A Method to Segment a 3D Surface Point Cloud for Selective Sensing in Robotic Exploration

Phillip Curtis, Pierre Payeur
School of Information Technology and Engineering
University of Ottawa
Ottawa, ON, Canada
[pcurtis, ppayeur]@site.uottawa.ca

Abstract—**Autonomous robotic exploration in a 3D environment requires the acquisition of 3D data to create a consistent internal model of the environment from which objects can be recognized for the robot to interact with. As the acquisition of 3D data with stereo vision or a laser range finder can be a relatively long process, selective sensing is desired to optimize the amount of data collected to accurately represent the environment in a minimal amount of time. In order to perform selective sensing, a coarse acquisition of the environment first needs to take place. Regions of interest, such as edges and other boundaries, can then be identified so that an acquisition with higher spatial resolution can occur over bounded regions. For that purpose a segmentation method of the coarse data is proposed so that regions can be efficiently distinguished from each other. The method takes a raw 3D surface profile point cloud of varying point densities, organizes it into a mesh, and then segments the surfaces present in this point cloud, producing a segmented mesh, as well as an octree of labeled voxels corresponding to the segmentation. This mesh and octree may then be used for sensory selection to drive a robot exploration task. The method is demonstrated on actual datasets collected in a laboratory environment.**

Keywords: segmentation; selective sensing; 3D modeling; octrees; robotic exploration.

I. INTRODUCTION

Robotic navigation and object manipulation tasks require that the various sensor acquisitions be fitted into a model of the environment. In practice, a large fraction of data acquired by sensors such as laser range finders or stereo-based systems are heavily inter-correlated while requiring large amounts of time to be collected. In order to optimize the acquisition process, minimizing the redundancy of information from a single point of view is desired. This can be accomplished by performing a coarse acquisition, and then selecting regions of interest within this acquisition to focus on for refinement. To perform this selective sensing, regions of similar stochastic properties and continuity must be separated from each other in order to determine what areas need to be enhanced. The current solutions for segmenting 3D range data mainly concentrate on precise modeling and regularly sampled datasets as opposed to speed that is needed for autonomous robotic tasks.

The proposed method focuses on speed of segmentation as opposed to precision as the purpose is for robotic navigation and task planning. Moreover the method does not rely on regularly sampled data, as many inexpensive commercially available sensors, such as stereoscopic sensors, tend to produce irregularly sampled sparse 3D data acquisitions. A common procedure for segmenting in 3D is to organize the points, usually by a Delaunay triangulation algorithm, and then using those triangles to estimate surface normals at the acquisition points, if not already provided by the sensor [1-4] .

Woo *et al.* [1] use point normal estimations to guide the subdivision process of the points in an octree: as the point normals in a cell start varying by more than a certain threshold, the cell is subdivided, leading to smaller cells where there is large variation, and large cells where there is small variation. Cells that are small are considered edge cells, and the remainder are merged using a spatial adjacency relationship, and are labeled as belonging to the same segment. This technique relies on a reasonable uniformity on sampling density.

Schall *et al.* [2] cluster similar data for the purpose of filtering, and use the point normals as well as the spatial locations to determine if points in a local neighbourhood belong to the same plane. The likelihood function that is used takes into account distance from the point in question when determining what plane it belongs to. When all local likelihoods are calculated, all points are moved to locations of higher probability using an adaptive gradient function. As the probabilities are maximized, the points are merged into a representative point, which represents the merged cluster of points and its corresponding surface. The highly iterative nature of this technique does not lead itself well to automated tasks, but is interesting in that it addresses the non-uniform point density problem.

Hubeli and Gross [3] propose a multi-resolution technique which uses several difference algorithms to achieve flexibility in segmentation of a mesh. The difference algorithms used are normal-based or polynomial fitting-based in order to calculate weights for individual edges. The importance of an edge is determined by using its weight as well as its neighbours weight, and a skeletonizing algorithm extracts features based on the importance of a edge. This technique is flexible in its approach, but the classification aspect of the method assumes reasonably uniform point density along the surface of the object.

