Range Image Segmentation for Modeling and Object Detection in Urban Scenes[*]

Cecilia Chao Chen
Graduate Center / CUNY
New York, NY 10016
cchen@gc.cuny.edu

Ioannis Stamos
Hunter College / CUNY
New York, NY 10021
istamos@hunter.cuny.edu

Abstract

We present fast and accurate segmentation algorithms of range images of urban scenes. The utilization of these algorithms is essential as a pre-processing step for a variety of tasks, that include 3D modeling, registration, or object recognition. The accuracy of the segmentation module is critical for the performance of these higher-level tasks. In this paper, we present a novel algorithm for extracting planar, smooth non-planar, and non-smooth connected segments. In addition to segmenting each individual range image, our methods also merge registered segmented images. That results in coherent segments that correspond to urban objects (such as facades, windows, ceilings, etc.) of a complete large scale urban scene. We present results from experiments of one exterior scene (Cooper Union building, NYC) and one interior scene (Grand Central Station, NYC).

1. Introduction

We present fast and accurate segmentation algorithms of range images of urban scenes. The utilization of these algorithms is essential as a pre-processing step for a variety of tasks, that include 3D modeling, registration, or object recognition. The accuracy of the segmentation module is critical for the performance of these higher-level tasks. The major challenges in this direction have to do with a) over- or under-segmentation, and b) generation of disconnected segments due to occlusions or incomplete information. In this paper, we present a novel algorithm for extracting planar, smooth non-planar, and non-smooth connected segments, and then merging all these extracted segments from a set of overlapping range images. Our input is a collection of registered range images. Our output is a number of segments that describe urban entities (e.g. facades, windows, ceilings, architectural details). In this work we detect different segments, but we do not yet identify (or recognize) them. Previous work [6] also segments range image into distinct surfaces (e.g. walls, chairs, lamps), utilizing recursive graph cut approach. Our approach, however, is more efficient, and more accurate at planar surface extraction.

The first step of most range image segmentation algorithms (either edge- or region-based [7][8][9][10][11]) is the surface normal computation at every 3D range point. This is a critical step for precise region extraction, as inaccurate normal of range points near region boundaries might cause inaccurate segmentation results (details in Sec. 2.1). Our algorithm calculates local surface normals for all planar points with high accuracy, and extracts precise boundaries of planar regions. Additionally, the algorithm accurately extracts smooth non-planar regions with gradually changing surface normals, since these regions appear almost planar at every point's local neighborhood. Finally, the remaining points are clustered into connected components.

In addition to segmenting each individual scan, our methods also merge registered segmented images. The merging results in coherent segments that correspond to urban objects (e.g. facades, windows, ceilings) of a complete large scale urban scene. Based on this, we generate a different mesh for each object. In a modeling framework, higher order processes can thus manipulate, alter, or replace individual segments. In an object recognition framework, these segments can be invaluable for detecting and recognizing different elements of urban scenes.

We present experimental results of one exterior scene (Cooper Union building, NYC) and one interior scene (Grand Central Station, NYC). Our paper is organized as follows: Section 2 presents the segmentation algorithm, Section 3 the merging algorithm, Section 4 the experimental results and conclusions.

2. Range Image Segmentation

The first step is to acquire a set of range scans $R_m (m = 1, \dots, M)$ that adequately cover the 3D scene. The laser range scanner used in our work is a Leica HDS 2500[1], an active sensor that sweeps an eye-safe laser beam across the

[*]Supported in part by NSF CAREER IIS-0237878 and NSF MRI/RUI EIA-0215962.

scene. It is capable of gathering one million 3D points at a maximum distance of 100 meters with an accuracy of 6mm. Each 3D point is associated with four values $(x, y, z, l)^T$, where $(x, y, z)^T$ is its Cartesian coordinates in the scanner's local coordinate system, and l is the laser intensity of the returned laser beam.

In this section we present our range image segmentation method. It consists of the following steps, the first one being the most critical: a) Normal computation on each 3D point (Sec. 2.1), b) Extraction of planar and smooth non-planar regions (Sec. 2.2), and c) Extraction of non-smooth connected regions (Sec. 2.3).

2.1. Normal Computation

Each range scan R_i is represented as a two-dimensional array of 3D points $\{\mathbf{r}(k,l), k = 1 \ldots N, l = 1 \ldots M\}$[1]. We take advantage of the adjacency of 3D points, gathered in a regular two-dimensional grid by the range scanning equipment, in order to speed up the normal computation and clustering processes. We can thus estimate the local surface normal of each 3-D point by fitting a local plane on the 3-D points of its $k \times k$ neighborhood. We note, however, that simply fitting a local plane within a $k \times k$ neighborhood will result in inaccurate calculation of surface normals close to surface boundaries, or in non-locally planar areas. We first present several types of local $k \times k$ neighborhoods, and then describe how our algorithm handles each case.

