunnel map given incomplete noisy depth datasets. There are two main contributions of this paper. First, we exploit organised point clouds and local surface normal features to implement a region growing algorithm for plane extraction which enhances the speed significantly. Then we combine prior knowledge encoded in a set of variable shaped templates to achieve geometrically accurate mapping of the environment which can be used to localise the robot in the environment. The reliable maps generated are currently being used to map and localise a caterpillar-inspired climbing robot in real-world repetitive structural tunnel environments. The remainder of this paper is organized as follows. Section 2 describes the proposed approach to eye-in-hand map generation, region growing, then template matching. Section 3 presents experimental results using data collected for both simulated and real-world tunnels and discusses the limitations and possible drawbacks to the current approach. Finally, Section 4 provides conclusions.

2. Methodology

Given a depth sensor mounted on a robot's end effector in an "eye-in-hand" configuration, and the nDOF manipulator pose, $\mathbf{q} = [q_1, \dots q_n]^T$, it is possible to compute the position and orientation of the sensor, ${}^0T_s(\mathbf{q})$

using forward kinematics. When the sensor is a Kinect-like depth camera it gives images comprised of a matrix of $M \times N$ depths values, $D = d_{m,n} \forall m \in M, n \in N$. A 3D point is computed using ${}^{0}T_s(\mathbf{q})$ for each image pixel with an index in m, and n, such that in the cameras coordinate frame a $\mathbb{R}^3$ point, $\mathbf{p_{m,n}}$ corresponds to the image's mth collum and nth row.

2.1. *Plane Extraction*

In a repetitive structural tunnel, we know *a priori* that the environment is bounded with four sets of main coplanar plane patches as well as many smaller plates with rivets connecting these. Extracting planes largely eliminates sensor noise, and since the templates required are a set of bounded planes, plane extraction is fundamental to template matching. The plane extraction problem is, given N 3D points, $\{p_i\}_{i=1,\ldots N}$, extract a set of planes, $\Pi_i = \{\mathbf{p_i}, \mathbf{n_i}\}$ for $i = 1, \ldots M$. Each plane detected should consist of a set of data including a point that is within that i'th plane's region, $\mathbf{p_i}$ and a normal, $\mathbf{n_i}$ of i'th plane. The plane model for Π_i is

$$\mathbf{n_i} \cdot (\mathbf{r} - \mathbf{p_i}) = 0. \tag{1}$$

Region growing based methods[9–11] are the most popular approach for plane segmentation since they extract bounded plane patches from point clouds. Our plane growing algorithm is based on[10] and consists of several steps: first, pick three or four adjacent points as a seed plane model and test their neighbor points, if a neighbor point combines then the plane model is still a plane, so grow the plane group and update the plane model. Iterate the test/add step until no point can be added, then a new plane is grown.

The popular coplanar criterion is mean square error (MSE) of plane fitting and can feed into the plane model update. This operation is time consuming since a plane fit is required for each test. We present a novel coplanar judgement and model update which utilises each point's local surface normal. The coplanar criterion used are, the normal angle between local surfaces at point and plane, as well as the point-to-plane distance.

$$\begin{aligned} &\|n^* \cdot (p_{new} - r^*)\| \leq \epsilon \\ &arccos(n_{new}, n^*) \leq \lambda \end{aligned} \tag{2}$$

where ϵ and λ are maximum distance and angle thresholds. The plane model is updated by averaging all points and their normals, thus significantly simplifying region growing computations.

Calculating surface normals at each point is generally time consuming. However, in organised point clouds, the adjacent points in the image are the adjacent points in space, and the order is maintained from the original depth image. We use a fast normal estimate algorithm[12] to compute surface normals using the cross product between two tangential vectors to the local surface and using integral images to rapidly compute tangential vectors.