Lee *et al.* [4] describe a technique for edge detection on 3D datasets. The first step is to perform a triangulation and then calculate surface normals. An edge is determined by finding areas where the surface normals vary beyond a threshold. Neighbouring surfaces with similar normals are grouped together in regions. To prevent missing gradual edges (e.g. smooth curves), seed faces for the region growing are used. They are also used for the region comparison to detect variation in the normals for finding edges. This algorithm appears to deal well with non-uniform surface point densities, as face size is a parameter used in edge detection. It shows good results when ¾ of the points are removed from the source data.

Mederos *et al.* [5] introduce a technique which creates a surface mesh based upon regions of similarity. The method clusters points in local regions together, and decides whether to subdivide the region into smaller regions based upon the ratio of eigenvalues of the point covariance matrix and the number of points in the local region. After local clustering is determined, a moving least-squares technique helps find a representative point for the region. A triangulation is determined from those representative points, and is continuously refined until certain criteria are met. The clustering technique is interesting as it effectively segments local regions of data with similar stochastic properties without having to perform a triangulation first, and minimizes the amount of data for mesh creation, but unfortunately many iterations seem to be required to produce reasonable results.

Payeur and Chen [6] develop a segmentation method which creates planar surface patches for the purpose of performing registration between differing 3D views. Since a line-based laser range finder is used, a straight line fitting is applied to each 3D line profile acquired, effectively segmenting a line profile into several straight line segments. Normals to the line segments, as well as lengths and centroids are calculated, and then used to combine similar neighbouring line segments of successive acquisitions together to form planar patches. This method is relatively quick, but relies on densely sampled structured data collected over relatively flat surfaces, which counteracts the goal of selective sensing.

II. Overview of Proposed Technique

The proposed technique deals with 3D data acquired from an in-house structured light stereoscopic system [7] which operates under many different conditions, and can acquire data over several types of surfaces. Delaunay triangulation is initially applied on the raw 3D point cloud to produce a surface profile. From this surface profile, the normals of the points are calculated from the weighted average of the normals of the surrounding triangles, with the weights being the inverse of the area of each triangle involved. By using the inverse of the triangle area, more local and densely sampled regions, which are represented by smaller triangles, are given larger weights than sparse regions with less locality, which are represented by larger triangles. The points are then inserted into a multi-resolution octree, which is subdivided based upon the standard deviation of the point normals contained in each voxel. Every voxel in the octree is considered as a seed segment. The latter are merged based upon neighbours determined from the connectivity of the original Delaunay triangulation, and depending on how much deviation there is between the normals of the segments to be merged. Fig. 1 presents the flow of processing to segment a 3D dataset.

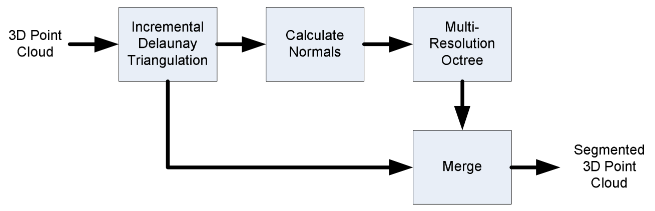

Figure 1. Flow of data for the proposed method.

III. Implementation

A transformation of the data from Cartesian coordinates to a space more suitable to the sensor's method of acquisition is first performed to ensure that projective occlusions in the triangulation do not influence the mapping. A spherical space defined by 3 parameters is used. The parameters respectively represent the angular distance from the Z-Axis to the X-Axis (θ), the angular distance from the Z-Axis to the Y-Axis (φ), and the Euclidean distance from the origin to the point in question (r), as shown in Fig. 2a. The transformations to and from this space are shown in eq. (1). Fig. 2b shows an example of an occlusion (shown as the thinner segment on object 1) that is created by object 2 when observing from the X-Axis in Cartesian coordinates. This occlusion does not occur in the spherical space where the entire length of object 1 can be observed from the origin, which corresponds to the range sensor's point of view.