Fig. 1 shows several cases of point P (the black dot) and its local neighborhoods (all the hollow dots). Without loss of generality, we display 5×5 neighborhoods. In each subfigure, the arrow shows the approximate scanning direction. Regions S, S_1 and S_2 are real surfaces, while F is the fitted local plane at P. In all cases except (f) S, S_1 and S_2 are approximately planar. In (f), S is a general surface. In Figs. 1 (a) and (b), P is a locally planar point, but the angle between its local plane and the scanning direction differs between (a) and (b). In Fig. 1 (a), the angle is small. Therefore, some of P's neighbors are farther away than others. In Fig. 1 (b), the angle is large, i.e. the viewing direction is almost vertical to P's local plane. Thus, all its neighbors are almost evenly distributed. In Figs. 1 (c),(d), and (e), P is an edge point. Fig. 1 (c) shows a jump edge with missing neighbors; Fig. 1 (d) shows a jump edge between two surfaces; Fig. 1 (e) shows a fold edge. Finally, in Fig. 1 (f), P is a non-planar point.

Previous methods [7][10][11][12] set a distance threshold for selecting the neighbors of a point P for surface fitting. The distance threshold is either a pre-selected constant, or is calculated from the point distribution in the range image. Although these approaches correctly calculate the

[1]The indices k, l define the position and orientation of the laser-beam which produces the 3-D point $\mathbf{r}(k, l)$.

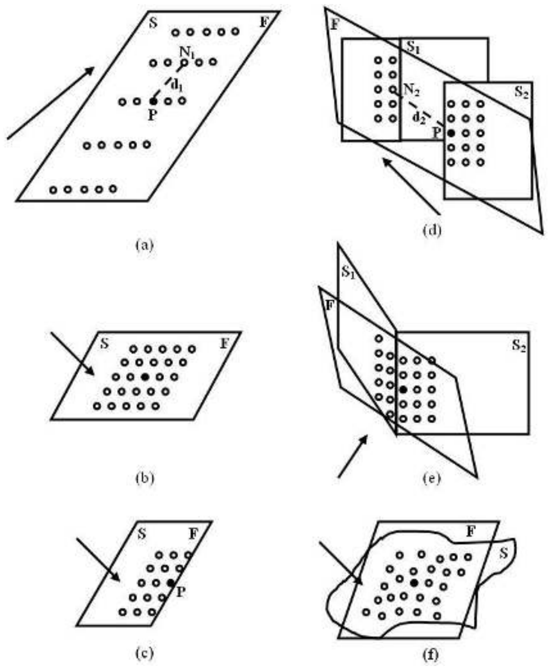


Figure 1. Different types of local surface.

surface normal for most locally planar points, they may fail in the following cases.

In Fig. 1 (a), all of P's neighbors should be used to calculate surface normal, while in Fig. 1 (d), only those neighbors located on plane S_2 should be used. If both such cases exist in a range image and $d_1 > d_2$, then some surface normal computations will be inaccurate, irrespective of the selected distance threshold. Another difficult case is caused by fold edges (Fig. 1 (e)). Neighbors from both S_1 and S_2 are used to compute surface normal of P, resulting in plane F. Previous approaches including optimal scale selection [2] and k-nearest neighbors [3] are not capable of computing $P's$ true local plane, S_2, in this case. Methods with robust estimators [4][5] are possible solutions, but they have higher time complexity comparing to our method.

In our algorithm, we carry out local plane fitting followed by refining steps in order to achieve accurate calculation of surface normals for points in all the above cases. During this process, points are categorized into four types: Interior planar points (type 1), wide jump edge points (type 2), narrow jump edge and fold edge points (type 3), and non-planar points (type 4). Interior planar points are used to refine the surface normal calculation for edge points. Below we describe the algorithm in more detail.

(1) First, we consider all existing neighbors in the $k \times k$ neighborhood of every point P and fit a local plane. We set k to be 5 in our experiments. During the plane fitting, the maximum distance between any neighbor point to the fitted

plane, d_{max}, is used to measure the goodness of fit. By setting a very small threshold θ_{fit} (in our algorithm $\theta_{fit} = 0.01m$, which is slightly larger than the maximum expected error of our range scanner 6mm), we identify planar points (Figs. 1 (a)(b)) and edge points (Fig. 1 (c)). The almost perfect fit assures us that only points of the same plane have been used. These points are marked as type 1 points.

(2) Second, we recalculate the surface normal for some points (i.e. points that fall into the case of Fig. 1 (d)). More specifically, we consider points on relatively wide jump edges. In this step, we discard neighbors of P by using a dynamic threshold that depends on its depth, $depth_P$, and the average angle $\hat{\alpha}$ between neighboring scanning beams in the range image, as explained below.

