Range Image Registration Based on Circular Features *

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

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

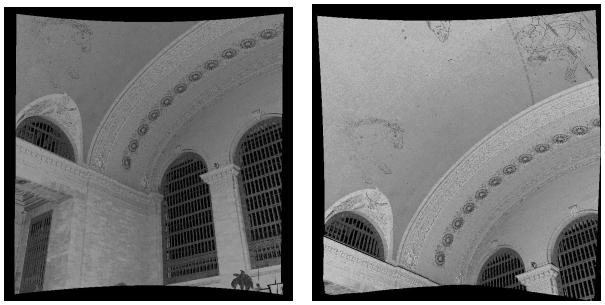
Abstract

Range sensing technology allows the photo-realistic modeling of large-scale scenes, such as urban structures. A major bottleneck in the process of 3D scene acquisition is the automated registration of a large number of geometrically complex 3D range scans in a common frame of reference. The generated 3D representations, after automated registration, are useful for urban planning, historical preservation, or virtual reality applications. Man-made urban scenes provide an abundance of linear features that can be used for the solution of the problem. Many scenes though require the utilization of non-linear primitives. This paper provides a solution of the registration problem based on the robust detection and matching of circular features from the range data sets. We present results from experiments with complex range scans from the interior of a large-scale landmark urban structure (Grand Central Terminal, NYC), where traditional methods would fail. This work is part of a larger range registration system that is based on extracted features of multiple geometric types.

1. Introduction

The photorealistic modeling of large-scale scenes, such as urban structures, requires a combination of range sensing technology with traditional digital photography. The range sensing part is critical for the accurate geometric description of the scene. A systematic and automatic way for registering 3D range scans is thus essential. This paper presents a novel method that utilizes non-linear features of circular nature extracted from range images to facilitate automated registration.

Features of this type are very common in many architectural environments (consider the range scans shown in Fig. 1 for an example). Our approach brings pairs of range scans into very accurate initial alignment making no assumptions about the relative position of the range scans. The presented method thus complements work based on linear features alone [9][10], or point signatures [8]. We envision a larger system that detects and matches features of various geometric shapes (lines, circular or elliptical arcs, spheres, cylinders, etc.).



(a) (b)

Figure 1: Two range images of the interior of a Grand Central Terminal, NYC.

A robust method that extracts distinguishable features from range images is very important for our method. Previous range image segmentation techniques include edge detection [1][2], region growing [3][4], and polynomial surface fitting [3][5]. Most of these methods provide edge maps and/or regions expressed as polynomial functions. This is useful for object modeling and reconstruction, but may not be suitable for feature matching. Our method detects precise edges and extracts geometric features with concise descriptors that make them appropriate for feature matching.

Iterative Closest Point (ICP) is one of the most popular range registration algorithms [6][7]. ICP provides very accurate results but requires a good initial guess of the registration transformation. We, on the other hand, utilize ICP as a post-processing step after our automated method brings scans into alignment. A method that does not require knowledge of an initial registration transformation is presented in [8][11] (spin images). The spin images approach does not rely on features of specific geometric type,

*Supported in part by NSF CAREER IIS-01-21239, and NSF MRI/RUI EIA-0215962.

but is sensitive to varying scan resolutions. Furthermore, the extracted point signatures have local support, the extent of which is specified by the user. Our approach detects circular features of any size as long as they exist in the range image. Thus, it does not suffer from limitations on feature support. Approaches that rely on linear features alone [9][10] provide accurate results in scenes of urban nature, but will fail in scenes that do not contain a sufficient amount of lines. Our method thus complements line-based approaches.

We are introducing a new range registration algorithm that extracts non-linear features (circles in this article) as well as lines. The proposed algorithm expands the capability of line-based algorithms to recognize more complicated geometric shapes in 3D scenes. The major steps include 3D edge detection, 3D line/3D circle extraction, and circle-based feature matching. Based on our current research, we propose a framework of registering range images based on a variety of geometric shapes and other feature descriptors.

2. 3D Edge Detection

Each range scan R_i is represented as a 2D array of 3D points $\{\mathbf{P}(k,l), k = 1 \ldots N, l = 1 \ldots M\}$[1]. Within each range image we consider 3D edges of the following two types: (a) edges caused by surface normal discontinuities (roof edges), and (b) edges caused by depth discontinuities (step edges). Step edges are further divided into edges caused by one surface occluding another (occlusion edges), and edges caused by 3D surface boundaries (boundary edges).

We briefly summarize the algorithm for detecting edges of various types. First the surface orientation change at each point is decomposed into variations along four grid directions. This grid is the 2D structured grid on which each range image is organized (as mentioned in the previous paragraph). We thus obtain four values at every 3D point, that we call directional variation values. In the structured 2D grid we form four 2D images. The intensity value at each pixel is the surface variation (we define it properly in the next paragraphs) of the corresponding 3D point. We call the four 2D images directional variation images. 2D Canny-based edge detection is performed on each image. Finally the 2D edges are combined and projected[2] back to the 3D range image space, providing the final 3D edges due to surface normal discontinuities or depth discontinuities.