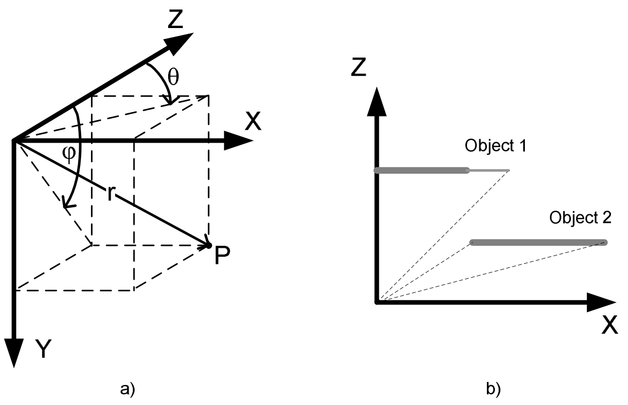

Figure 2. a) Graphical representation of Cartesian and spherical space mappings, and b) graphical representation of occlusion in Cartesian space that does not occur in the spherical space.

$$\begin{aligned} \theta &= atan(x,z) & & & z &= \frac{r}{\sqrt{1+tan^2(\theta)+tan^2(\varphi)}} \\ \varphi &= atan(y,z) & &\leftrightarrow & x &= z \cdot tan(\theta) \\ r &= \sqrt{x^2+y^2+z^2} & & & y &= z \cdot tan(\varphi) \end{aligned} \quad (1)$$

A. Triangulation

While any triangulation method would be acceptable for forming the surface profile mesh, the common Delaunay triangulation was chosen since reasonable performance can be obtained, and it guarantees that there will not be too many long thin triangles [8], and thereby maintaining a strong locality relationship between points. The incremental Delaunay algorithm is used with a triangle data structure that keeps track of neighbouring triangles.

B. Point Normal Calculation

Since the point clouds are not guaranteed to have a regular sampling density, the method takes this fact into consideration. The normal of each triangle, $\vec{N}_t$, is calculated as well as the area of the triangle, A_t. For every 3D point, p, the point normal, $\vec{N}_p$, is calculated by a weighted average of all triangles, t, from which the point is a vertex, as shown in eq. (2). Since regions of higher point densities are represented by smaller triangles, and lower density areas are represented by larger triangles, the inverse of triangle area is used as the weight to ensure that neighbour points that are more closely related to the point for which the normal is currently calculated receive a higher weight than points that are located further away and are represented by larger triangles. The final result of the weighted average is normalized.

$$\vec{N}_p{}' = \sum_{\forall t \in p} A_t^{-1} \cdot \vec{N}_t$$
$$\vec{N}_p = \frac{\vec{N}_p{}'}{\left\| \vec{N}_p{}' \right\|} \qquad (2)$$

C. Octree Generation

The points are inserted into a multi-resolution octree which is subdivided based upon the standard deviation of the voxel's, v, normal. The first step in this process is to determine the area which a point is associated with, A_P. It is calculated in eq. (3) as the sum of the areas of the triangles to which the point is a vertex. The inverse of A_p is used as the weight applied to the corresponding point normals for calculating the voxel normal, $\vec{N}_v$, as shown in eq. (4). Again this ensures that sparsely sampled areas do not dominate over denser sampled areas in the calculation of the voxel normal.

$$A_p = \sum_{\forall t \in p} A_t \qquad (3)$$

$$\vec{N}_v{}' = \sum_{\forall p \in v} A_p{}^{-1} \vec{N}_p$$
$$\vec{N}_v = \frac{\vec{N}_v{}'}{\left\| \vec{N}_v{}' \right\|} \qquad (4)$$

The standard deviation of the point normals, σ_v , contained in a voxel is then calculated as shown in eq. (5). A threshold is applied to the standard deviation along each axis, $\sigma_{v,\theta}$ for the θ-axis and $\sigma_{v,\varphi}$ for the φ-axis. If either one exceeds this threshold, the voxel is split into eight different voxels. A threshold of 7.6° deviation in both directions was determined to work well through trial and error.

$$\sigma_v^2 = \begin{bmatrix} \sigma_{v,\theta}^2 \\ \sigma_{v,\varphi}^2 \end{bmatrix} = \begin{bmatrix} \sum_{\forall p \in v} A_p^{-1} \left(N_{p,\theta} - N_{v,\theta} \right)^2 \Big/ \sum_{\forall p \in v} A_p^{-1} \\ \sum_{\forall p \in v} A_p^{-1} \left(N_{p,\varphi} - N_{v,\varphi} \right)^2 \Big/ \sum_{\forall p \in v} A_p^{-1} \end{bmatrix} \qquad (5)$$