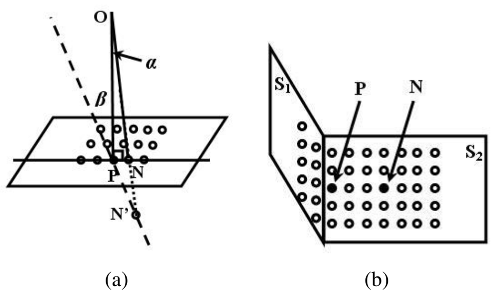


Figure 2. (a) Distance estimation between neighbors (see text). (b) Refinement of P's surface normal with its neighbor N.

Assume point P's local surface plane is perpendicular to the scanning direction (see Fig. 2(a)). The distance between P and its neighbor N is $|PN| = dist_P = depth_P * tan(\alpha) \doteq depth_P * \alpha$ (see footnote[2]). This value is the expected distance between neighboring points. Note, however, that the scanning direction may not be exactly perpendicular to surface normal. As the angle β between surface line PN and scanning direction decreases, the distance $|PN|$ becomes larger. When $\beta = 15°$, as the dashed line PN' shows, the distance $PN' \approx \frac{PN}{sin(\angle PN'N)} \approx \frac{PN}{sin\beta} \approx 4 \cdot PN \doteq 4 \cdot depth_P \cdot \alpha$. In order to accommodate for cases where β ranges between $15°$ and $90°$ (most surfaces in our scans are within this range), we calculate $4 \cdot depth_P \cdot \alpha$ at every point P and use it as a loose distance threshold for its immediate neighbors.

Since the threshold derived above is a very loose one, it can only be used to accurately compute surface normals for points near wide jump edges (Fig. 1 (d)). Similarly to step 1, we measure the goodness of local plane fitting by comparing the fitting error d_{max} with θ_{fit}. Well-fitted points (i.e. points with $d_{max} < \theta_{fit}$) are categorized as type 2 points (Fig. 1(d) with large d_2). The processing of other narrow jump edges is described in step 3 below.

(3) This step aims at correcting normal calculations (surface fitting) for narrow jump edges (Fig. 1 (d)) as well as fold edges (Fig. 1 (e)). The previous two steps identified all planar points that are relatively far from planar region boundaries. Points near boundaries would still have inaccurate surface normals. Let us illustrate the calculation of surface normals of these points using Fig. 2(b). P is not identified as planar due to the large plane fitting error. However, it is at most k points away from an identified planar point N that is on the same plane with it because N's surface normal was accurately calculated with its $k \times k$ neighborhood. As such, our method searches around a neighborhood slightly larger than $k \times k$, and locates the best neighbor point N so that the distance from P to N's local plane is minimum.

Specifically, the algorithm considers every point P that is neither of type 1 nor of type 2. We go through P's $(k+2) \times (k+2)$ neighborhood, and compute the distances of P to the fitted local planes of all type 1 or type 2 neighbors. Also, we compute the distance between P and the center of its own fitted local plane (if P has been fitted twice in steps 1 and 2, the one with smaller fitting error is selected). Among these point-to-plane distances, the smallest one is selected. If this is the distance of P to N's plane, and the distance is smaller than threshold θ_{fit}, we then consider point P to be belonging to N's local plane. In that case, P's local surface normal is set to be the same as N's surface normal. The planar points identified in this step are categorized to be type 3 points (Fig. 1(d) with small d_2 and Fig. 1(e)).

(4) All the remaining points are categorized as type 4 points (Fig. 1(f)), and their surface normal is calculated from either step 1 or step 2, choosing the one that produces the smallest surface fitting error.

2.2. Extraction of Planar and Smooth Non-planar Regions

After the calculation of local surface normals, a region growing procedure is implemented. That involves clustering neighboring points using the following metric: two neighboring points P and N are part of the same region if a) they are spatially close, b) they have similar local surface normals, and c) they have similar laser intensity. In previous algorithms, the distance threshold is static - either preselected, or determined empirically. This, as we explained earlier, may cause inaccurate segmentations. In our algorithm, we replace the point-to-point distance threshold with point-to-plane distance threshold. In particular, we determine if a point P's neighbor N belongs to the same local plane as P by checking the distance from N to P's local plane instead of distance between P and N. This is advanta-

[2]We actually calculate $depth_P * \alpha$ for better performance, because $tan(\alpha) \doteq \alpha$ when $\alpha < 15°$. In our range image data, angle between neighboring scanning beams is at most $0.002°$, and in most range images, α is small enough to satisfy $tan(\alpha) \doteq \alpha$.

geous because, for planar points, point-to-plane distance is always very small, regardless of surface orientation. Therefore, we may set a constant threshold in that case. The next paragraph describes this step in more detail.

