An Algorithm for Surface Growing from Laser Scan Generated Point Clouds

G. Paul, D.K. Liu, and N. Kirchner

ARC Centre of Excellence for Autonomous Systems (CAS), University of Technology, Sydney, P.O. Box 123, Broadway, NSW 2007, Australia
{gpaul, dkliu, nathk}@eng.uts.edu.au

Abstract. In robot applications requiring interaction with a partially/unknown environment, mapping is of paramount importance. This paper presents an effective surface growing algorithm for map building based on laser scan generated point clouds. The algorithm directly converts a point cloud into a surface and normals form which sees a significant reduction in data size and is in a desirable format for planning the interaction with surfaces. It can be used in applications such as robotic cleaning, painting and welding.

1 Introduction

Due to newly introduced OH&S regulations, it is no longer desirable for humans to perform the highly necessary task of sandblasting to remove paint from metal structures such as bridges. Issues arise because of lead contaminated paints being removed and the likelihood of working in close proximity with asbestos. With the long-term health damage done by lead and asbestos being common knowledge there is motivation to perform a significant portion of this manual work with an autonomous system.

One of the major hurdles in building an autonomous system to work in an unknown or partially known environment is mapping. Data must be collected and subsequently processed into a format for use by the autonomous system and image rendering. Due to the abrasive and dusty nature of sandblasting it is not possible to build the environment map (i.e. surfaces of structural members) using traditional techniques of lasers, cameras, infrared or sonar while blasting. This means that mapping of the environment and planning of robot paths and motion must be performed prior to the commencement of sandblasting. Due to time constraints where a sandblasted area requires painting within a couple of hours to avoid corrosion of the newly exposed metal, the time needed for mapping and planning must be kept to a minimum. The whole mapping process, from scanning an area from various locations, fusing the data, to generating surfaces which can be directly used for planning, control and collision detection, must be very fast and efficient. Therefore, an efficient 3D mapping technique is required.

This paper explores an efficient mapping algorithm for surface generation by using laser scanner data at a limited number of discrete positions in an environment (*workspace*) where a 6DOF robot arm is used for cleaning or painting. This algorithm allows for simplicity in gathering and fusing scanned data. Generated maps can be used for 3D path planning, collision detection and robot control.

T.-J. Tarn et al. (Eds.): Robot. Weld., Intellige. & Automation, LNCIS 362, pp. 481–491, 2007.
springerlink.com

2 Related Works

There has been work done in the past on semi-automated sandblasting systems [1, 2]. However, the issue of mapping the environment/area to be sandblasted was excluded and direct tele-operation with a camera was opted for. These systems are reliant upon human control, based upon human decisions and hence require direct human supervision.

Much research has been performed on the creation of 3D CAD objects for CAM and painting with some CAD packages allowing point clouds to be imported and rendered. The generation of virtual objects from scans data is time consuming and the aim is to create enclosed objects. Software packages exist (Qhull) with the ability to wrap and render points into a convex hull shape. However simply rendering the surface is not sufficient for the application in this paper. Besides graphically representing an environment, the usefulness of the environment map in robot path and motion planning, collision detection/avoidance and control is also vital.

Current point cloud graphing methods such as marching cubes [3], tetrahedrons [4] and volumetric method [5] can produce complex detailed surfaces with millions of triangles. Rendered images are optimised for display purposes but are not directly usable for planning and control. In order to identify significant surfaces additional processing is required. In general these methods have focused on creating hulls out of point clouds. However, for a robot system in complex structural environments, it is not possible to scan structures completely. Generally it is only possible to create an unwrapped 3D mesh shell which has no thickness. Processing must be directly on the point cloud with no assumptions on the number of view points or completeness of shape. Other work has been done on surface matching from a point cloud to create surfaces out of a point cloud [6]. This method however, is time consuming and can not be directly used for this application.

Research has also been performed on information-based visual multi-robot mapping [7]. Alternate techniques of 3D mapping are proposed with the focus upon collision avoidance, [8]. These algorithms are fast but they do not build a 3D map for navigation of an end effector over surfaces. There are 3D mapping techniques [9, 10] by integrating tilted 2D scan data. However these systems are expensive, meant for longer range and do not focus upon detailed surface interaction with the environment. Some theoretical algorithms which turn point clouds directly into a path [11] can not be directly utilised in this application.