Before providing more details, let us first define the notation in this paper. During the edge detection process on 2D directional variation images, 3D information such as range depth and 3D distance between range points is often needed. To minimize ambiguity, we use capital letters to denote features in 3D space, and lower case letters for 2D space. The features include points, circles, etc. For example, point P refers to a point in 3D space, and p refers to the corresponding pixel in the 2D grid image.

The directional variation images are obtained as follows: At each point P, let B_1 and B_2 be its two neighbors along one of the four grid directions (see Fig. 2(a)). The vector from P to B_1 is $\mathbf{V_1}$, and from P to B_2 is $\mathbf{V_2}$. The variation at each direction for point P is defined as $Angle(\mathbf{V_1}, \mathbf{V_2})/\pi$. This provides a value in $(0, 1]$ as the intensity value for this 2D directional variation image. Note that if the pixel p (corresponding to P) has at least one empty neighbor point on the grid, the value is set to 0, and if p is itself an empty point, the value is set to -0.1 (to detect boundary edges). Each 2D directional variation image thus emphasizes surface normal change along one direction (Fig. 2). The combination of the edges detected from them provides us with a complete set of edge points. Representative previous methods involve the estimation of surface normals first, followed by decomposition of x/y directions [4] or analysis of the angle between surface normals of neighboring points [2]. We implemented these methods as well but they are not as robust for curved surfaces for the following reasons: (1) Surface normal computation smooths out the orientation change; (2) Decomposing to x/y direction causes poor results at diagonal edges; and (3) The angle between neighboring surface normals only provides one degree of information, but the direction of change is discarded.

2D edge detection is performed on each of the four directional variation images in Fig. 2. First, Gaussian smoothing is applied to suppress noise. Then, gradients along x and y direction, g_x and g_y, are computed at each pixel using Sobel operators. With g_x and g_y we compute the gradient magnitude g. Edge direction dir at each pixel is determined by slope angle $angle = arctan(g_y/g_x)$: if $angle \in [0, \frac{\pi}{8}] \cup [\frac{7\pi}{8}, \pi]$, dir is horizontal; if $angle \in (\frac{\pi}{8}, \frac{3\pi}{8})$, dir is positive diagonal; if $angle \in [\frac{3\pi}{8}, \frac{5\pi}{8}]$, dir is vertical; and if $angle \in (\frac{5\pi}{8}, \frac{7\pi}{8})$, dir is negative diagonal.

The traditional Canny edge detection then carries out non-maximum suppression to obtain a thin edge, followed by hysteresis thresholding to output a specified amount of edge points. In our algorithm, we reverse the order of these two procedures, due to the following considerations: 1) Instead of deciding the number of edge points by ratio, we aim at finding all the points whose neighborhoods contain more significant change than expected in the high resolution range scans of large-scale urban scenes. 2) Applying thresholding in the last step causes discontinuous edges, but we prefer to keep edges as continuous as possible, for the purpose of accurate edge linking and circle fitting in the later phases. So, our algorithm uses a generous threshold 0.35 (allowing the angle change of 5 degrees[3] in any direction)

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

[2] Each pixel $p(k,l)$ in the grid-point image corresponds to a 3D point $\mathbf{P}(k,l)$.