At the initial state of region growing, all points are unlabelled. Starting from the upper-left corner of the range image's grid, we select the next unlabelled point as a seed to grow a new region. Each time a new region is formed, points in this new region are labelled with an integer as their region label. The first region is labelled as region 0, and the succeeding formed regions are labelled with $1, 2, 3, ...$ Assuming the seed for a new region is point P, we test whether each of its neighbor points, N, belongs to P's local plane. If it does, N is labelled with P's region label, and N's neighbors are further compared with N to continue region growing. This recursive procedure continues until no more neighbor points are found to belong to this region. When comparing N with P, three thresholds are used: a) the distance between N to P's local plane should be smaller than a threshold $\theta_{point2plane}$; b) the intensity difference should be smaller than a threshold $\theta_{intensity}$; and c) the surface normal difference between P's and N's local planes should be smaller than an angle threshold θ_{angle}. In our algorithm, these three thresholds are $0.015m$, 0.01, and $2°$ respectively. Note that $\theta_{intensity}$ and θ_{angle} are not sensitive to surface orientation. Varying the values for these thresholds leads to coarser or finer segmentation, and the values in our algorithm are selected empirically. The last step consists of discarding regions containing very few points (in our algorithm < 100) to avoid generating very small regions

2.3. Extraction of Non-smooth Regions

The previous region growing procedure calculates major planar and smoothly varying surfaces (Fig. 3(c)). Based on the unlabelled points remaining from the earlier region growing phase, we carry out a second round of point clustering, to cluster non-smooth regions. In urban areas, after large smooth surfaces are identified, the remaining non-smooth regions are usually objects in the scene, such as windows, lights, and furniture. In range images, these objects have varying local surface normals on their points, and they may be separated from each other by smooth background regions. Therefore, the clustering criteria of non-smooth region points is defined with higher emphasis on spatial closeness and lower emphasis on surface normal similarity. The process for clustering non-smooth regions is similar to that of growing planar region. The difference is in the details of comparing neighboring points. Due to the reason we described above, we compare intensity, point-to-point distance, and angle between surface normals of two neighboring points with thresholds $\theta_{intensity}$, $\theta_{point2point}$, and θ'_{angle}. The value for $\theta_{point2point}$ is $dist_P * r$, r being an empirically calculated constant larger than 1. The threshold θ'_{angle} is a large value (in our algorithm it is $65°$) to avoid growing regions with the neighbors having large angles between their surface normals. We assume that a large angle between surface normals will appear at boundaries of semantically different regions.

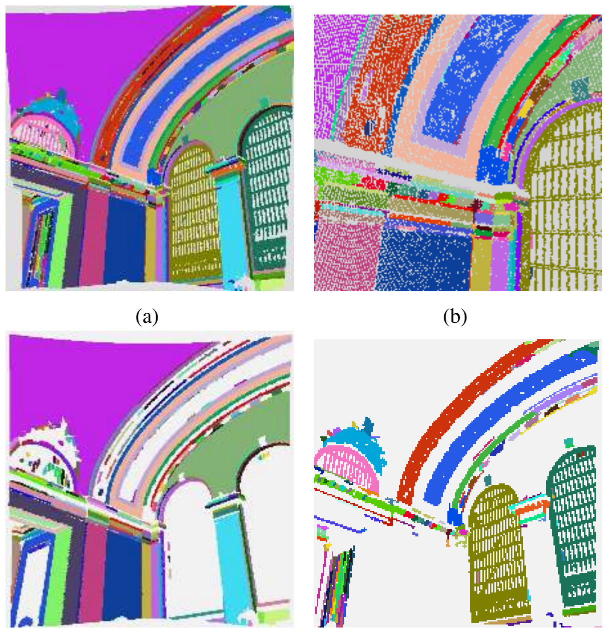


Figure 3. Segmentation results. (a) Segmented regions as colored patches. (b) Details at region boundaries. (c) Planar and smooth regions. (d) Non-smooth regions.

Fig. 3 shows the segmentation result of a range image of an interior scene. Each segmented smooth region and non-smooth region is assigned to a random color computed with its label. In Fig. 3 (d), some regions seem to be smooth, but are clearly non-smooth when zoomed in. e.g., the long curved stripe is in fact rosette pattern as shown in Fig. 3 (b). Fig. 4 shows the segmentation result of a range image of an exterior scene.

3. Merging Segmented Regions

Every range image is partitioned into sequentially labelled segmented regions, determined by the order of region extraction. The next task is to merge these scans to reconstruct a complete model of the urban structure and to generate a graphics model for it. Using range image registration systems in [12][8], partially overlapped range images are registered to form a combined 3-D scene. In this combined scene, overlapping areas between range images may

