2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

RGB-D Edge Detection and Edge-based Registration

Changhyun Choi, Alexander J. B. Trevor, and Henrik I. Christensen
Center for Robotics & Intelligent Machines
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332, USA
{cchoi,atrevor,hic}@cc.gatech.edu

***Abstract*— We present a 3D edge detection approach for RGB-D point clouds and its application in point cloud registration. Our approach detects several types of edges, and makes use of both 3D shape information and photometric texture information. Edges are categorized as occluding edges, occluded edges, boundary edges, high-curvature edges, and RGB edges. We exploit the organized structure of the RGB-D image to efficiently detect edges, enabling near real-time performance. We present two applications of these edge features: edge-based pair-wise registration and a pose-graph SLAM approach based on this registration, which we compare to state-of-the-art methods. Experimental results demonstrate the performance of edge detection and edge-based registration both quantitatively and qualitatively.**

I. Introduction

Visual features such as corners, keypoints, edges, and color are widely used in computer vision and robotic perception for applications such as object recognition and pose estimation, visual odometry, and SLAM. Edge features are of particular interest because they are applicable in both textureless and textured environments. In computer vision, edges features have been used in a variety of applications, such as tracking and object recognition. Recently, RGB-D sensors have become popular in many applications, including object recognition [1], object pose estimation [2], [3], and SLAM [4], [5], but edge features have so far seen limited use in the RGB-D domain. In this paper, we present an edge detection method for RGB-D point clouds, and explore the application of these features for registration and SLAM.

Dense point cloud registration methods such as Iterative Closest Point (ICP) [6] are omni-present in SLAM and geometric matching today. The runtime of these algorithms scales with the number of points, often necessitating downsampling for efficiency. In many environments, the bulk of the important details of the scene are captured in edge points, enabling us to use only these edge points for registration rather than a full point cloud or a uniformly downsampled point cloud. Edge detection can be viewed as a means of intelligently selecting a small sample of points that will be informative for registration. We will demonstrate that this approach can be both faster and more accurate than alternative approaches.

In 2D grayscale images, edges can be detected from image gradient responses that capture the photometric texture of a scene, as is done by the Canny edge-detector [7]. In addition to traditional images, RGB-D cameras also provide 3D shape information, enabling edge detection using both 3D geometric information and photometric information. We present detection methods for several types of edges that occur in RGB-D data. Depth discontinuities in the 3D data produce two related types of edges: *occluding* and *occluded*. Another type of 3D edge occurs at areas of high-curvature where surface normals change rapidly, which we call *high curvature* edges. Edges detected on the 2D photometric data can also be back-projected onto the 3D structure, yielding *RGB* edges.

II. Related Work

Edges have been widely used since the early age of computer vision research. The Canny edge detector [7] is one of the most widely employed methods to find edges from 2D images due to its good localization and high recall. Edges were used to find known templates in a search image via the chamfer distance [8], [9] and further employed to model-based visual tracking [10], [11] and object categorization [12], [13].

3D edge detection has been mainly studied in computer graphics community. Most of work has found edges or lines from polygonal mesh models or point clouds. Ohtake *et al.* [14] searched ridge-valley lines on mesh models via curvature derivatives on shapes. While this work requires polygonal meshes, several approaches [15], [16] found crease edges directly from a point cloud. Although these approaches could detect sharp 3D edges from 3D models, they were computationally time-consuming because they relied on expensive curvature calculation and neighbor searching in 3D. Hence these approaches may not be an ideal solution for robotic applications where real-time constraints exist.

In 3D point cloud registration, the Iterative Closest Point (ICP) algorithm [6] is the best known technique to align one point cloud to another one. Several efforts have enhanced the ICP algorithm by considering two sets of correspondences to improve the point matching procedure [17] and by introducing a measure of rotational errors as one of the distance metrics [18]. The most recent state-of-the-art technique was shown by Segal *et al.* [19] who presented a probabilistic approach for plane-to-plane ICP. While these techniques were originally designed for pair-wise registration, it is possible to align multiple scans via the pair-wise alignment [20].

For large scale registration applications such as SLAM and 3D reconstruction, it is often required to perform a global optimization over multiple scans. Recently, several popular software packages have been released to solve such non-linear optimization problems, including the g2o library [21], the Google Ceres Solver [22], and GTSAM [23]. SLAM pose-graphs have been well studied in the literature. The 6D SLAM [24] is one example of a pose-graph approach using the ICP algorithm. Pose-graphs can also be constructed by using features to compute relative transforms between landmark measurements, as was done by Pathak *et al.* [25] using planar features. Pose-graphs using RGB-D cameras have also been studied previously. One approach is described in [26], which making use of SURF keypoints detected in 2D, and then back-projected onto the 3D structure. These are then used to compute a relative pose between frames via RANSAC, refined by ICP, and optimized globally via pose graph optimization using HOGMAN [27]. A related approach was proposed by Henry *et al.* [4], which takes a visual odometry approach between sequential poses, and uses SIFT keypoint matches optimized using sparse bundle adjustment to compute loop closure constraints between these frames. While these approaches were based on the sparse features, Newcombe *et al.* [28] showed a dense approach called KinectFusion. It builds a Truncated Signed Distance Function (TSDF) [29] surface model of the environment online, and the current camera pose is computed relative to this by using point-to-plane ICP. This approach relies heavily on access to high-end GPUs, which may not be realistic for some mobile robots.