[3] We know that at each non-boundary point, the directional variation

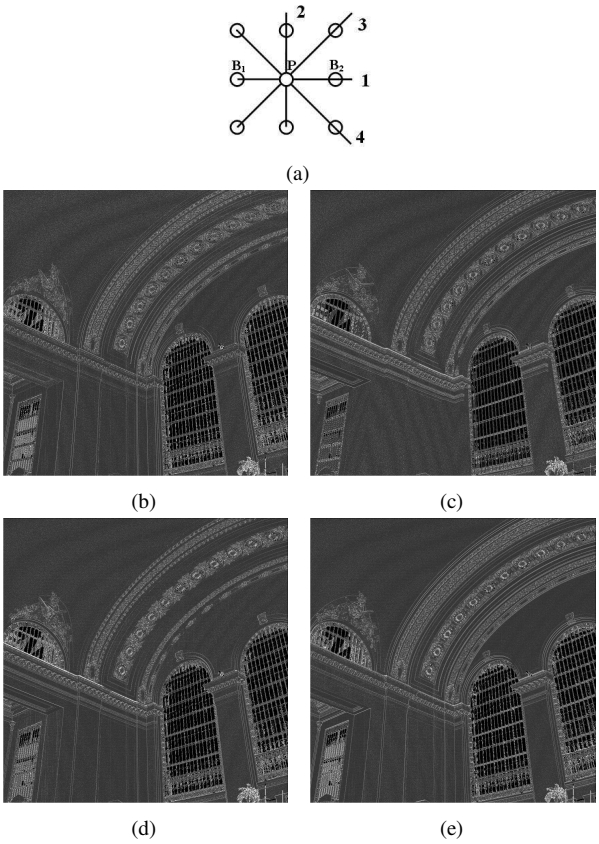


Figure 2: Directional variation images. (a) Four grid directions: 1-Horizontal, 2-Vertical, 3-Positive Diagonal, 4-Negative Diagonal. B_1 and B_2 are P's neighbors along direction 1. (b)–(e) Directional variation images of Fig. 1(a) in the above four directions respectively. Brighter points have larger variation values, and darker points have smaller variation values (see text). The intensities are slightly scaled for best display. Note that each type of directional variation image responds strongest to edges perpendicular to the direction of computing variation.

for edge magnitude, followed by non-maximum suppression.

To this point, we have detected all roof and step edges. However, occlusion edges need to be identified and only the foreground edges should be kept in order to reflect the true geometry of the scene (similar to the shadows in 2D images). In our algorithm, non-maximum suppression votes off edge points based on magnitude, regardless of whether it is on a foreground surface or a background surface (Fig. 3). We therefore find and remove all background edge points, while add back those foreground edge points voted off by a background neighbor. To find these points, we map the 2D edge points back to 3D range scan and label a 3D point P if its corresponding pixel p is an edge point. For an edge point P, let B_1 and B_2 be its two neighbors perpendicular to its edge direction.[4] If $\mathit{Distance}(P, B_1) >> \mathit{Distance}(P, B_2)$ and $\mathit{Depth}(B_1) << \mathit{Depth}(P)$, then B_1 is a foreground point, and P is a background edge point. In that case P is labeled as non-edge, and B_1 is labeled as an edge point (Fig. 3(a)). Notice that in the case of Fig. 3(b), when P already has two other neighbors on both sides of B_1 being edge points, it is unnecessary to add B_1 to form a thick edge and consequently be fitted to two lines. By symmetry, edges of other directions can be analyzed and processed similarly.

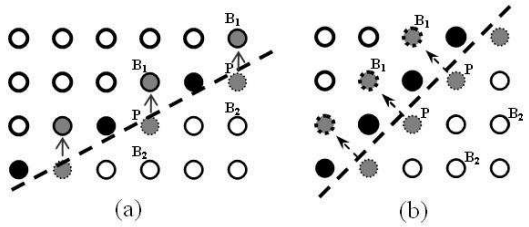


Figure 3: Occlusion edge and its correction. Dash line: real occlusion edge. Black points: foreground edge points. Gray points below the edge: background edge points to be removed from edge map. Gray points above the edge: foreground points to be added as edge points. In (a), they are added into edge; in (b), they stay as non-edge since the edge is already connected and traceable.

Another type of edge points to be eliminated from every directional variation image are the corner points. These points appear as high curvature points in 3D space. There are two reasons for removing these points: 1) By removing corner points we break the connections between edges of different directions, thereby simplifying edge linking and fitting (e.g. the corner points connecting edges in Fig. 5(a)). 2) Many delicate structures are detected as edge points, but they do not provide information on major geometric shapes. These clusters of high curvature points sometimes show interesting patterns. Although discarded in our current algorithm, they could be utilized for region matching in the future (e.g. the rosette in Fig. 5(a)). We detect corner points by applying Harris corner detector to every edge point, and testing whether there are more than one principle directions formed by all edge points in its local neighborhood.

The next step is the combination of four edge maps by taking the union of all edge points. From the combined edge map, isolated edge points are deleted, and short gaps (1 or 2 pixels) are filled along the local edge direction. Then continuous edge points are linked by tracing along edge di-

value $v \in (0, 1]$. The absolute difference between any two points is thus $d_v \in (0, 1]$. Let us set the threshold to be $5^o \approx 0.0873$. Sobel operator enlarges the difference by a factor of at most 4, which gives 0.35 for g_x and g_y. Since only one of them could reach that maximum value, the threshold of magnitude is decided to be the same for both.

[4] In here as well as later in this paper, "edge direction at a point" means the 2D edge direction of its corresponding pixel.

rections. The edge linking utilizes the structured grid on which the range image is represented for resolving neighbors. Only long edges (30 points or more) are being kept for later processing. The final combined edge map is shown in Fig. 4(e). Fig. 5 shows the details in areas of a corner and a circular window.