Methods of facial recognition using Principal Component Analysis (PCA) have been examined [12, 13] as well as 2D position estimation using PCA [14]. PCA mathematical methods have been utilised in this paper to assist in the processing and reduction of 3D range data and to determine surfaces and normals. The aim for this present application is to map and work within the robot arm's *workspace* at known discrete base positions, hence methods of localisation while mapping are left for future work.

3 Sandblasting Environment and Scan Acquisition

In this sandblasting application an autonomous system will work in the environment underneath bridges (Fig. 1a). Tracks will be placed on the scaffold in the channels between sets of two I-beams. A robot arm is mounted on a base platform which moves along the track. The developed 3D laser scanning tool maps the environment to be cleaned.

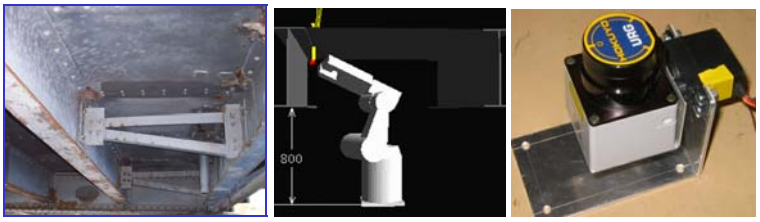


Fig. 1. a) Actual I-beam channel requiring mapped and sandblasted (b) The autonomous system (left); (c) Laser mounted on a digital servo for high precision 3D scanning (right)

The 6DOF robot arm is able to move the end effector to various positions within its *workspace* (Fig. 1b). The 3D scanning operation is performed by a separate scanning tool developed from a 20→4000mm, -120°→120° URG Hokuyo 2D laser scanner (URG-04LX) mounted on a high accuracy, repeatable digital servo (Fig. 1c). The setup is inexpensive, easy to control and highly effective.

The laser scan produces a complete 2D range data set D at 10Hz. To minimise the error and noise, n scans are taken at each increment. Basic filtering is performed on the n range values d_j by first removing q outliers from either side of the data set and then averaging the remaining $n\text{-}2q$ values.

$$D = \sum_{j=q}^{n-q} \frac{d_j}{n-2q} \tag{1}$$

After each scan is taken the servo is tilted by 1/3° through its -60°→60° range (Fig. 2a). The tilt increment of 1/3° was chosen to parallel the 2D laser range scanner's pan increments of 0.36° (Fig. 2b). It is desirable for the point cloud to be spaced relatively evenly between 2D scan sweep lines and next tilt angles (Fig. 2c).

In a complete 2D scan the laser scanner determines the distance and angle to maximum of 768 points. The 120° range of the servo tilt system by increments of 1/3° produces 360 scanning positions (≈0.5sec). At each robot arm pose 240° pan x 120° tilt scan at ranges 20→4000mm can be conducted to produce a maximum of 276480 points in 3D space taking a maximum of 180sec. In each discrete platform position the several scans are taken to map the *workspace*.

During successive scan iteration, the previous scan generated point cloud is processed to determining a valid 3D map for virtual interaction and graphical output.

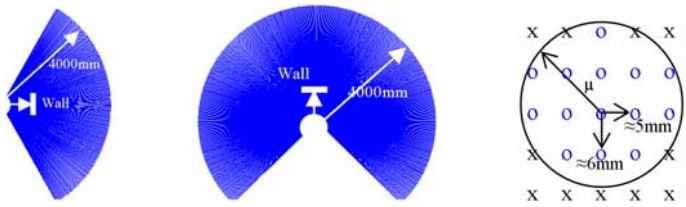


Fig. 2. a) Tilt Scan (Side view): Increments of 1/3°; b) Pan Scan (Top view): Increments of 0.36°; c) 3D Scan Points on wall at 1m range (within *workspace*): distance between points (o=points within circle radius μ; x=points outside circle)

The surfaces and identified important points, sections, features and corners can be used to plan an efficient collision free movement of the robot arm.