We present an efficient 3D edge detection algorithm from an RGB-D point cloud by exploiting the organized structure of RGB-D images. Our approach does not rely on time-consuming curvature derivatives and thus is very efficient. We also examine the performance of pair-wise registration using our edge features and compare with several state-of-the-art approaches. The edge-based pair-wise registration is further applied to an RGB-D SLAM system which is based on the efficient incremental smoothing and mapping [30]. To the best of our knowledge, this work is the first effort to find 3D edges from organized RGB-D point clouds and apply these edges to 3D point cloud registration. This paper is organized as follows. We introduce our edge detection from organized point clouds in Section III, including edges from depth discontinuities in Section III-A, and edges from photometric texture and geometric high curvature regions in Section III-B. The usage of these edge features for pair-wise registration is explained in Section IV, and usage in SLAM is described in Section V. Quantitative and qualitative results are presented in Section VI, followed by conclusions in Section VII.

III. Edge Detection from Point Clouds

In this section, we describe our edge detection approach for RGB-D point clouds. Geometric shape information from the depth channel and photometric texture information from the RGB channels are both considered to detect reliable 3D edges. For shape information, we exploit depth discontinuities and high curvature regions to search for salient geometric edges. We also use the RGB image to detect 2D edges, which we back-project to get 3D points. Since the point clouds from RGB-D sensors are organized as images (rows and columns of pixels), neighbor search is done with the row and column indices instead of performing a time-consuming 3D search, as is necessary for general unorganized point clouds. Algorithm 1 shows the procedure of our edge detection which takes RGB color data $\mathbf{C}$ and depth data $\mathbf{D}$ as input and returns edge labels $\mathbf{L}$. Detailed explanations of each edge type are provided below.

Algorithm 1: RGB-D Edge Detection $(\mathbf{C}, \mathbf{D})$

```
Input: $\mathbf{C}, \mathbf{D}$
Output: $\mathbf{L}$
Params: $\tau_{rgb_\perp}, \tau_{rgb_\top}, \tau_{hc_\perp}, \tau_{hc_\top}, \tau_{dd}, \tau_{search}$
$\mathbf{I} \leftarrow$ RGB2GRAY($\mathbf{C}$)
$\mathbf{E}_{rgb} \leftarrow$ CannyEdge($\mathbf{I}, \tau_{rgb_\perp}, \tau_{rgb_\top}$)
$\{\mathbf{N_x}, \mathbf{N_y}, \mathbf{N_z}\} \leftarrow$ NormalEstimation($\mathbf{D}$)
$\mathbf{E}_{hc} \leftarrow$ CannyEdge($\mathbf{N_x}, \mathbf{N_y}, \tau_{hc_\perp}, \tau_{hc_\top}$)
$\{H, W\} \leftarrow$ size($\mathbf{D}$)
for $y \leftarrow 1$ to $H$ do
    for $x \leftarrow 1$ to $W$ do
        if $\mathbf{D}(x,y) = Nan$ then
            continue
        $\mathbf{n} \leftarrow$ 8-Neighbor($\mathbf{D}, x, y$)
        $invalid \leftarrow 0$
        for $n \leftarrow 1$ to $8$ do
            if $\mathbf{n}(n) = Nan$ then
                $invalid \leftarrow 1$
                break
            else
                $\mathbf{d}(n) \leftarrow \mathbf{D}(x,y) - \mathbf{n}(n)$
        if $invalid$ = $0$ then
            $\{\widehat{\mathbf{d}}, \widehat{idx}\} \leftarrow$ max($abs$($\mathbf{d}$))
            if $\widehat{\mathbf{d}} > \tau_{dd} \cdot \mathbf{D}(x,y)$ then
                if $\mathbf{d}(\widehat{idx}) > 0$ then
                    $\mathbf{L}(x,y) \leftarrow$ OCCLUDED_EDGE
                else
                    $\mathbf{L}(x,y) \leftarrow$ OCCLUDING_EDGE
        else
            $\{dx, dy\} \leftarrow$ SearchDirection($\mathbf{D}, x, y$)
            $\mathbf{L}(x,y) \leftarrow$ BOUNDARY_EDGE
            for $s \leftarrow 1$ to $\tau_{search}$ do
                $\tilde{x} \leftarrow x + \lfloor s \cdot dx \rfloor$
                $\tilde{y} \leftarrow y + \lfloor s \cdot dy \rfloor$
                if $\mathbf{D}(\tilde{x}, \tilde{y}) \neq Nan$ then
                    $d \leftarrow \mathbf{D}(x,y) - \mathbf{D}(\tilde{x}, \tilde{y})$
                    if $abs$ $(d) > \tau_{dd} \cdot \mathbf{D}(x,y)$ then
                        if $\mathbf{d} > 0$ then
                            $\mathbf{L}(x,y) \leftarrow$ OCCLUDED_EDGE
                        else
                            $\mathbf{L}(x,y) \leftarrow$ OCCLUDING_EDGE
        if $\mathbf{E}_{rgb}(x,y) = 1$ then
            $\mathbf{L}(x,y) \leftarrow$ RGB_EDGE
        if $\mathbf{E}_{hc}(x,y) = 1$ then
            $\mathbf{L}(x,y) \leftarrow$ HIGH_CURVATURE_EDGE
```