D. Merging Octree Voxels

The octree representation of sparse and irregularly sampled datasets will inevitably contain gaps between voxels, and the voxels will have neighbours that are statistically similar. In order to prevent the fragmentation of the environment into many voxel sized segments, and to maximise the area of statistically similar regions, a merging of voxels into larger segments needs to occur. The initial step is to label the points contained in each voxel of the octree as belonging to their own segment corresponding to that voxel. These initial segments are used as seed segments, which are then merged together to form larger segments of stochastically similar contiguous regions. This merging relies on the information of adjacency contained in each triangle from the Delaunay triangulation and the information already calculated when determining whether or not to subdivide the octree, which corresponds to surface orientation.

The representative area, A_v, of the points contained by the voxel is calculated, as shown in eq. (6), by summation of the areas associated with points. The normal for each segment is then calculated using a weighted mean, eq. (7). The inverse of the area is used as the weighting factor to ensure that voxels that contain a denser contribution of points have more weight than a less dense voxel.

$$A_v = \sum_{\forall p \in v} A_p \qquad (6)$$

$$\vec{N}_{Sk}' = \sum_{\forall v \in S_k} A_v^{-1} \vec{N}_v$$
$$\vec{N}_{Sk} = \frac{\vec{N}_{Sk}'}{\left\| \vec{N}_{Sk}' \right\|} \qquad (7)$$

Each voxel contains a collection of points, p_j, and these points belong to triangles, $t_{l,k}$. Triangles can be considered as constructs that connect together different points that may or may not be in the same segment. Triangles that have all three of their points belonging to the same segment, S_k, are considered as interior triangles belonging to that segment, $T_{I,k}$. Triangles that have less than three points belonging to a particular segment are considered as edge triangles as they lie on a boundary between two segments. Furthermore, two classifications of edge triangles are made. If a triangle's normal is within a certain range of the segment's normal then the triangle is considered a valid one to consider for searching along its vertices for segments to merge with the current segment ($T_{VE,k}$). If it is not, then the triangle probably lies on a boundary between different regions, and is not a valid triangle to use to search for possible segment merging ($T_{IE,k}$), as is shown in eq. (8).

$$t_{l,k} \subset \begin{cases} T_{I,k} & \text{iff } \forall p_j \in t_{l,k}, p_j \in S_k \\ T_{VE,k} & \text{iff } t_{l,k} \notin T_{I,k}, \left|\vec{N}_{l,k} - \vec{N}_{Sk}\right| \leq 25^\circ \\ T_{IE,k} & \text{otherwise} \end{cases} \tag{8}$$

Merging segments together relies on searching the vertices of valid edge triangles for other segments, and then determining if these segments are statistically similar to the current segment. If the difference between the segment normals is less than a threshold, as shown in eq. (9), then the segments are merged together, as shown in eq. (10). A threshold of 25° was found to work well for our purposes. Fig. 3 illustrates this classification: triangle *d* is entirely in segment *A*, so it is considered an internal triangle. Triangle *e* is an edge triangle, as it has a vertex in both segments *A* and *B*. Triangle *f* is also an edge triangle, since it has a vertex in segments *A* and *C*, but since it also crosses a transition region (an edge or other discontinuity) and hence its normal is significantly different than either segments, it is considered an invalid triangle to use to search for regions to merge.

$$\left|\vec{N}_{Sk} - \vec{N}_{Sj}\right| \leq 25^\circ \tag{9}$$

$$S_k^{'} = S_k \cup S_j \tag{10}$$

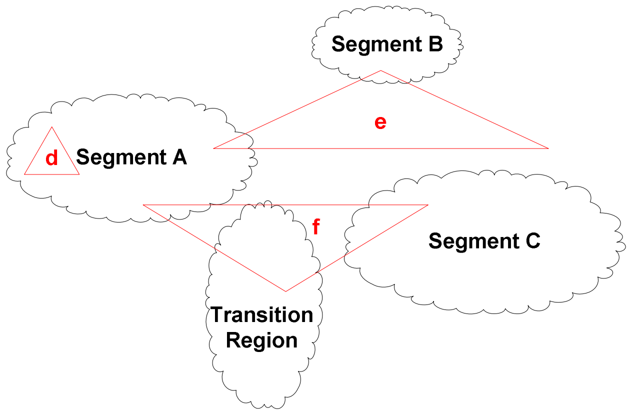

Figure 3. Merging process between seed segments.