4 The Surface Growing Algorithm

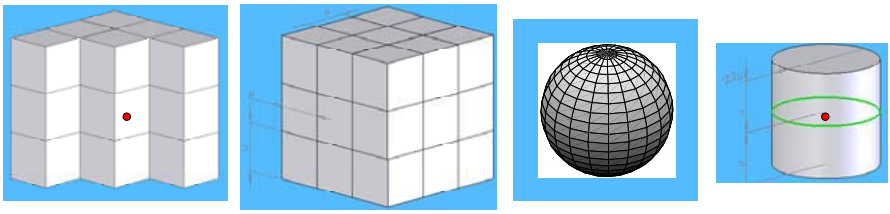
Once the 3D point cloud has been attained the surface growing algorithm works towards creating a 3D occupancy grid, identifying important surface coverage points and generating bound planes. The algorithm is performed in several stages. First the *workspace* is divided into equally sized adjoining cubes; points are indexed to the cubes within which they exist. Points are then grouped around a randomly selected *home point*. The data is analysed to determine the plane of best fit for the points. This along with the normals is outputted. Then finally the planes and normals which make up the surfaces are displayed.

4.1 Cubes Index

Initially the *workspace* of the 6DOF robot arm is described by several hundred thousand raw data points in 3D space. This *workspace* is divided into adjoining cubes with dimensions μ. An index is developed of each cube and the points contained. Indexing is done very quickly taking less than a second for several hundred thousand points. This step reduced the time of the closest points search used in the next step, since searching is limited to the points in the current and surrounding indexed cubes. The cube index makes the calculation of the surfaces and the important points at least 10 times quicker.

4.2 Growing Surfaces

The first step is to randomly select a point which is then called the *home point*. Then use the cube matrix to search the enclosing 1 and surrounding 26 cubes (Fig. 3a and 3b) and find a set of points, *P*, within a certain distance, μ, from the initial *home points* (Fig. 3c). Searching the 3D cube matrix rather than the entire *workspace* reduces the calculation time to microseconds rather than seconds.



(a) (b) (c) (d)

Fig. 3. a) Home point somewhere in center cube; b) Determine the distance to points with a cube index of the home point (red) or within surrounding 26 cubes; c) Do PCA on points within μ of home point to get plane; d) Register points within ±μ of the plane

If there are enough points within the volume surrounding the *home point* ($V=4/3\pi\mu^3 mm$), then PCA is performed on these points (Fig. 3c). This involves determining the covariance matrix *cov(P)* for this set of 3D points and then evaluating the 3 eigenvectors v_λ and corresponding eigenvalues λ. If there is a small enough correlation $<\alpha$ between an eigenvector and the remaining two then that eigenvector $v_{\min(\lambda)}$ is determined to be the normal to the data set .If there are enough points within μ of the *home point* and μ of the plane (Fig. 3d), plane equations, boundaries, enclosed points and *home points* are registered.

$$If\ \min(\lambda) < \frac{1}{\alpha} mid(\lambda)\ then\ \vec{n} = v_{\min(\lambda)} \tag{2}$$

An attempt is made to register all points. However, if there are insufficient points within μ, of a *home point*, then this *home point* will not be registered and is considered scanner noise. The output of this step is a number of planes (Fig. 3d).

4.3 Graphical Output

The planes generated can be directly used for robotic planning and collision detection. For the purposes of creating a graphical representation of the environment, the surfaces of structural members are built based upon these planes. As identified, rendering surfaces from point clouds has been extensively examined by other researchers. Conversely this Surface Growing algorithm's aim is to identify surfaces to interact with from a point cloud.

Initially a single set of points representing the surface of a sphere radius μ is determined (Fig. 4a). For each identified plane, shift the point-represented sphere to have the centre at the *home point* and determine the points of the sphere within a threshold of the plane (Fig. 4b). Finally draw a disk from the points within the threshold and the plane's normal vector from the *home point* (Fig. 4c). The culmination of disks, the graphical output, is shown in section 5.3.

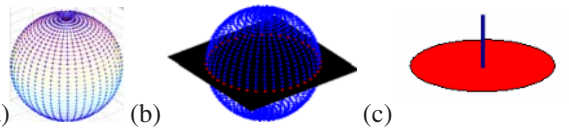


Fig. 4. Graphing algorithm's output planes to display surfaces: a) Point cloud represents the surface of a sphere with the center at *home point*; b) Determine pts lying ≈ on the plane; c) Use points to display an approx. disk

4.4 Important Characteristics of the Algorithm

This algorithm combines different mathematical techniques such as 3D occupancy grids, principal component analysis on 3D data and point cloud analysis to create a novel solution to this surface growing problem. It achieves speed by a simple yet effective method, and is able to output results in a format optimised to meet the requirements of the specified application.