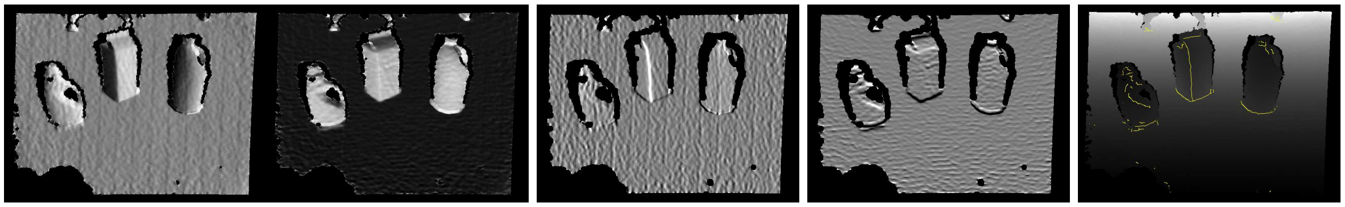
(a) $\mathbf{N_x}$ (b) $\mathbf{N_y}$ (c) $\mathbf{G_x}$ (d) $\mathbf{G_y}$ (e) High curvature edges

Fig. 1: **Detecting high curvature edges.** We employ a variant of Canny edge detector which takes the $\mathbf{x}$ and $\mathbf{y}$ components of normals, $\mathbf{N_x}$ and $\mathbf{N_y}$, as input images. The first-order normal gradient images, $\mathbf{G_x}$ and $\mathbf{G_y}$, are then obtained by applying the Sobel operator on each input image. Following the non-maximum suppression and the hysteresis thresholding, our method returns the high curvature edges.

A. *Occluding, Occluded, and Boundary Edges*

The depth channel provides reliable geometric information for the scene. To detect reliable edges in the depth data, we search for depth discontinuities, which are abrupt local changes in depth values. From these depth discontinuities, three types of edges are detected: *occluding*, *occluded*, and *boundary* edges. Occluding edges are depth discontinuous points on foreground objects in a given point cloud, whereas occluded edges are depth discontinuous points on the background. To determine these edge points, local 8-neighbor search is performed via $8-\texttt{Neighbor}(\mathbf{D}, x, y)$ so that the maximum depth difference $\mathbf{d}(\widehat{idx})$ from the current location to local neighbors is calculated. If the magnitude of maximum depth difference $\widehat{\mathbf{d}}$ is bigger than the depth relative threshold value for depth discontinuities $\tau_{dd} \cdot \mathbf{D}(x, y)$, the center location is a candidate for one of the edges. When the maximum depth difference is positive, the point is regarded as an occluded edge point because the depth of the center location is deeper than its neighbors. Similarly, if the maximum depth difference is negative, the pose is regarded as an occluding edge point. This procedure is presented in line numbers from 10 to 22 in Algorithm 1.

Unfortunately, the occluded and occluding edges are not always well defined. Since the depth sensor of RGB-D cameras relies on reflected infra-red patterns for the depth calculation, depth values of some surfaces where their surface normals are nearly orthogonal to $\mathbf{z}$-axis of the sensor are not available. These unavailable depth values make edge detection more difficult, and must be handled appropriately. To handle this issue, our edge detection algorithm searches for corresponding points by skipping across these invalid points. The search direction is determined by $\texttt{SearchDirection}(\mathbf{N}, x, y)$ which averages the relative locations of the invalid points. If the current point and its corresponding point are determined, they are classified to either occluded or occluding edge points based on the criterion described above. When there is no corresponding point, the point is regarded as boundary edge point which is part of outer boundaries of the point cloud. Algorithm 1 line numbers from 23 to 33 describe the edge detection process around invalid points. The first row of Fig. 2 represents occluding, occluded, and boundary edges in green, red, and blue points respectively. Note that our occluding and occluded edges are similar to the obstacle and shadow borders in [31] respectively. The veil points in [31] are common in Lidar sensors, but for RGB-D sensors these points are typically invalid points instead of a point interpolated between the two surfaces.

B. *RGB and High Curvature Edges*

RGB-D cameras provide not only depth data but also aligned RGB data. Thus it is possible to detect 2D edges from RGB data and back-project these to the 3D point cloud. The Canny edge detector [7] is employed to find the edges due to its good localization and high recall. Algorithm 1 line numbers from 1 to 2 and from 34 to 35 show the RGB edge detection.