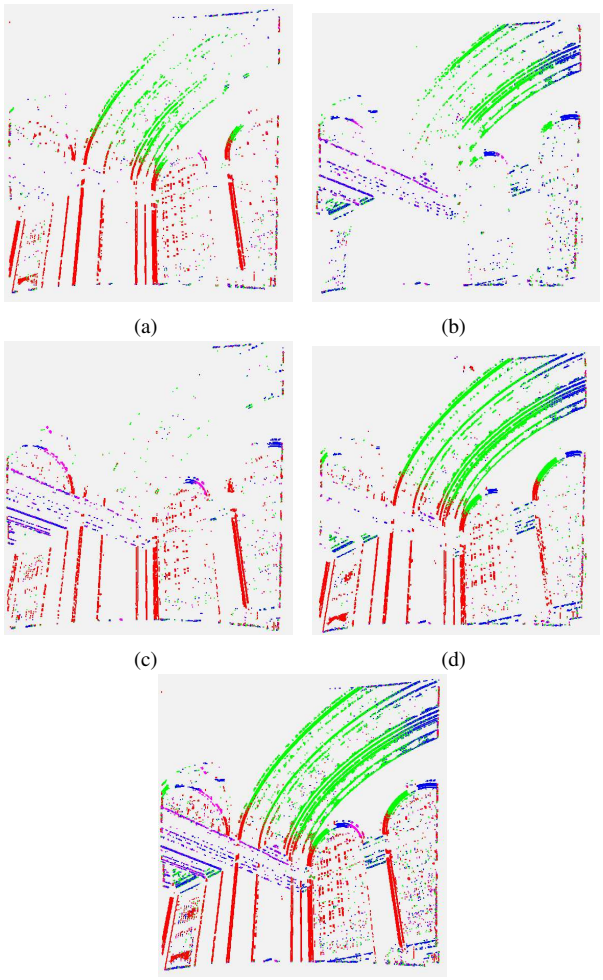


Figure 4: Edge points of range images of Fig. 1(a). Note that the color at each point (red/green/yellow/magenta) indicates its edge direction (see text), hence the same point usually has the same color in the four edge images. (a)-(d) Edge points detected from Fig. 1(b)-(e) respectively. (e) Combined edge image from (a)-(d).

3. 3D Circle Extraction

Each linked edge describes a curve in 3D space. For the purposes of this work we are interested in circular features, which are non-linear planar curves. Therefore we first remove linear ones, and then keep only planar ones for circle fitting. For each linked edge from Fig. 4(e), its best-fit line

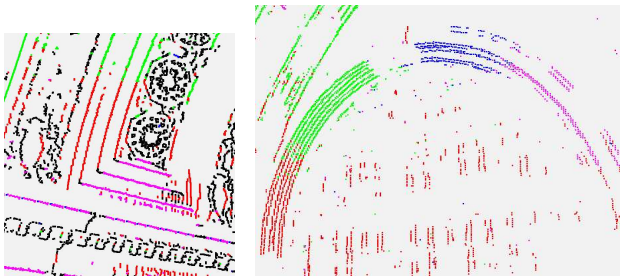


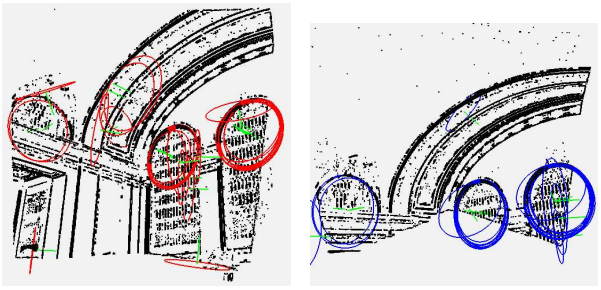
Figure 5: Zoom-in on edge points. (a) Black points are corner points (to be removed from consideration). (b) circular edges along the window frame.

direction $\mathbf{V_{max}}$ and best-fit plane normal $\mathbf{V_{min}}$ are computed. A curve is considered linear if the line fitting error (average distance of all points to the fitted line) is less than a threshold 0.03m (approximate distance between two neighboring 3D range points). For nonlinear curves, the average perpendicular distance of all points to the fitted plane is used to discard 3D curves that are non planar (a generous threshold of 0.5m is used). For each of the remaining planar curves, all points are projected onto their fitted plane. After this process, the 3D curve becomes a set of 2D points in the 2D space Π of the fitted plane. Circle fitting is done in this space.

Taking the common approach of least square fitting[5], we compute the center (a, b) and radius r of the circle by finding an approximate null-vector of a $n \times 4$ design matrix, where n is the number of points on the curve. Consider the circle function $(x-a)^2 + (y-b)^2 - r^2 = 0$. It can be written as $x^2 + y^2 - 2ax - 2by + a^2 + b^2 - r^2 = 0$. Let (x_i, y_i) be the 2D coordinates of all points $p_i (i = 1, ..., n)$ on the curve. Then the circle equation for all points can be expressed as as a multiplication of the $n \times 4$ matrix $M = [M_1 \quad M_2 \quad ... \quad M_n]^T$ where $M_i = [x_i^2 + y_i^2 \quad -2x_i \quad -2y_i \quad 1]$ (for $i = 1, ..., n$), with unknown vector $[1 \quad a \quad b \quad a^2 + b^2 - r^2]^T$. The null-vector of the design matrix, computed by SVD, provides the solution. Finally, the circle fitting error is computed as $c_{err} = \sqrt{\frac{\Sigma_i^n (distance(p_i - center) - r)^2}{n}}$. The ratio $(\frac{c_{err}}{r})$ must fall below a threshold (0.02) to verify that the planar 3D curve is a circular arc. Finally, the center of the fitted circle is converted back from Π to the 3D space. We now have three parameters to represent each oriented 3D circle: 3D center point, radius, and plane normal. Fig. 6 shows all the circles with radii between 3.0m and 5.0m. These are the ones most useful for matching in the next step. In the execution, we detect all circles with radii between 2.0m and