5 Results and Discussion

This section explains the application specific thresholds and the testing of the algorithm. The results are examined including plane/point registration, timing and memory reduction. Finally the usages of the algorithm outputs are discussed.

5.1 Application Specific Thresholds

There are several important thresholds that need to be calculated for this specific application. Firstly, based upon likely data produced by the laser scanning tool in the sandblasting environment the optimal value for μ was determined. μ is used for: the dimensions of cubes that divide the *workspace*; the radius of the closet points sphere used to find the surface and the maximum distance from the points to the surface. Fig. 5.1 describes how computational time, number of planes and percentage of point registered vary for μ=1→50mm. These functions are optimised so as to achieve: the minimum time, maximum number of planes and maximum percentage registered. It was found that μ=12mm produced the best results. Other considerations included the maximum possible cubes enclosed in a *workspace,* the specified laser scanner error (±10mm), the output of surfaces for planning and the sandblasting nozzle which outputs a blast stream covering a radius of 5→15mm.

The other value that needs to be empirically evaluated was the minimum number of points, *min(pts),* that constitute a plane. Insufficient points result in planes created at incorrect angles since there are too few points to correctly perform Principle Component Analysis (PCA) upon. Requiring an excessive number of points results in difficulties in point registration, gaps in the surface and time inefficiency. Fig. 5.2 shows how the time, number of planes and percentage of point registered changes with different values of *min(pts)*. The graphically displayed functions were optimised to determine the maximum possible *min(pts)*, which results in the maximum planes and

percentage of point registration, while minimizing time. The optimal result was determined to be *min(pts)* = 10 points. Additionally, by reexamining Fig. 2a and 2b and considering that the radius of the *workspace* is approximately 1000mm, the grid of points are about 5mm apart at this distance. Fig. 2c shows the point grid with greater than 10 points around the centre *home point* for μ=12mm anywhere within the *workspace* range.

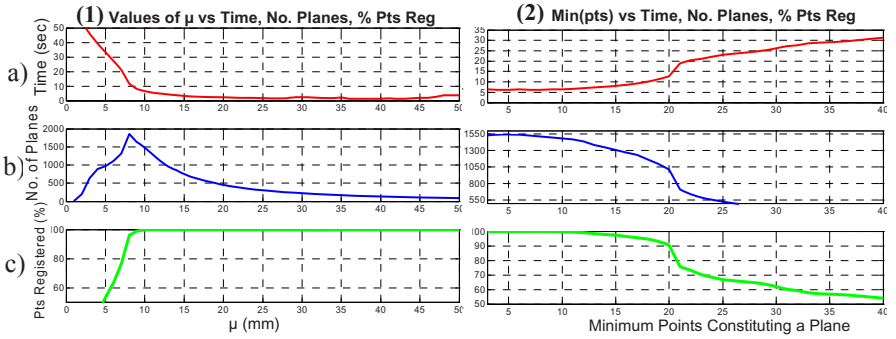


Fig. 5. (1) With respect to μ(mm) (a) • time; (b) No. of planes; (c) Points registered (%) (2) With respect to min(pts); (a) • time; (b) No. of planes ; (c) Points registered (%)

5.2 Testing Overview

This section shows results from tests on different working environments with different volumes of scan data. This allows for comparisons of the output, the time taken, the percentage of points registered and the quality of surfaces created. Quality is based upon comparisons between the generated surface and the actual surfaces from the environment. An in-depth test case is presented to highlight the methods. Following this are the results of 50 test cases as well as a breakdown of the timing of the entire process. This includes: initial scanning; fusion of data from several locations; division into 3D cubes, numerical analysis to minimise points, grow surfaces/normals; then finally rendering and graphing an output for the user.