A plane model update consists of updating $(\mathbf{p}^*, \mathbf{n}^*)$. It is straightforward to prove that the plane centroid is on the optimal plane under MSE. Additionally, optimal plane normals, n^* are usually computed with principal component analysis; however, an approximate plane normal can be computed by averaging all points' normal as

$$(\mathbf{p}^*, \mathbf{n}^*) = \left(\frac{1}{N} \sum_{i=1}^{N} p_i, \frac{1}{N} \sum_{i=1}^{N} n_i \right) \tag{3}$$

then adopting a iterative model update,

$$\begin{aligned} \mathbf{p}^*_{\mathbf{N+1}} &= \tfrac{1}{N+1}(Np^*_N + p_{N+1}) \\ \mathbf{n}^*_{\mathbf{N+1}} &= \tfrac{1}{N+1}(Nn^*_N + n_{p_{N+1}}) \end{aligned} \tag{4}$$

so only N,r^*_N,n^*_N need to be stored for incremental plane model updates.

2.2. *Robust Tunnel Detection*

After plane extraction a map of the tunnel (Fig. 1) needs to be generated. In order to ensure that maps are constrained within expected bounds and to enable efficiency gains during later trajectory and step planning procedures, tunnel detection uses a combination of prior knowledge and template matching. Tunnel detection requires that at least four planes are detected. In the case of looking down a tunnel almost parallel to the surfaces (i.e. the image plane is not perpendicular to any surfaces), and due to the nature of a depth camera that projects light and requires a reflection from the surface, there are often several spurious points detected and the points on the surfaces are noisy and patchy. However, it turns out that the largest detected planes are the walls, roof and floor planes. Generally the tunnel can be assumed to follow a rectangular frame as shown in Fig. 1, even though the interior walls consist of jagged zone planes of varying thickness, and there are several challenging zones. Tunnel detection must overcome these issues in order to robustly generate a tunnel map that incorporates prior knowledge and is idealised for step and path planning.

In order to determine the tunnel width and height, the angle between the normals of two candidate planes, Π_i and Π_j can be determined, $\mathbf{n_i} \cdot \mathbf{n_j}$. Then, if the normals are approximately in opposite directions, the distance

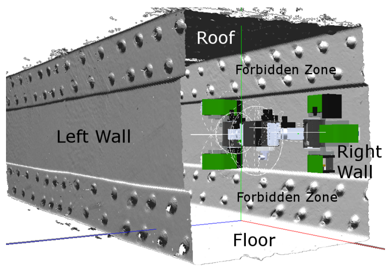

Fig. 1. Simulated robot attached to the right wall by its foot (right footpad) while the sensor-mounted hand (left footpad) looks into the tunnel. Note the simulated tunnel dataset shown here has been manually reconstructed from raw field scan data. Zones adjacent to tunnel edges contain rivets, and are therefore referred to as forbidden zones since the robot must avoid stepping on them. The middle zone plates are the robot's moving base. Multi-thickness zones can cause inconsistent tunnel dimensions to be measured. Rivets on forbidden zones and surface rust can also skew plane normals detected.

between the two planes at the points detected as the center of the planes can be found by the average of the two point to plane distances as,

$$D \approx \frac{1}{2}\left(\frac{\mathbf{n_i}\cdot\mathbf{p_j}-\mathbf{n_i}\cdot\mathbf{p_i}}{|\mathbf{n_i}|}+\frac{\mathbf{n_j}\cdot\mathbf{p_i}-\mathbf{n_j}\cdot\mathbf{p_j}}{|\mathbf{n_j}|}\right), \tag{5}$$

and simplified to, $\frac{1}{2}(\mathbf{n_i}+\mathbf{n_j})\cdot(\mathbf{p_i}-\mathbf{p_j})$, since the normals are of unit length.