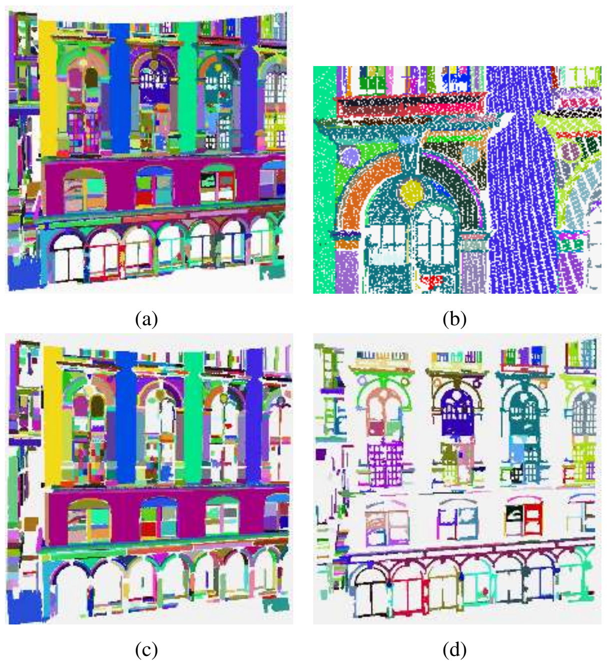


Figure 4. Segmentation results. (a) Segmented regions as colored patches. (b) Details at region boundaries. (c) Planar and smooth regions. (d) Non-smooth regions.

have different region labels, thereby causing inconsistency in displaying and modeling.

As seen from the previous figures, each segmented range image is displayed with all points colored based on their region labels: points in the same region have the same color, while points from different regions have different colors. There are overlapping regions captured in multiple scans, and their labels in different range image are assigned in each individual segmentation procedure. This causes different region labels for the same region, which is then displayed with different colors in the combined view. The mixed coloring points lead to confusion when observing regions exist in multiple scans, and result in false region boundaries for large regions that spread across a few range images. It is thus necessary to detect and unify overlapping regions.

Let us first introduce some notations and data structures in order to describe the segment merging algorithm:

1) There are n segmented scans to merge, denoted as I_i ($i = 1..n$). As mentioned earlier, we have computed the transformation matrices that register these scans under a common coordinate system of a pivot scan. The pivot scan is one of the n range scans, denoted as I_p ($1 \leq p \leq n$). Let us further denote M_i^p as the matrix to transform scan I_i to the coordinate system of scan I_p. With matrix composition, we can also select another scan I_q as the pivot scan, and calculate the transformation from all other scans to I_q, denoted as M_i^q ($1 \leq p \leq n, i = 1..n$).

2) Each range scan I_i is associated with a Z-buffer Z_i. Z_i is a 2D array defined based on the 2D rectangular area covered by the scanning laser beams, separating the space of scanning direction into bins (Fig. 5(a)). The dimension of the array is the same as the scan's resolution, ensuring that no more than a few neighboring points fall into the same bin. In each cell, a set of three values is recorded: depth, region label, and surface normal of the point that falls into this cell and is closer to the origin (Fig. 5(b)). To fill up the z-buffer, every point in the range image finds its corresponding bin, and updates the value set in that bin with its depth, region label and surface normal if the following holds true: when the bin is not filled, or the point's depth is smaller than the current depth value stored in the bin.

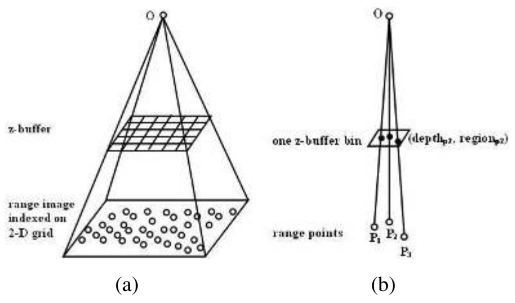


Figure 5. Generating z-buffer. (a) The range of each cell of the z-buffer is determined by the pivot scan. (b) O is the origin. P_1, P_2 and P_3 are range points that fall into one bin. Since P_2 is closest to O, P_2's information is used.

3) We also generate a transformed z-buffer Z_i^p for each range image I_i with respect to a pivot scan I_p. Similar to Z_i, Z_i^p is a 2D array where each bin is populated with point depth, region label and surface normal. However, there are two differences between Z_i^p and Z_i. First, the bins of Z_i^p are based on the 2D scanning area of I_p; second, all the points in scan I_i are transformed to the coordinate system of I_p, and the depth and surface normal of these transformed points in I_p's coordinate system (together with their original region label) are used to filled into Z_i^p. If one point falls out of the range of Z_i^p, the point is simply not considered; if multiple points fall in one bin, the one with the smallest depth is selected and its information is filled in that bin. Note that Z_i^i is equivalent to Z_i from step 2. From this point forward, we will refer to both as Z_i^i.

In the following section we first describe how the overlapping regions between two range images are combined. We then explain the merging procedure for overlapping regions between more than two range images.