High curvature edges occur where surface normals change abruptly, such as a corner where two walls meet. These can be thought of as "ridge" or "valley" edges, or "concave" and "convex" edges. Such high curvature regions do not necessarily correspond to depth discontinuities. In computer graphics, several approaches [14], [15], [16] have been proposed to search for high curvature edges in 3D mesh models or point clouds. However, these require high quality measurements and are usually computationally expensive, making them unsuitable for our efficient use on noisy RGB-D data.

To develop a suitable and efficient method, we utilize the organized property of RGB-D point clouds so that high curvature edges are detected using a variant of Canny edge detection [7]. Recall that the Canny edge algorithm finds first-order image gradients $\mathbf{G_x}$ in $\mathbf{x}$ and $\mathbf{G_y}$ in $\mathbf{y}$ directions by applying a gradient operator to the input image, such as Sobel. While the original Canny edge algorithm applies a gradient operator to a gray scale image, our high curvature edge algorithm applies the operator to a surface normal image. Once normals $\mathbf{N}$ are estimated from the depth $\mathbf{D}$, we apply the Sobel operator to the $\mathbf{x}$ and $\mathbf{y}$ components of the normals, $\mathbf{N_x}$ (Fig. 1a) and $\mathbf{N_y}$ (Fig. 1b), to detect regions that have high responses in normal variations. After the first-order *normal* gradients, $\mathbf{G_x}$ (Fig. 1c) and $\mathbf{G_y}$ (Fig. 1d), are obtained, non-maximum suppression and hysteresis thresholding are used to find sharp and well-linked high curvature edges as shown in Fig. 1e. The high curvature detection is related to Algorithm 1 line numbers from 3 to 4 and from 36 to 37. In Fig. 2, the center and bottom rows show RGB (cyan) and high curvature (yellow) edges respectively.

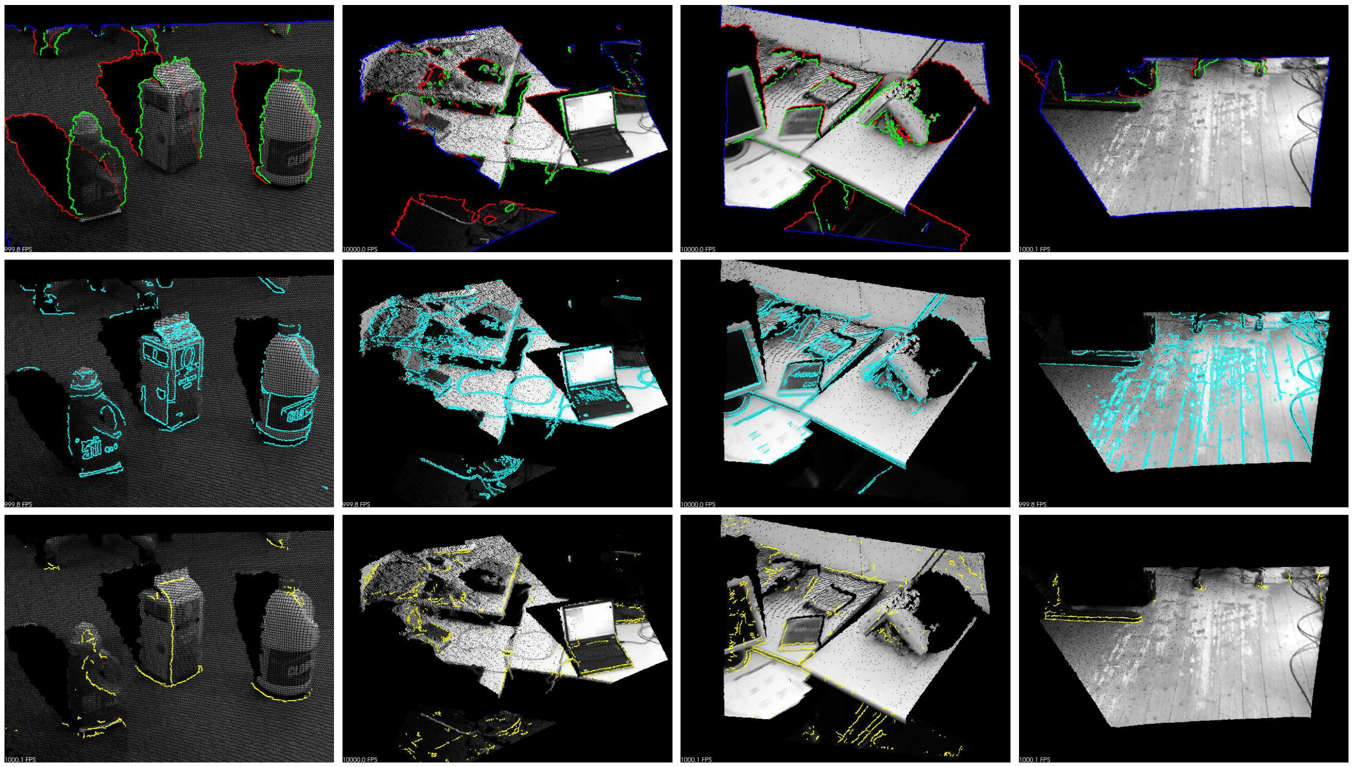

Fig. 2: **Detected edges in several example scenes.** The first row shows *occluding* (green), *occluded* (red), and *boundary* (blue) edges. And the middle row represents *RGB* edges in cyan points. The yellow points on the bottom row correspond to *high curvature* edges. (Best viewed in color)