[5] A 3D hough transform method will be inefficient, since the radii of circles are unknown and may vary wildly.

20.0m.



(a) (b)

Figure 6: Circles extracted from the range images of Figs. 1(a) and 1(b), respectively. All edge points are in black, and all fitted circular arcs are represented with colored full circles, with green lines indicating their normals. Note that three circular windows are detected in both images. The images are rotated to the best angle to observe all circles. Therefore some of them appear as ellipses due to the viewing direction.

4. Feature Matching

After the oriented 3D circles are extracted from range images, possible matchings between them are hypothesized. The computed transformations are graded using surface consistency[11] and average point distance in the overlapping area between the scans.

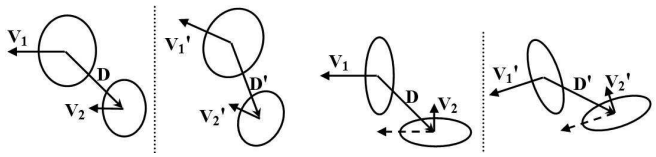
Similarity of radii, orientation and relative position between pairs of circles is utilized in the matching phase. In particular, consider a pair of circles (C_1, C_2) from scan R_1 and another pair of circles (C_1', C_2') from scan R_2. The pairs would be considered as matching iff

1. Circles C_1, C_1' have equal radii within a threshold (maximum difference of 0.1m);
2. Circles C_2, C_2' have equal radii within a threshold (maximum difference of 0.1m);
3. The distance between the centers of C_1, C_1' equals the distance between the centers of C_2, C_2' within a threshold (maximum difference of 0.2m);
4. The angle between the normals of C_1, C_1' equals the angle between the normals of C_2, C_2' within a threshold (maximum difference of 10^o).

Furthermore, consider a pair of circles (C_1, C_2) from scan R_1 and another pair of circles (C_1', C_2') from scan R_2 that could be considered a valid match according to the previous definitions. A transformation (rotation R followed by a translation T) can be computed by converting the correspondence of a pair of oriented circles to a pair of 3D oriented lines. This approach leads to a robust transformation computation, since it is based on relative position and orientation of the features rather than exact position and orientation of each feature. In particular, two cases are considered:

Case 1: Circles (C_1, C_2) have parallel normals $\mathbf{V_1}$ and $\mathbf{V_2}$ (the same is true for the normals $\mathbf{V_1'}$ and $\mathbf{V_2'}$ of circles (C_1', C_2')) (Fig. 7(a)). Let us consider the oriented line $\mathbf{D}$ that connects the centers of (C_1, C_2) and the oriented line $\mathbf{D'}$ that connects the centers of (C_1', C_2'). If $\mathbf{D}$ is not parallel to $\mathbf{V_1}$ (that means that $\mathbf{D'}$ is not parallel to $\mathbf{V_1'}$), the match of the oriented line $\mathbf{D}$ with $\mathbf{D'}$ and $\mathbf{V_1}$ with $\mathbf{V_1'}$ can provide a reliable transformation (closed form formula [9]). Otherwise ($\mathbf{D}$ is parallel to $\mathbf{V_1}$) a reliable transformation can not be computed.

Case 2: Circles (C_1, C_2) do not have parallel normals $\mathbf{V_1}$ and $\mathbf{V_2}$ (the same is true for the normals $\mathbf{V_1'}$ and $\mathbf{V_2'}$ of circles (C_1', C_2')) (Fig. 7(b)). Then, the two pairs of oriented lines $(\mathbf{V_1}, \mathbf{V_2})$ and $(\mathbf{V_1'}, \mathbf{V_2'})$ are used for the computation of a reliable transformation (closed form formula [9]).