5.3 In Depth Case Study

This test case is on an I-beam channel structure including features (corners, I-beam webs etc). Several scans are taken from the underneath side of the I-beam channel (Fig. 6a) to generate a point cloud (Fig. 6b). Table 1 shows the details of this case study. The Surface Growing algorithm was performed and the outputs displayed using the technique described. Fig. 6d shows the front view with a close-up showing the normals (Fig. 6e). Fig. 6c shows a side view of surfaces created with a close-up of the plotted planes (disks) in Fig. 6f.

Table 1. Details Of Scanning And Surface Growing Times (sec)

No. Pts	Scans	Scan Time	Fusion Time	Process Time	Graph Time	Planes
75K	5	112.1	3	21.2	2.8	1975

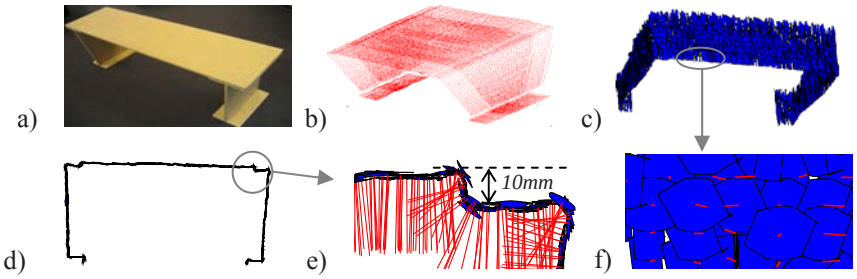


Fig. 6. a) Photo of structure; b) Fused raw data point cloud; c) bottom right view; d) Front viewof planes created; e) Expanded view of the corner; f) front - showing normals (red lines)

5.4 Overall Results

This section shows the results of 50 tests. Data was collected by scanning different structures with the identified *workspace* size several times, fusing the data, and using the Surface Growing algorithm to process this data. The results (Fig. 8a) show the overall algorithm time spent for the processing over the different number of points. Fig. 8b shows the resulting number of planes produced, the percentage of point registered stayed above 99% across the tests. The processing of the data also significantly reduces the amount of data needed to describe the surface sufficiently (turning 100000 scanned {x,y,z} points into 2000 surfaces is a data reduction > 1/20) Fig. 8c shows the reduction in data from the testing results.

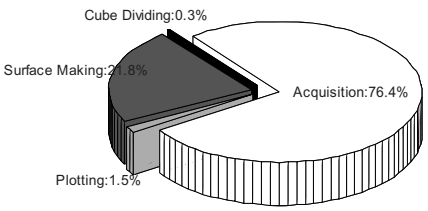


Fig. 7. Overall time breakdown averaged over all tests

Fig. 7 shows the overall breakdown of the total time. It is obvious how the data acquisition time is significantly more time consuming than the other steps in the process. Once the scan data is acquired it can be processed while the next acquisition is being performed.

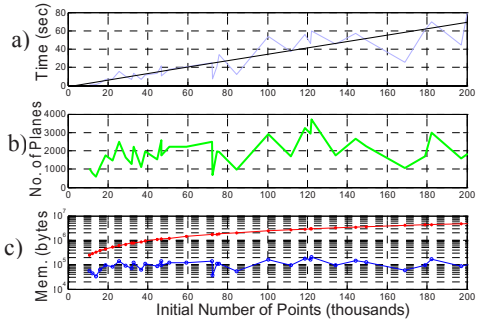


Fig. 8. Results for increasing No. of initial points; a) Overall time (grey) and the trend line (black); b) No. planes; c) Data reduction: original (red *) and post processing (blue o)

5.5 Discussion - Scanning and Surface Growing Output

This algorithm is particularly useful in autonomous structural maintenance system since there are various ways that the output of this algorithm can be applied, including in path planning and motion planning, collision avoidance and as part of the next best possible view by providing 3D occupancy grids.

Initially a complete 3D environment map is not available but there are several Next Best View (NBV) algorithms [15] that could be used to determine the laser scanning tool's position for the next scan. Inverse kinematics is used to determine the 6DOF robot arm pose such that the end effector has a desired position and bearing. The 3D occupancy grid can be used to check if the robot arm in that configuration collides with any points generated from initial scans [16]. If there is an impending collision then it is not a valid pose and if all poses to reach an identified NBV are exhausted then it is not a valid NBV. Hence, poses where the robot arm will collide with some identified, but as yet partially processed points, are discarded in exchange for another NBV pose.