IV. Edge-based Pair-wise Registration

Pair-wise registration—the process of aligning two point clouds of a partially overlapping scene—is an essential task for many 3D reconstruction and SLAM techniques. This is typically done by using the Iterative Closest Point (ICP) algorithm or one of its variants, using points or local planes as measurements [6], [19]. However, these algorithms can be quite computationally expensive for large point clouds, which usually necessitates downsampling the data. With the introduction of RGB-D sensors, 2D keypoints [32] and their back-projected 3D points have become popular features [4], [33] for pair-wise registration.

We propose using edge points as described above with the ICP algorithm for registration. Edge points detected in many indoor scenes are much sparser than the full point cloud, yet still capture much of the important structure required for 3D registration. Instead of uniformly downsampling the point cloud, the edge detection step can be viewed as an intelligent downsampling step that favors points that retain more of the important structure of the scene. Moreover, our occluding or high curvature edges are applicable in textureless scenes, while RGB edges can well represent textured scenes. To verify this hypothesis, we perform extensive experiments to compare the performance of ICP algorithms using our edge features with other alternatives in Section VI-B.

V. Edge-based SLAM

The above pair-wise registration algorithm can also be used for SLAM problems by using a pose-graph approach. Only the sensor trajectory will be optimized, by using only constraints between poses. We use the GTSAM library in this work, and in particular we use iSAM2 [30], which allows fast incremental updates to the SAM problem.

Each time a new point cloud is received, we proceed with edge detection as described in Section III and then perform pair-wise registration using the parameters chosen as described in Section VI-A and VI-B. A new pose $\mathbf{X}_n \in SE(3)$ is added to the pose graph, with a pose factor connecting $\mathbf{X}_{n-1}$ to $\mathbf{X}_n$ enforcing the relative transform between these poses as given by ICP. To improve robustness, we additionally add pose factors between $\mathbf{X}_{n-2}$ to $\mathbf{X}_n$ and $\mathbf{X}_{n-3}$ to $\mathbf{X}_n$.

In addition to adding pair-wise constraints between sequential poses, we also add loop-closure constraints between other poses. For each new pose $\mathbf{X}_n$, we examine poses $\mathbf{X}_0, \cdots, \mathbf{X}_{n-2}$, and compute the Euclidean distance between the sensor positions. Pairs of poses are only considered for a loop closure if the relative distance between them is less than a specified threshold, 0.2 m for this work, the angular difference is less than a specified threshold, and the pose was more than a specified number of poses in the past (we used 20 poses). If these conditions are met, we perform pair-wise registration between the two poses. If the ICP algorithm converges and the fitness score is less than a given threshold, we add a pose factor between these two poses. Along with the above factors to previous poses, this means each pose is connected to at most 4 other poses.