IV. Experimental Results

Our preliminary datasets include data acquired from an in-house 3D stereo acquisition system described in [7]. When reading the point data, all information other than positional data was removed and the points order randomized to ensure optimal performance from the Delaunay algorithm. The datasets used include an acquisition of a cube (Fig. 4), a mock up chair (Fig. 5), and a mock up of a car door with a ding in it (Fig. 6).

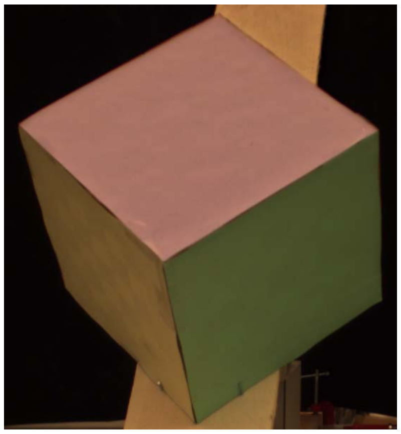

Figure 4. Image of cube used in cube dataset.

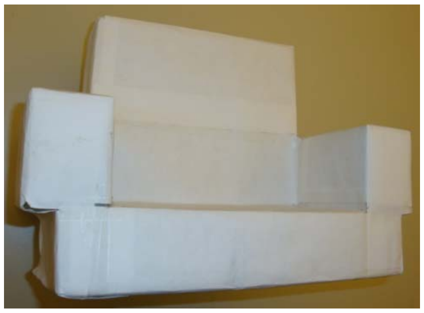

Figure 5. Image of chair used in chair dataset.

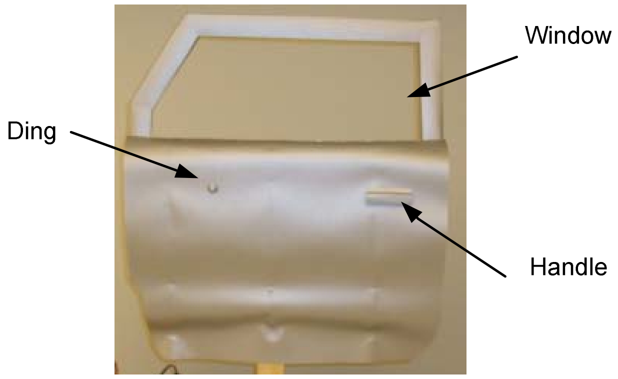

Figure 6. Image of car door used in door dataset.

The segments shown in Fig. 7-9 are coloured according segment labels mapped to unique hues of HSV space to better show separation between segments. White triangles represent edge triangles, and may appear as shades of grey in the images due to lighting and shading effects.

The segmented cube dataset (Fig. 7) shows five significant segments, one for each visible face coloured with pink, purple and blue, and two segments at the bottom in orange and green which constitute background surface (refer to Fig. 4). Each of these are a separate planar surface, and are labeled as different segments as expected. The edges of the cube contain very small segments as well as edge triangles.

The chair (Fig. 8) shows two large segments, namely the front of the chair in light blue, the back rest portion in dark blue, and smaller segments on the inner right armrest in orange and red. These constitute different planar surfaces of the chair, and are correctly labeled as different segments.

The door with the ding (Fig. 9) shows several distinct segments. One for the main portion of the door in light green, one for the window frame segment in light blue, one for the handle in dark blue, a small one for the ding in purple, several for where the door curves towards the window segment, and several segments for the background wall, with the predominant segments in pink and orange. There are also several smaller segments within the main region of the door due to noise present in the data.

The multi-resolution octree visualizations (Fig. 10-12) show the same coloured segmentation methodology as for the meshes. As can be observed, a region where there is large surface variation is represented by smaller voxels, while regions with low variation are represented by larger voxels similar to [1]. This effect is most visible in the chair (Fig. 11) where the back rest has large voxels in the middle and smaller voxels near the edges, and the cube (Fig. 10) where the faces have larger voxels than the edges.