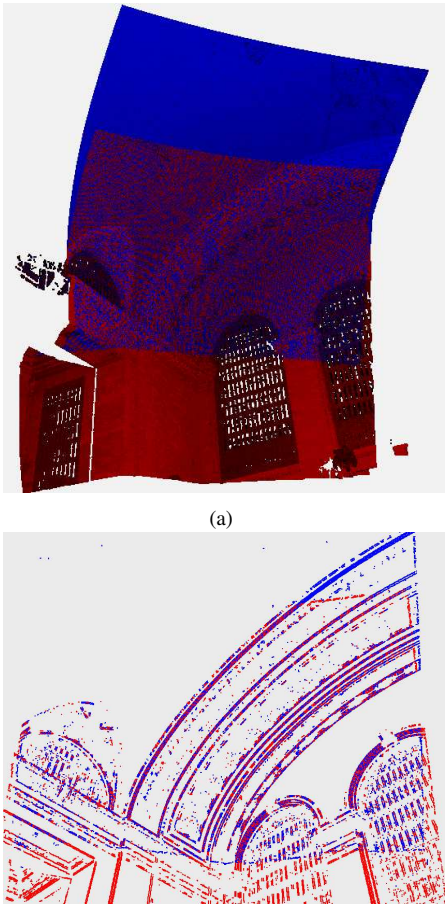
(a) (b)

Figure 7: Two cases of matching circle pairs. The dash line separates scan R_1 from R_2. The radii and relative position of the two circles from R_1 must be similar to those from R_2. (a) Case 1: two circles have parallel normals. V_1, D and $V_1', D'$are used to compute transformation. (b) Case 2: two circle normals are not parallel. V_1, V_2 and $V_1', V_2'$are used to compute transformation.

From each valid matching circle pairs, a candidate transformation is computed as described above. Each transformation is verified for correctness as follows. Based on the fact that overlapping images are captured from nearby positions, we discard all rotation matrices with diagonal elements smaller than 0.7 (allowing 45^o tilting of the range scanner about each of its $x/y/z$ axes). Note that this step reduces the number of possible transformations and thus speeds up the algorithm, but is not otherwise necessary. Then we test whether the transformation causes surface inconsistency (see next paragraph). Finally, from all verified transformations, the one achieving the smallest average distance between overlapping range points is chosen as the best.[6]

[6]Note that an approach similar to association graphs [12] would generate a very large search space.

Surface inconsistency between two range images R_1 and R_2 is decided in summary as follows (method described in [11]). After image R_1 is transformed to the coordinate system of the image R_2, consider a ray cast from R_2's center of projection toward the overlapping area of the two images. The following two situations should not happen: (1) along the ray there is a point from R_1 but no point from R_2 (Free Space Violation); (2) along the ray there is a point from R_1 significantly in front of a point from R_2 (Occupied Space Violation). We follow the suggestion in [11] and implement surface inconsistency detection with two z-buffers, one for each scan. The coordinates on both z-buffers are based on the 2D grid of scan R_2 (999×999 points), but sampled into 250×250 bins. One point falls into one bin if the ray from R_2's center of projection toward the point passes through that bin. For each bin in R_2's z-buffer, among all points that fall into it, the smallest distance from the center of projection is recorded. R_1's z-buffer is filled in the same manner, but with points from transformed R_1. Now comparing these two z-buffers we are able to detect cases of FSV and OSV. Only at those bins where distance values from two images differ no more than a certain threshold (2.0m), i.e. overlapping regions, we record the difference, and take the average of all such differences to evaluate the goodness of the transformation. We choose the threshold 2.0m, because this value is a loose upper bound for point distance. Setting the threshold too low would limit the comparability of two transformations, since only those few points already very close are used for distance computation. Therefore the difference between good transformations and bad ones is not obvious. In our experiment, we also set a loose threshold (10%) on overlapping area to filter out invalid transformations.



(a)

(b)

Figure 8: Registered images of Figs. 1(a) and 1(b). They are colored to highlight overlapping area. (a) All image points. (b) Edge points at overlapping area.

5 Experimental Results

Our automated method is used for registration of the interior scans of Grand Central Terminal in NYC (a large-scale landmark urban structure). The best transformation of the two corner scans of Fig. 1 provides a registration error (average point distance in the 55.7% overlapping area) of 0.95cm. Within a few iterations of ICP an optimal transformation with a registration error of 0.90cm is obtained (Fig. 8).

Also, we registered other scans of this hall with the same technique. The entire hall is roughly in rectangular shape with an arched ceiling. Fig. 9 shows a few typical scans on the front wall ((a)(c)) and the side wall ((e)(g)), together with circles extracted from them. Note that the lower parts of the walls (e.g.(c)(g)) contain lines and planes, and are therefore registered with the linear-feature based technique of [10]. The upper regions with very few linear features, e.g.(a)(e), are registered with their lower neighboring scans (c)(g) respectively, by matching overlapping circular windows.