In order to robustly detect a rectangular tunnel shape in the presence of uneven wall surfaces, classification and clustering methods are utilised. Tunnels have geometric structure properties that are known *a priori* and exploitable, such as that multiple zones associated with each wall are a plane shifted by small distances, and all zones are symmetric about the tunnel axis that extends into the tunnel. The zone plane normals and positions can thus be dealt with independently since they are uncorrelated. To prepare candidate zone planes, stringent plane parameters are chosen so as to detect planes that are small and accurate. This results in a large set of small planes from the same zone. Initially, each small plane is classified and assigned to one of the four wall classes: left, right, top and bottom, based on the plane's normal orientation. The distribution of the $\mathbb{R}^3$ axial components of the normals in each class can be assumed to follow a Gaussian distribution. The sample mean becomes the wall normal, and is made more accurate after outlier rejection based on histogram analysis. The zone's position on a wall is then determined by forcing each zone plane normal to be equal to the sample mean, and performing a statistical analysis on newly formed planes' positions. Clustering of the positions is used to categorise zones and the cluster's mean becomes the zone's position. For each pair of parallel walls in a tunnel (top-bottom, left-right), the clusters that are the furthest apart are assigned to be the distance between that set of parallel walls.

If there are too few planes detected that are sufficiently large, then the robot moves to a different pose, and takes another scan until the tunnel is detected. Once four walls are detected and dimensions are found to be within the template threshold for tunnel width and height (given the context from previous scans), then a map is generated and combined with prior knowledge about forbidden zone locations.

3. Results

Two experiments where conducted in steel bridge tunnel environments using a 7DOF climbing robot with two cameras mounted to the end effector: a Structure Sensor depth camera, and a Logitech C930e RGB camera. Fig. 2a shows the robot attached to the roof while scanning into the lab tunnel. Experiment 1 was conducted in the laboratory, with the robot attached to the roof or walls, depth data was collected (Fig. 2b) and fused with RGB and triangulated to generate a mesh (Fig. 2c), then analysed to detect the tunnel. Experiment 2 was conducted in the field in a steel bridge tunnel and depth images were collected at 3 different distances (i.e. 1.8m, 3m and 5m) from the manhole as the robot walked along the tunnel. Fig. 2d and Fig. 2e show field data: a depth image and the fused mesh and tunnel.

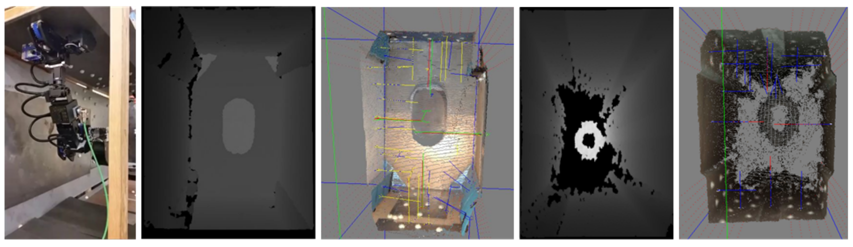

Fig. 2. *a*) The robot scanning the tunnel from the roof; *b*) Lab: depth image of tunnel (aspect ratio altered); *c*)Lab: fused RGB-D and detected tunnel boundary (blue) and traversable planes (red); *d*) Field: depth image (aspect ratio altered); *e*) Field: fused RGB-D and resulting tunnel boundary (blue) and main traversable planes (red)

Planes are extracted from VGA (i.e. 640 × 480) frames and all plane areas and contours are computed in under 100ms, which is real-time enough for the intended application. In experiments, the values for parameters in our Plane Growing and Clustering (PG&C) algorithm are as follows: the region growing's point normal to plane normal angle is 4^o, and point to plane distance is 15mm; the point to candidate point distance is 40mm; the min. points that constitute plane is 40; the min. allowable plane area is $0.004m^2$; the max. usable range from the sensing plane is $5m$; the candidate search of n^2-1 neighboring points has $n = 5$; the max. allowable differences are 40mm for parallel wall normals, and 10mm for similar walls.