Table 1 shows the number of points for each dataset, as well as the execution times for each step. These tests were run on a Pentium IV 3.4 GHz Windows XP based computer with 2.5 GB of memory. It should be noted that a full segmentation of the dataset of the door containing slightly more than 46000 points needs about 5.0 seconds, which corresponds to a maximum segmentation time of the order of 0.1 ms per 3D point in a dataset. Given that measurements sparsely collected in a typical robotic exploration task would contain a few thousands of points, the proposed segmentation method is suitable to perform real-time selective sensing to improve autonomous robot navigation and path planning.

TABLE I. SEGMENTATION EXECUTION TIMES

Data Set	**Segmentation Results**					
	# Points	***Delaunay (s)***	***Normal Calc. (s)***	***Octree Gen. (s)***	***Merge Voxels (s)***	***Total (s)***
Door	46255	0.519	0.204	0.307	3.971	5.001
Chair	1544	0.013	0.006	0.008	0.025	0.052
Cube	5432	0.055	0.027	0.022	0.050	0.154

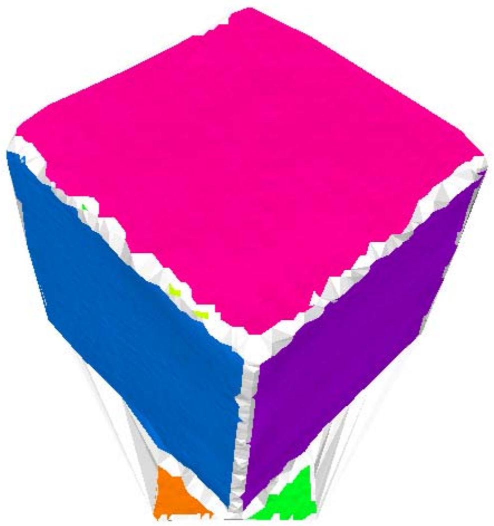
Figure 7. Mesh of segmented cube dataset.

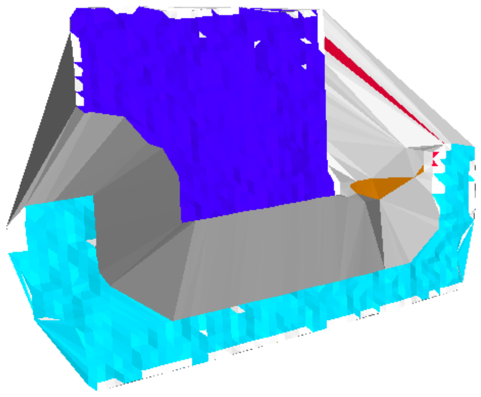
Figure 8. Mesh of segmented chair dataset.

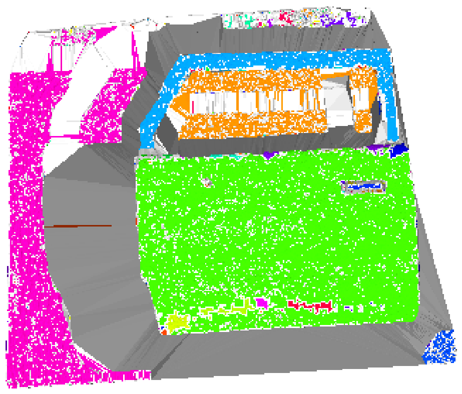
Figure 9. Mesh of segmented door dataset.

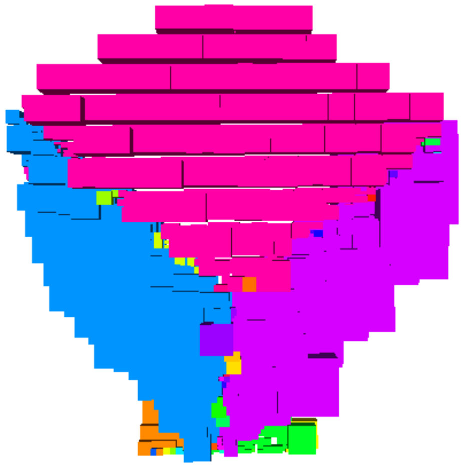

Figure 10. Octree visualisation of segmented cube dataset.