TABLE I: RMSE of translation, rotation, and average time in Freiburg 1 sequences

Sequence	SIFT keypoints	Points	Points down 0.01	Points down 0.02	Occluding edges	RGB edges	HC edges
FR1 360	14.1 ± 8.6 mm 1.27 ± 0.84 deg 410 ± 146 ms	18.6 ± 9.2 mm 0.95 ± 0.50 deg 14099 ± 6285 ms	21.7 ± 10.7 mm 0.95 ± 0.48 deg 3101 ± 2324 ms	22.9 ± 11.8 mm 1.01 ± 0.52 deg 913 ± 787 ms	23.0 ± 19.3 mm 1.03 ± 0.81 deg **132 ± 112**† ms	**11.2 ± 7.2**† mm **0.55 ± 0.33**† deg 321 ± 222 ms	20.5 ± 14.1 mm 0.77 ± 0.40 deg 606 ± 293 ms
FR1 desk	13.2 ± 8.6 mm 1.15 ± 0.78 deg 743 ± 223 ms	16.3 ± 8.0 mm 0.95 ± 0.50 deg 9742 ± 4646 ms	17.7 ± 8.9 mm 0.96 ± 0.51 deg 923 ± 824 ms	17.9 ± 8.6 mm 0.96 ± 0.51 deg 269 ± 265 ms	**7.3 ± 4.5** mm 0.82 ± 0.48 deg **100 ± 67** ms	8.6 ± 5.2 mm **0.70 ± 0.42** deg 427 ± 270 ms	10.5 ± 6.1 mm 0.89 ± 0.48 deg 230 ± 124 ms
FR1 desk2	13.2 ± 8.2 mm 1.16 ± 0.74 deg 620 ± 209 ms	18.8 ± 9.9 mm 1.12 ± 0.60 deg 12058 ± 6389 ms	20.7 ± 10.7 mm 1.12 ± 0.59 deg 1356 ± 1089 ms	20.7 ± 10.7 mm 1.12 ± 0.60 deg 380 ± 328 ms	**6.7 ± 3.6** mm 0.73 ± 0.37 deg **111 ± 71** ms	8.9 ± 5.4 mm **0.70 ± 0.43** deg 436 ± 268 ms	17.2 ± 13.8 mm 0.98 ± 0.55 deg 298 ± 161 ms
FR1 floor	18.1 ± 15.5 mm 0.76 ± 0.58 deg 595 ± 204 ms	16.9 ± 13.6 mm 0.65 ± 0.49 deg 6108 ± 3489 ms	16.9 ± 13.4 mm 0.65 ± 0.48 deg 474 ± 495 ms	16.9 ± 13.5 mm 0.72 ± 0.52 deg 125 ± 83 ms	112.9 ± 108.8 mm 8.16 ± 7.97 deg **46 ± 28** ms	**15.7 ± 15.2** mm **0.47 ± 0.39** deg 232 ± 129 ms	124.8 ± 122.2 mm 13.84 ± 13.66 deg 140 ± 32 ms
FR1 plant	14.8 ± 8.7 mm 1.04 ± 0.64 deg 668 ± 152 ms	14.3 ± 7.1 mm 0.78 ± 0.38 deg 10890 ± 5242 ms	21.6 ± 11.0 mm 0.87 ± 0.42 deg 2589 ± 1374 ms	21.8 ± 11.2 mm 0.86 ± 0.42 deg 851 ± 460 ms	**5.2 ± 3.0** mm 0.62 ± 0.32 deg **192 ± 132** ms	6.9 ± 3.6 mm **0.49 ± 0.27** deg 515 ± 283 ms	11.2 ± 6.3 mm 0.76 ± 0.38 deg 526 ± 234 ms
FR1 room	14.7 ± 11.4 mm 0.87 ± 0.59 deg 612 ± 201 ms	14.6 ± 6.9 mm 0.81 ± 0.42 deg 10410 ± 4482 ms	16.9 ± 8.3 mm 0.81 ± 0.42 deg 1703 ± 1353 ms	17.8 ± 9.0 mm 0.84 ± 0.44 deg 490 ± 460 ms	6.5 ± 3.9 mm 0.54 ± 0.27 deg **117 ± 82** ms	**6.2 ± 3.6** mm **0.48 ± 0.27** deg 368 ± 210 ms	10.8 ± 6.7 mm 0.68 ± 0.36 deg 340 ± 207 ms
FR1 rpy	14.2 ± 10.3 mm 1.05 ± 0.66 deg 642 ± 184 ms	12.6 ± 7.9 mm 1.04 ± 0.55 deg 13503 ± 6057 ms	17.5 ± 11.3 mm 1.12 ± 0.59 deg 2024 ± 1810 ms	17.5 ± 11.7 mm 1.17 ± 0.63 deg 677 ± 675 ms	**6.1 ± 3.4** mm 0.73 ± 0.37 deg **139 ± 90** ms	7.2 ± 4.3 mm **0.67 ± 0.39** deg 501 ± 281 ms	15.8 ± 11.4 mm 1.16 ± 0.62 deg 410 ± 251 ms
FR1 teddy	19.2 ± 13.0 mm 1.33 ± 0.88 deg 682 ± 211 ms	20.2 ± 11.4 mm 0.98 ± 0.58 deg 12864 ± 7904 ms	26.6 ± 14.9 mm 1.05 ± 0.61 deg 3647 ± 2315 ms	26.9 ± 15.1 mm 1.08 ± 0.63 deg 1194 ± 774 ms	21.9 ± 20.4 mm 0.99 ± 0.63 deg **187 ± 116** ms	36.5 ± 35.3 mm **0.92 ± 0.74** deg 667 ± 305 ms	**18.3 ± 12.6** mm 1.00 ± 0.54 deg 676 ± 322 ms
FR1 xyz	8.3 ± 4.9 mm 0.66 ± 0.34 deg 840 ± 181 ms	8.7 ± 5.1 mm 0.50 ± 0.24 deg 8358 ± 3440 ms	9.8 ± 5.9 mm 0.53 ± 0.27 deg 699 ± 462 ms	11.2 ± 6.2 mm 0.58 ± 0.30 deg 217 ± 157 ms	**4.3 ± 2.2** mm 0.51 ± 0.25 deg **96 ± 57** ms	4.7 ± 2.4 mm **0.41 ± 0.22** deg 352 ± 210 ms	6.3 ± 3.7 mm 0.55 ± 0.28 deg 195 ± 78 ms

† For each sequence, the first, second, and third lines represent translational, rotational, and time RMS errors, respectively. The best results are indicated in **bold** type.

VI. Experiments

In this section, we evaluate the performance of our edge detection, pair-wise registration using the edges, and edge-based pose-graph SLAM. For the evaluations, we use a publicly available standard dataset for RGB-D SLAM systems [5] which provides a number of RGB-D sequences with their corresponding ground truth trajectories, which come from an external motion tracking system. While the dataset was originally created for SLAM system evaluation, we use this dataset for the evaluations of edge detection and pair-wise registration as well. All of the experiments reported below are done in a standard laptop computer with an Intel Core i7 CPU and 8GB memory.