The results are presented in Table 1. Our PG&C approach is compared to two alternative algorithms: a RAndom SAmple Consensus & Least Squared Error (R&L) fitting approach[13](p.59) that iteratively perform RANSAC and returns the result with the LSE; and Qhull algorithm[14] (i.e. MATLAB's convhulln function to determine the bounding minimum convex set containing the points, then an axial alignment). Processed was done on an Intel i7-2620M with 8GB RAM. Although speed is not directly comparable since PG&C was written in C++ while the alternative algorithms are in MATLAB, both PG&C and the Qhull approach can detect the tunnel from a 640x480 depth image and create a tunnel map at a rate of approximately 1FPS, whereas the R&L, which is set to iterate 100 times and output the LSE result, runs at 0.07FPS.

Approach	PG & C		Qhull		RANSAC & LSE	
Exp. 1: Lab	$\bar{w}(\sigma)$	$\bar{h}(\sigma)$	$\bar{w}(\sigma)$	$\bar{h}(\sigma)$	$\bar{w}(\sigma)$	$\bar{h}(\sigma)$
Pose 1	824(2.9)	1310(1.8)	849(94.1)	1260(7.8)	752(0.4)	1199(0.7)
Pose 2	827(3.8)	1307(0.8)	803(6.1)	1260(16.1)	751(0.9)	1198(0.8)
Pose 3	824(0.6)	1312(4.2)	827(32.3)	1279(44.8)	751(0.5)	1198(1)
Exp.2: Field	$\bar{w}(\sigma)$	$\bar{h}(\sigma)$	$\bar{w}(\sigma)$	$\bar{h}(\sigma)$	$\bar{w}(\sigma)$	$\bar{h}(\sigma)$
5m to plate	841(1.1)	1234(2.4)	825(10.4)	1174(2.4)	761(2.6)	1099(3.3)
3m to plate	841(3.7)	1225(15.9)	824(14)	1169(11.1)	756(4.6)	1093(9.2)
1.8m to plate	844(1.7)	1213(15.8)	814(5.1)	1155(7.7)	762(1.6)	1083(59.7)

The dimensions of the lab test rig in Experiment 1 are width=0.82m x height=1.3m. The robot was moved to 3 different poses with 10 depth frames captured at each. Our PG&C approach's average width and height measurements (i.e. $\bar{w}$ and $\bar{h}$) were within 12mm of the correct dimensions, and only had a maximum standard deviation, σ of 4mm. The Qhull method detected $\bar{w}$ and $\bar{h}$ within 49mm of the correct result, however values varied significantly with σ up to 45mm. The R&L method was the slowest but most invariant due to the 100 iterations and internal LSE calculation. However, R&L constantly underestimated the dimensions ($\bar{w}$ by 70mm and $\bar{h}$ by 100mm) since parts of another inner surface were often taken to be part of an incorrect inner surface. In Experiment 2 data was collected in the field at 3 locations as the robot walked in a steel bridge with dimensions, [0.84m x 1.2m]. The locations are indicated by their distance to the manhole shown in Fig. 2e. A larger distance from the manhole plate means more tunnel depth data is available for detecting the tunnel, which improves detection accuracy. Conversely, close proximity to a manhole results in less data being available to detect the tunnel, and thus poorer results. In the field, PG&C detected the width correctly and was within 34mm of the correct height, although the variance was larger than the lab (16mm) due to the rough, rusted surface condition. The Qhull method once again slightly underestimated the dimensions with the largest σ=15mm. R&L once again underestimated the dimensions but with a low variance, except for at 1.8m where insufficient data resulted in several incorrect detections.

The presented approach has been shown to work well in both lab, and in the real field environment, where the sensor data was often noisy and sparse for metal surfaces whose condition and reflectivity varied significantly. The approach has been shown to have a similarly low variability to a RANSAC method run over 100 LSE iterations, similar speed as a QHull method, and the suitably high accuracy in both lab and field environments.