In Fig. 10, registered edge points from 23 scans are visualized. There are another 14 scans not shown for clarity of presentation. Among all 37 scans, 20 of them are lower parts registered with lines, 13 of them are the upper parts registered with their lower neighbor scans based on overlapping circular windows. Three scans are manually registered, because they are cylindrical ceiling patches without any distinguishing geometric shape information. In Table 1 we report the performance of 13 registrations based on circles. When registering the last two pairs, a long execution time is experienced due to a large number of valid transforms from the precisely extracted circles around the window frame (as in Fig. 9(b)(d)). To avoid unnecessary computations, we set the program to terminate when the average distance falls below 0.03cm (approximate distance between two neighboring points). The values in columns *RT*, *D_mat*,

Time are therefore recorded up to the point when an accurate enough result is reached. In Fig. 11, more screen shots are shown, including registered scenes, edge points, and part of a 3D model constructed using the registered range points.

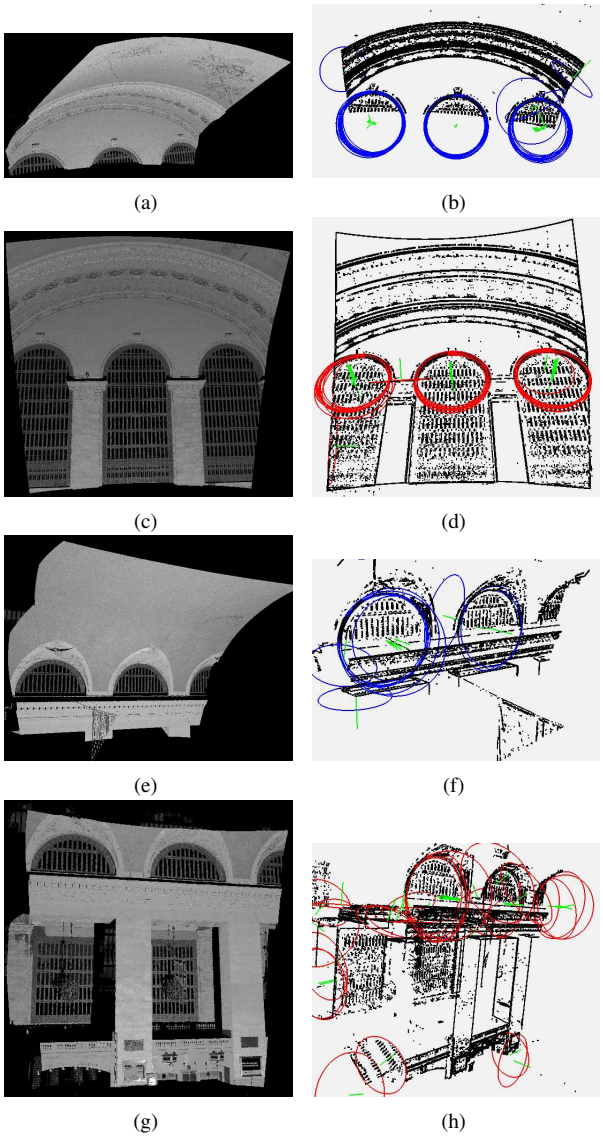


Figure 9: Four side wall scans, extracted edges and fitted circles.

Circles	RT	D_mat	D_opt	Overlap	Time
26x24	544	0.95cm	0.90cm	55.7%	6min
24x39	980	1.11cm	1.00cm	29.1%	9min
21x39	15	3.42cm	1.28cm	16.1%	1min
24x20	748	2.13cm	0.77cm	38.4%	7min
13x30	126	2.01cm	0.84cm	28.9%	2min
21x26	534	1.68cm	0.90cm	35.5%	6min
23x11	29	4.26cm	0.87cm	28.7%	1min
14x31	18	2.65cm	0.93cm	27.8%	2min
31x13	58	2.34cm	0.98cm	23.0%	2min
37x26	67	3.83cm	0.87cm	37.2%	2min
23x35	310	1.20cm	0.84cm	26.7%	7min
49x41	3054	2.81cm	1.02cm	38.7%	58min
50x38	931	1.83cm	0.92cm	44.6%	10min

Table 1: Experimental results. Meanings of columns: Number of circles from two images to register; Number of candidate transformations; Average point distance from best transformation; Average point distance after ICP optimization; Overlapping area of two scans; Execution time, including circle extraction and matching (on a Linux-based 2GHz Xeon-Processor with 2GB of RAM).

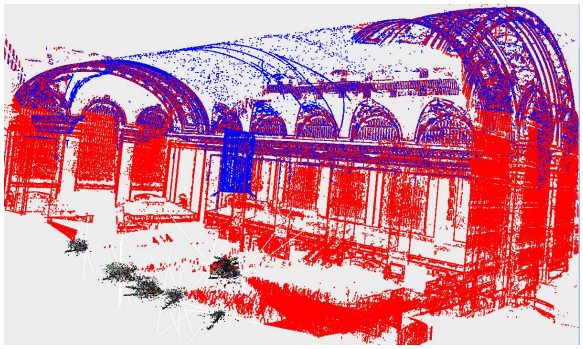


Figure 10: Registered edges of 23 scans, with color indicating scans from upper parts (blue) and lower parts (red). The other 14 scans are not displayed for clarity of presentation; these scans compose a side wall closer to our point of view and symmetric to the wall being displayed.