3.1. Merging Two Range Images

Assume that the two range images to be merged are I_1 and I_2. I_1 is the pivot scan, and I_2 is transformed to I_1's coordinate system with transformation matrix M_2^1. Based on the 2D grid of I_1, we generate I_1's z-buffer Z_1^1, as well as I_2's transformed z-buffer Z_2^1. Since the range scanning captures the first surface point it reaches, each depth value in Z_2^1's z-buffer should be equal toor larger than the depth value in Z_1^1's corresponding bin[8][13]. If the depth and surface normal in the corresponding bins of two z-buffers are equal, we conclude that these points from two images are overlapping. The corresponding regions that these two points belong to have therefore the possibility of being one same region. If a pair of regions are both smooth regions or both non-smooth planar regions (this information was obtained in segmentation process), and they overlap in sufficient number of z-buffer bins, they are considered the same region and should be merged and labelled with the same region label. In short, the methodology consists of first identifying and then merging all the region pairs that overlap in significant number of bins. However, not all the overlapping regions should be merged, as minor segmentation error might result in small overlaps near region boundaries. With these considerations, our algorithm of merging regions is as follows:

1. Relabel image I_2 so that its region label starts from the largest region label in I_1. This way, each region from either of the two images has a unique label. Then, after generating Z_1^1 and Z_2^1, go through each bin, and compare the values in Z_1^1 and Z_2^1 to find out all the bins that have same value sets, thus forming overlapping region pairs. For each region pair, keep a record of the number of bins that contain it.

2. Only retain region pairs with overlaps satisfying either one of the following two conditions:

a) Condition 1: The number of overlapping bins reaches a certain proportion (0.01%) of point numbers of both range images; b) Condition 2: The overlapping bin count reaches a certain proportion (5% in our algorithm) of the number of bins occupied by the smaller region in the pair. Region pairs with a large overlap will satisfy condition 1, while possibly small regions, but with proportionally large overlap, will satisfy condition 2.

3. Form overlapping region sets based on all overlapping pairs. Assign a new label (the largest existing label plus 1) to each overlapping region set, resulting in overlapping regions satisfying above conditions being labelled the same.

Fig. 6 shows a range image that overlaps with Fig. 4(a). Note that the main overlapping area contains a few windows and the large side wall. The wall in Fig. 6 is separated into a few individual planes in Fig. 4(a). Fig. 7(b) shows the merging result. The wall becomes a large region, and the overlapping windows are also well merged.

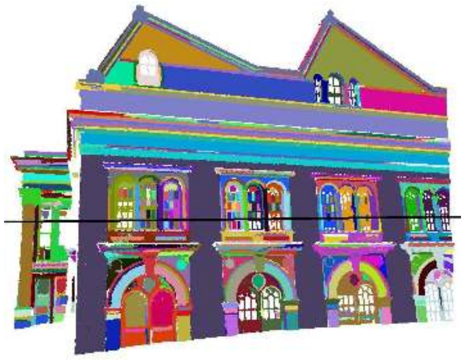
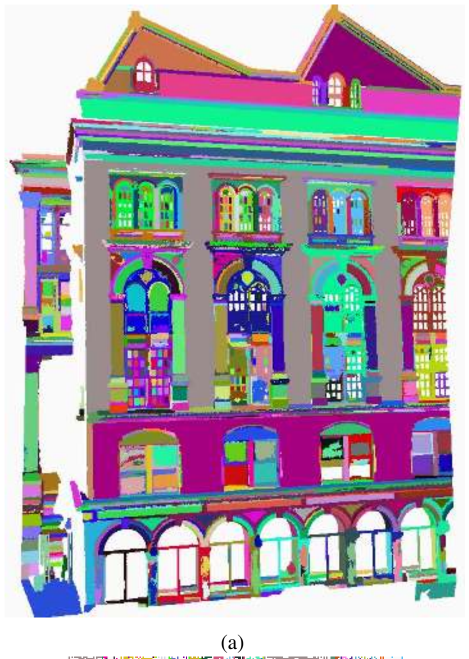


Figure 6. This range image overlaps with Fig. 7(a) at the area below the black line.



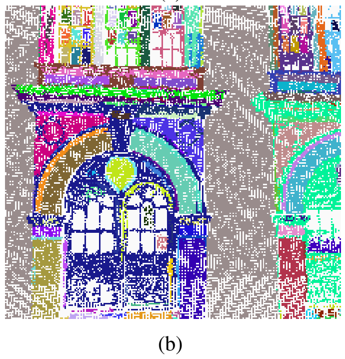


Figure 7. (a) Merging results of two range images; (b) details at overlapping area.

3.2. Merging Multiple Range Images

Merging multiple range images consists of multiple processes of merging two range images, but with added sophistication. The increased complexity results from the fact that multiple range images might overlap at the same area, and their labels need to be changed altogether. Our algorithm considers one range image as pivot at a time, and computes overlaps within this pivot scan's z-buffer area. After all range images have been used as pivots, overlapping region labels are summarized and new labels are generated for regions to be merged.

First, region labels from all range images are relabelled, so that all labels are unique. The goal is then to detect overlapping region sets from all range images (similar to the overlapping pairs in the context of merging two range images), and assign each set a new label. In order to generate overlapping region sets, we first identify all overlapping region pairs, and then combine all mutually overlapping pairs to form overlapping sets.