4. Conclusions

This paper presents an approach that uses a combination of plane extraction, and template matching techniques to infer from the restricted dataset a usable map. Surface normal features are used which enhance the plane growing speed significantly. A set of variable shaped templates are built using prior knowledge, plane clustering and classification. A sufficiently accurate tunnel map is thus generated from noisy and spurious sensor data, and used to localise a climbing robot in a real-world steel bridge environment. Future work will combine the approach with an exploration approach for automatically detecting the tunnel, and extend this method to become a localisation solution for a climbing robot which is subject to sag.

Acknowledgments

This work is supported by the National Nature Science Foundation of China under Grant 61473258, by the Joint Center for Robotics Research between Zhejiang University, Hangzhou, China and the University of Technolygy, Sydney, NSW, Australia, by the Australian Research Council Linkage Grant (LP100200750), and by the NSW Roads and Maritime Services.

References

1. P. Ward, G. Paul, P. Quin, D. Pagano, C. Yang, D. Liu, K. Waldron, G. Dissanayake, P. Brooks, P. Mann, W. Kaluarachchi, M. P. and L. Matkovic, Climbing robot for steel bridge inspection: Design challenges, in *9th Austroads Bridge Conference*, (Sydney, 2014).
2. F. Rochat, P. Schoeneich, B. Luthi, F. Mondada, H. Bleuler and R. Moser, Cy-mag3d: A simple and miniature climbing robot with advance mobility in ferromagnetic environment., in *Emerg Trends Mob Robot - Proc. 13th Int. Conf. Climbing Walk Robot Support Technol Mobile Machines*, 2010.
3. M. Eich and T. Vogele, Design and control of a lightweight magnetic climbing robot for vessel inspection, in *19th Mediterranean Conference on. MED. Control Automation,*, (Corfu, 2011).
4. D. Ferguson, A. Morris, D. Haehnel, C. Baker, Z. Omohundro, C. Reverte, S. Thayer, W. Whittaker, W. Burgard and S. Thrun, An autonomous robotic system for mapping abandoned mines, in *Proc. Conf. on Advances in Neural Information Processing Systems*, (Vancouver, 2003).
5. P. Quin, G. Paul, A. Alempijevic, D. Liu and G. Dissanayake, Efficient neighbourhood-based information gain approach for exploration of complex 3d environments, in *Int. Conf. on Robotics and Automation (ICRA)*, 2013.
6. P. Quin, G. Paul, A. Alempijevic and D. Liu, Nearest neighbour exploration with backtracking for robotic exploration of complex 3d environments, in *Proc. Australasian Conference on Robotics and Automation*, 2013.
7. G. Paul, S. Webb, D. K. Liu and G. Dissanayake, *Robotics and Autonomous Systems* **59**, 543 (2011).
8. S. Sehestedt, G. Paul, D. Rushton-Smith and D. Liu, Prior-knowledge assisted fast 3d map building of structured environments for steel bridge maintenance, in *Int. Conf. Automation Science and Engineering (CASE)*, 2013.
9. D. Hähnel, W. Burgard and S. Thrun, *Robotics and Autonomous Systems* **44**, 15 (2003).
10. J. Xiao, J. Zhang, B. Adler, H. Zhang and J. Zhang, *Robotics and Autonomous Systems* **61**, 1641 (2013).
11. C. Feng, Y. Taguchi and V. Kamat, Fast plane extraction in organized point clouds using agglomerative hierarchical clustering, in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014.
12. S. Holzer, R. B. Rusu, M. Dixon, S. Gedikli and N. Navab, Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images, in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.
13. M. YÖRÜK, Performance comparison of point and plane features for slam, PhD thesis, Middle East Technical University,2014.
14. C. B. Barber, D. P. Dobkin and H. Huhdanpaa, *ACM Transactions on Mathematical Software (TOMS)* **22**, 469 (1996).