Another way which the output of this algorithm can be used is in path planning. Since the entire area needs to be covered during sandblasting or painting, this algorithm outputs a set of important discrete goal points on the surfaces that would cover the entire known surface. Those goal points can be directly used in path planning which determines the path of the robot arm to follow. The Surface Growing algorithm also outputs the normals to these points which the end effector must align to within a certain angle (5° to 40° to the normal) in order to optimise the blasted surface quality.

The output can also be used in collision detection by applying spherical approach to represent the robot arm and surfaces with a set of spheres with center at the *home point* and radius of μ. This significantly simplifies closest surface calculations and ultimately collision detection calculations.

6 Conclusions

The surface growing algorithm presented in this paper fulfils the requirement of providing useful maps of a partially known or unknown environment consisting of various structural members. It has the ability to transform unstructured 3D points, gathered and fused from multiple laser scans, into a useable format for interaction with surfaces in the environment. It creates 3D occupancy grids, identify important points, normals and bound planes. This can be directly outputted for robot arm path planning and motion control.

Future work will involve integration with 3D path planning and collision detection between the robot arm and the environment, and the incorporation of a more advanced method of rendering the output for graphical display.

Acknowledgement

This work is supported by the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government, and by the UTS Partnership grant with the Roads and Traffic Authority (RTA).

References

1. S. Moon, L. E. Bernold, "Vision-Based Interactive Path Planning for Robotic Bridge Paint Removal," *Journal of Computing in Civil Engineering*, vol. 11, pp. 113-120, 1997.
2. W. Yan, L. Shuliang, X. Dianguo, Z. Yanzheng, *et al.* "Development and application of wall-climbing robots," *IEEE Int. Conf. on Robotics and Automation*, 1999.
3. E. L. William, E. C. Harvey, "Marching cubes: A high resolution 3D surface construction algorithm," *14th annual conf. on Computer graphics and interactive techniques*: ACM Press, 1987.
4. K. S. Levinski, A, "Interactive function-based artistic shape modeling," *presented at First International Symposium on Cyber Worlds*, 2002.
5. B. Curless, M. Levoy, "A Volumetric Method for Building Complex Models from Range Images," Stanford University 1996.
6. K. H. Ko, T. Maekawa, N. M. Patrikalakis, "Algorithms for optimal partial matching of free-form objects with scaling effects," *Graphical Models*, vol. 67, pp. 120-148, 2005.
7. V. A. Sujan, S. Dubowsky, "Efficient Information-based Visual Robotic Mapping in Unstructured Environments," *Int. Journal of Robotics Research*,v.24, pp.275-293,2005.
8. J. Minguez, L. Montano, "Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios," *Trans. on Robotics and Automation,* vol. 20, pp.45- 59, 2004.
9. M. Callieri, A. Fasano, G. Impoco, P. Cignoni, *et al*. "RoboScan: an automatic system for accurate and unattended 3D scanning," *Proceedings. 2nd Int. Symposium on 3D Data Processing*, Visualization and Transmission, 2004.
10. H. Surmann, A. Nüchter, J. Hertzberg, "An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments," Robotics and Autonomous Systems, vol. 45, pp. 181–198, 2003.
11. M. Varsta, P. Koikkalainen, "Surface modeling and robot path generation using self-organization," *presented at 13th Int. Conf. on Pattern Recognition*, 1996.

12. L. I. Smith, "A tutorial on Principal Component Analysis," 2002.
13. G. Afzal, S. Ressler, P. Grother, "Face Recognition using 3D surface and color map information: Comparison and Combination," *presented at SPIE's symposium on "Biometrics Technology for Human Identification"*, Orlando, FL, 2004.
14. J. L. Crowley, F. Wallner, and B. Schiele, "Position estimation using principal components of range data," *IEEE Int. Conf. on Robotics and Automation*, 1998.
15. J. M. Sanchiz and R. B. Fisher, "A next-best-view algorithm for 3d scene recovery with 5 degrees of freedom," *British machine vision conference*, Nottingham, UK, 1999.
16. J.-S. Gutmann, M. Fukuchi, and M. Fujita, "A Floor and Obstacle Height Map for 3D Navigation of a Humanoid Robot," *Int. Conf. on Robotics and Automation*, ICRA 2005.