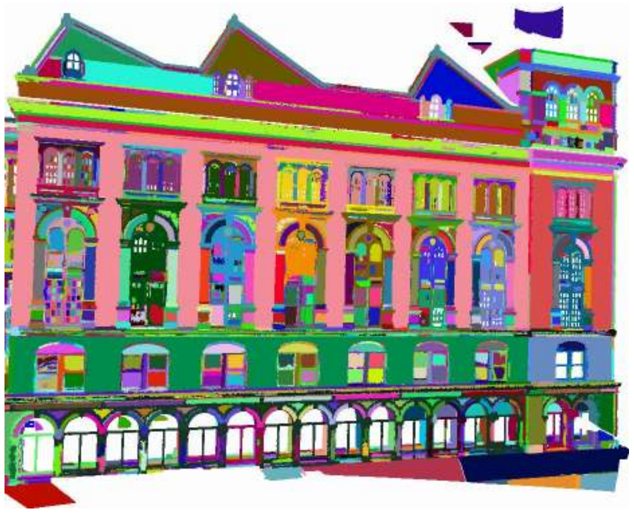
From the range image list $I_1, I_2, ...I_n$, let us first select I_1 as the pivot scan. We generate n z-buffers Z_i^1 ($i = 1..n$). By comparing corresponding bins in Z_1^1 with each of the transformed z-buffers Z_t^1 ($t = 2..n$), we accumulate the overlapping bin counts for all region pairs that overlap within the boundary of I_1. Note that in any region pair in this step, e.g. (R_1, R_2), one region R_1 is always from the pivot scan I_1, and the other region R_2 is from one of the other range images I_t.

Then, the next image I_2 is considered as the pivot scan. Overlapping counts for region pairs are accumulated similarly. After every image has been used as the pivot, all the possible overlapping regions have been counted with their overlapping size respectively. Note that for any region pair (R_1, R_2), its overlapping size is counted once when R_1's image is the pivot, and counted again when R_2's image is the pivot. This is a redundant computation. Therefore, in our algorithm, for any pivot scan I_t, we only compare Z_t^t with transformed z-buffers $Z_{t+1}^t, ..., Z_n^t$.

All the overlapping region pairs are then combined to form overlapping region sets. Using a new label for each overlapping set, the segments are merged in all range images. Note that during the merging procedure, every two scans are compared once. Therefore, the order of merging, which determines the later order of pivot selection and z-buffer comparison, is arbitrarily determined. A different order will not change the segment merging result.

4. Experiments and Conclusions

We used our method to merge two sets of range scans of large-scale urban structures in NYC, generating 360 degree interior and exterior views respectively. In this paper, for clarity of display, we only show the results of merging a subset of each set of range scans. Fig. 8 shows the merged segments from 8 exterior scans of Cooper Union building, and zoom-in details of the area where four images overlap. The execution time for segmenting each image is 2 minutes, and for merging is 4 minutes (2GHz Xeon Processor, 2GB RAM). The merged segmentation contains 1760 smooth regions and 382 non-smooth regions, and the average surface fitting error for smooth regions is 3mm. We have also generated surface mesh with Ball Pivoting Algorithm[14], and we can see that the density difference across the image boundaries diminished in the mesh surface. Fig. 9 shows the merged segments from 15 interior scans of Grand Central Station and generated surface meshes. The execution time for segmenting each image is 2 minutes, and for merging is 10 minutes. The merged segmentation contains 1393 smooth regions and 787 non-smooth regions, and the average surface fitting error for smooth regions is 4mm.


(a)

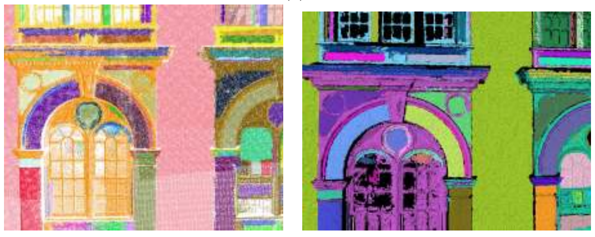
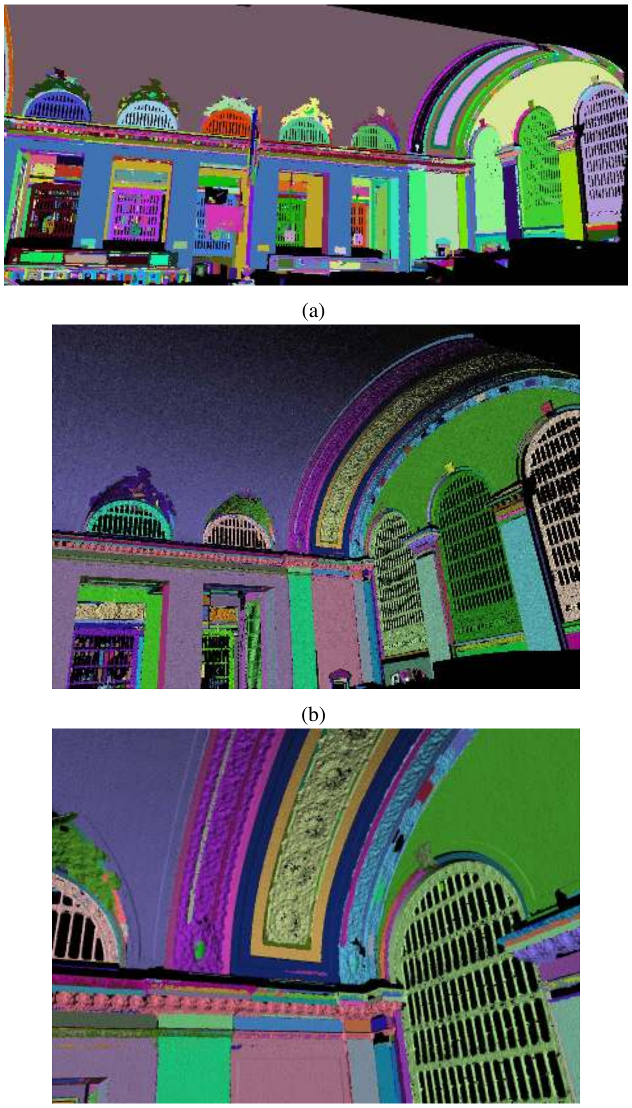