A. Edge Detection

TABLE II: Average computation time of edges in Freiburg 1 sequences

Sequence	Occluding edges†	RGB edges	HC edges
FR1 360	23.66 ± 1.03 ms	12.05 ± 0.99 ms	101.24 ± 5.28 ms
FR1 desk	24.06 ± 1.22 ms	13.04 ± 0.93 ms	96.24 ± 4.97 ms
FR1 desk2	24.71 ± 0.79 ms	12.76 ± 0.89 ms	99.55 ± 5.22 ms
FR1 floor	24.08 ± 1.87 ms	12.04 ± 0.94 ms	102.09 ± 5.03 ms
FR1 plant	24.61 ± 1.71 ms	13.04 ± 1.13 ms	99.25 ± 6.89 ms
FR1 room	23.86 ± 1.47 ms	13.07 ± 1.35 ms	100.91 ± 9.27 ms
FR1 rpy	23.89 ± 0.99 ms	12.87 ± 0.73 ms	98.33 ± 2.61 ms
FR1 teddy	25.20 ± 2.57 ms	13.14 ± 1.22 ms	103.70 ± 7.15 ms
FR1 xyz	24.45 ± 1.36 ms	13.27 ± 0.87 ms	98.85 ± 5.31 ms

† Please note that the computation time for occluding edges includes the time taken for both occluded and boundary edges as well, since our edge detection algorithm detects these three edges at the same time.

We evaluate the proposed edge detection both qualitatively and quantitatively. Our edge detection code is publicly available in the Point Cloud Library (PCL) [34]. The code in PCL was implemented as explained in this paper, but at the time the experiments were performed, PCL did not include an efficient canny implementation, so we adopted OpenCV Canny edge implementation for the following experiments. PCL now includes an efficient Canny implementation, which is used in the available code. As edge detection parameters, we empirically determined the following to work well for the RGB-D sensor used: $\tau_{rgb_{\perp}} = 40$ and $\tau_{rgb_{\top}} = 100$ for RGB edges, $\tau_{hc_{\perp}} = 0.6$ and $\tau_{hc_{\top}} = 1.2$ for high curvature edges, and $\tau_{dd} = 0.04$ and $\tau_{search} = 100$ for occluding edges.

Fig. 2 presents detected edges from four example scenes. The point clouds are displayed in grayscale for clarity, with detected edges highlighted in their own colors. In the top row, occluding and occluded edges clearly show the boundaries of the occluding objects and their corresponding shadow edges. These edges are obtained from depth discontinuities and well capture the geometric characteristics of the scenes. High curvature edges on the bottom row are also computed from geometric information but rely on high normal variation from the surfaces. These edges well represent the ridge or valley areas via our normal-based Canny edge variation.

Many applications, such as robotics, require near real-time performance. Table II shows the average computation time and standard deviation of edge detection in the nine Freiburg 1 sequences of the RGB-D SLAM benchmark dataset [5]. According to the table, detecting occluding edges per frame