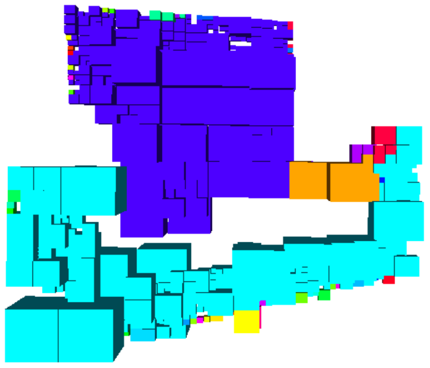

Figure 11. Octree visualisation of segmented chair dataset.

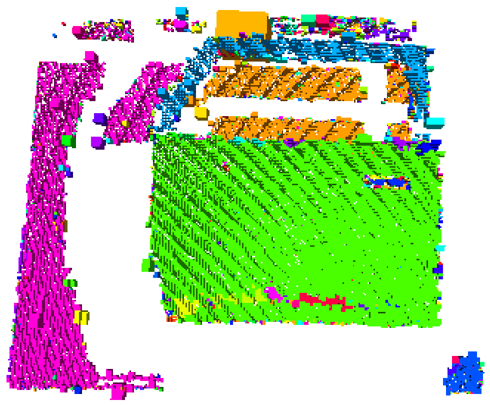

Figure 12. Octree visualisation of segmented door dataset.

Conclusion

This paper introduced a quick and efficient segmentation technique of 3D point clouds for the use in robotic exploration tasks. Regions of the surface of objects that are separated by a discontinuity or an edge are automatically labeled as different segments. A surface mesh and an octree are both produced in the segmentation process, and can be refined and reused to aid in the navigation and task planning processes. The method can handle data with non-uniform sampling density, as the technique takes into account local densities of sampling points. As such a robotic sensory system can perform a coarse acquisition over the environment and then, based on the task, perform a finer acquisition only over regions of interest that correspond to specific segments or transition regions determined by the proposed segmentation algorithm. Future work will integrate an automatic mechanism to select regions of interest for finer scanning around the candidate segments and refine the approach for the segmentation to rely on parameters that go beyond relative surface orientation.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the support from Communications and Information Technology Ontario Centre of Excellence, and from the Natural Sciences and Engineering Research Council of Canada to this research work.

References

[1] H. Woo, E. Kang, S. Wang, K.H. Lee, “A new segmentation method for point cloud data”, *International Journal of Machine Tools & Manufacture,* vol. 42, pp. 167-178, 2002.

[2] O. Schall, A. Belyaev, H.-P. Seidel, “Robust filtering of noisy scattered point data”, *Proc. of the IEEE Eurographics Symposium on Point-Based Graphics*, pp. 71-77, Long Island, New York, USA, 20-21 June, 2005.

[3] A. Hubeli, M. Gross, “Multiresolution feature extraction from unstructured meshes”, *Proc. of the IEEE Conference on Visualization*, pp. 287-294, San Diego, California, USA, 2001.

[4] Y. Lee, S. Park, Y. Jun, W.C. Choi, “A robust approach to edge detection of scanned point data”, *International Journal of Advanced Manufacturing Technology*, Springer London, vol. 23, no. 3-4, pp. 263-271, 2004.

[5] B. Mederos, L. Velho, L. H. de Figueiredo, “Moving least squares multiresolution surface approximation”, *Proc. of the IEEE Brazilian Symposium on Computer Graphics and Image*, pp. 19-26, São Carlos, Brazil, 12-15 Oct., 2003.

[6] P. Payeur, C. Chen, "Registration of range measurements with compact surface representation", *IEEE Transactions on Instrumentation and Measurement*, vol. 52, no. 5, pp. 1627-1634, Oct. 2003.

[7] A. Boyer, P. Curtis, P. Payeur, “3D modeling from multiple views with integrated registration and data fusion:, *Proc. of the Canadian Conference on Computer and Robot Vision,* pp. 252-259, Kelowna, BC, 25-27 May 2009.

[8] Ø. Hjelle, M. Dæhlen, “Delaunay triangulations and Voronoi diagrams”, in *Triangulations and Applications*, Springer Berlin Heidelberg, pp. 47-71, 2006.