(b) (c)

Figure 8. (a) 8 scans merged. (b) Enlarged (a) at overlapping area. (c) Segment meshes (rendered with random colors).

In this paper, we have introduced a novel segmentation algorithm which accurately calculates the surface normal of every point, and then segments range images into mean-


(a)

(b)

(c)

Figure 9. (a) 15 scans merged. (b) Segment meshes. (c) Enlarged (b) at overlapping area.

ingful planar and non-planar regions. We further explained our approach of merging segmented range images, which results in a combined regions. Using these combined segmented regions, we can build 3D models. These models would enable us to represent large scale urban structures in a more descriptive way.

This work, where we explore building the semantic structures from the extracted planar and non-planar regions, is the first step in the computer vision system we envision. Our future work will be composed of two parts. The first part would consist of the analysis of the spatial relationship of extracted planar and non-planar regions, aiming at higher level scene understanding. The second part would consist of the feature extraction from each individual non-planar region, for the purpose of 3D shape recognition and object matching. This research has applications in urban planning, virtual reality, and robotic vision.

References

[1] Leica Geosystems. *http://hds.leica-geosystems.com/*.

[2] R. Unnikrishnan, J. Lalonde, N. Vandapel, and M. Hebert. Scale Selection for the Analysis of Point-Sampled Curves. *3rd Int'l Symposium on 3D Processing, Visualization and Transmission*, June 2006.

[3] N. Mitra and A. Nguyen. Estimating surface normals in noisy point cloud data. *Proc. of the 9th Annual Symposium on Computational Geometry*, 2003.

[4] K. Lee, P. Meer and R. Park. Robust Adaptive Segmentation of Range Images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, February 1998.

[5] Ranjith Unnikrishnan and Martial Hebert. Robust Extraction of Multiple Structures from Non-uniformly Sampled Data. *Proc. of the 2003 IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, October 2003.

[6] Y. Yu, A. Ferencz and J. Malik. Extracting Objects from Range and Radiance Images. *IEEE Trans. on Visualization and Computer Graphics*, Oct.-Dec. 2001.

[7] O. R. P. Bellon and L. Silva. New Improvements to Range Image Segmentation by Edge Detection. *IEEE Signal Processing Letters*, Feb. 2002.

[8] C. Chen and I. Stamos. Range Image Registration Based on Circular Features. *3rd Int'l Symposium on 3D Data Processing, Visualization and Transmission*, Jun. 2006.

[9] P. J. Besl and R. C. Jain. Segmentation Through Variable-Order Surface Fitting. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Mar. 1988.

[10] I. Stamos and P. K. Allen. Geometry and Texture Recovery of Scenes of Large Scale. *Journal of Computer Vision and Image Understanding*, Nov. 2002.

[11] P. F. U. Gotardo, O. R. P. Bellon, K. L. Boyer and L. Silva. Range Image Segmentation Into Planar and Quadric Surfaces Using an Improved Robust Estimator and Genetic Algorithm. *IEEE Trans. on Systems, Man, and Cybernetics*, Dec. 2004.

[12] C. Chen and I. Stamos. Semi-automatic Range to Range Registration: a Feature-based Method. *The 5th Int'l Conf. on 3-D Digital Imaging and Modeling*, Jun. 2005.

[13] D. F. Huber and M. Hebert. Fully Automatic Registration of Multiple 3D Data Sets. *Image and Vision Computing*, 2003.

[14] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, G. Taubin. The Ball-Pivoting Algorithm for Surface Reconstruction. *IEEE Trans. on Visualization and Computer Graphics*, Oct.-Dec. 1999.