6. Conclusions and Future Work

We have introduced novel algorithms and results on registering range images based on circular feature extraction and matching. This work complements range scan registration system based on linear features or point signatures. The edge detection provides precise 3D edges, and the circle fitting extracts circular edges feasible for feature matching. Finally, the correct transformation is selected from many candidates by point-based verification and distance minimization. The registration achieves cm-level accuracy. The computation time is in the order of minutes.

In the future, we would like to improve the estimation of thresholds in our algorithm. Although they are reasonably estimated based on our range scan data, it would be a great advancement if they could be decided from any provided range data automatically. Thresholds determining corner points, straight lines, and circles are rather important. They affect how many edge points and features are used for matching: too few may cause inability to register, while too many may dramatically slow down the registration process. This is a common problem in feature detection.

All of our current achievements are part of a larger project of range image registration based on multiple geometric features. Since there are certainly scenes without robust circular features, our next step would involve including more geometric shapes into our segmentation analysis, such as elliptical curves or spherical and cylindrical surfaces. If none of the preliminary geometric shapes can be detected, we would consider generating statistical descriptors for arbitrary feature areas selected either automatically or manually.

References

[1] M. A. Wami and B. G. Batchelor. Edge-Region-Based Segmentation of Range Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Mar. 1994.

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

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

[4] K. Pulli and M. Pietikinen. Range Image Segmentation Based on Decomposition of Surface Normals. *Proceedings of the Scandinavian Conference on Image Analysis*, 1993.

[5] D. Marshall, G. Lukacs and R. Martin. Robust Segmentation of Primitives from Range Data in the Presence of Geometric Degeneracy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Mar. 2001.

[6] P. J. Besl and N. D. McKay. A Method for Registration of 3D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Feb. 1992.

[7] S. Rusinkiewicz and M. Levoy. Efficient Variants of the ICP Algorithm. *The 3rd International Conference on 3-D Digital Imaging and Modeling*, Jun. 2001.

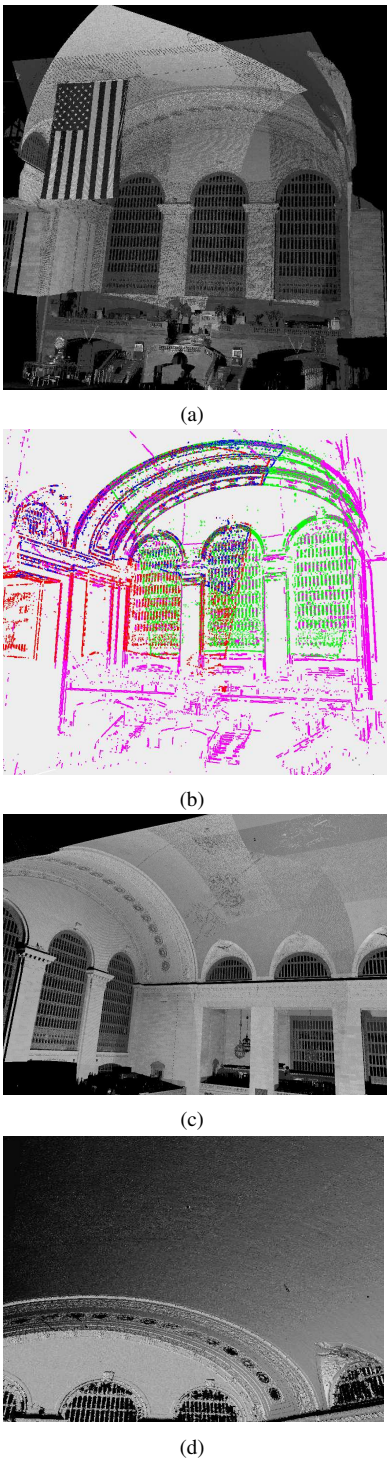
[8] A. E. Johnson. Spin-Images: A Representation for 3-D Surface Matching. *Ph.D. thesis, Carnegie Mellon University*, Aug. 1997.

[9] I. Stamos and M. Leordeanu. Automated Feature-Based Range Registration of Urban Scenes of Large Scale. *IEEE International Conference of Computer Vision and Pattern Recognition*, Jun. 2003.

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

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

[12] A. P. Ambler, H. G. Barrow, C. M. Brown, R. M. Burstall, and R. J. Popplestone. A Versatile Computer-Controlled Assembly System. *The 3rd International Joint Conference on Artificial Intelligence*, 1973.



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

Figure 11: Registration results. (a) Four out of the 37 automatically registered scans shown for clarity. (b) Edge points of (a). Four colors represent edge points from four scans. (c) Eight out of the 37 automatically registered scans shown for clarity. (d) 3D model generated with Ball Pivoting Algorithm. The smooth ceiling implies the registration is tight and seamless.