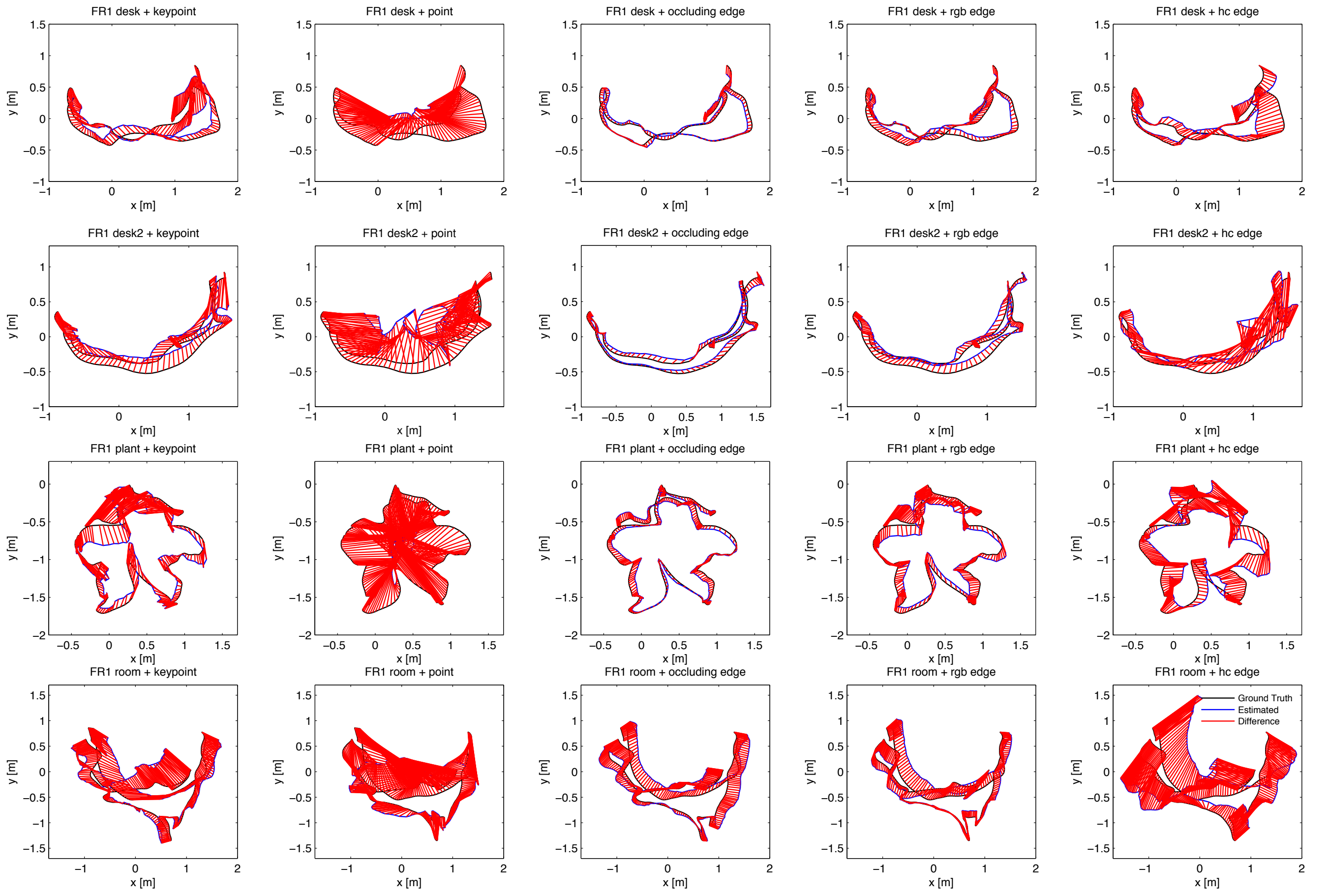

Fig. 3: **Plots of trajectories from pair-wise registrations.** Since resulting trajectories are estimated solely from pair-wise ICP, no loop closures were performed. In spite of that, our edge-based registrations report outstanding results, especially with occluding and RGB edges.

takes about 24 ms with no more than 3 ms standard deviation. The most efficient edge detection is RGB edges as it only takes about 13 ms per frame. But please note that our edge detection finds occluding, occluded, and boundary edges at the same time, and hence the reported time for occluding edges includes the computation time taken for both occluded and boundary edges as well. High curvature edge detection takes about 100 ms due to the additional computation cost of surface normals.

B. Pair-wise Registration

To examine the performance of our edges when used for the ICP algorithm, we compare our solution with SIFT keypoints and general point-based ICP algorithms. As evaluation metrics, [5] introduced the relative pose error (RPE):

$$\mathbf{E}_i := (\mathbf{Q}_i^{-1}\mathbf{Q}_{i+\Delta})^{-1}(\mathbf{P}_i^{-1}\mathbf{P}_{i+\Delta}) \qquad (1)$$

where $\mathbf{Q}_i \in SE(3)$ and $\mathbf{P}_i \in SE(3)$ are i-th ground truth and estimated poses respectively. When the length of camera poses is n, $m = n - \Delta$ relative pose errors are calculated over the sequence. While [5] used the root mean squared error (RMSE) by averaging over all possible time intervals Δ for the evaluation of SLAM systems, we fix $\Delta = 1$ since we are only interested in pair-wise pose errors here.

SIFT keypoints, full-resolution point clouds, and downsampled point clouds are compared with our edge points for use with ICP algorithms. For SIFT keypoint-based ICP, we used the implementation of [33] that employed SIFT keypoint and Generalized-ICP [19] and kept their best parameters for a fair comparison. They [33] originally reported their performance with SIFTGPU [35] for faster computation, but for a fair comparison we ran with SIFT (CPU only) because all other approaches do not rely on parallel power of GPU. For points, Generalized-ICP was also employed since it shows the state-of-the-art performance on general point clouds. As the size of original resolution point clouds from RGB-D camera is huge ($640 \times 480 = 307200$ in maximum), we downsampled the original point clouds via voxel grid filtering to evaluate how it affects its accuracy and computation time. Two different leaf sizes, 0.01 and 0.02 m, were tested for the voxel grid. For edge-based ICP, Generalized-ICP does not outperform the standard ICP [6] since locally smooth surface assumption is not valid for edge points, and thus the standard ICP is employed for edges. We used the same edge detection parameters used in the previous edge detection experiment in Section VI-A. Except the SIFT-based ICP, all ICPs are based on the implementation of PCL. To ensure fair comparison, we set the same max correspondence distance (0.1 m) and termination criteria (50 for the maximum number of iterations and 10^{-4} for the transformation epsilon) for all tests.

TABLE III: Performance of RGB-D SLAM [33] and our edge-based SLAM over some of Freiburg 1 sequences

Sequence	SIFT-based RGB-D SLAM [33]			Edge-based Pose-graph SLAM		
	Transl. RMSE	Rot. RMSE	Total Runtime	Transl. RMSE	Rot. RMSE	Total Runtime
FR1 desk	**0.049** m	**2.43** deg	199 s	0.153 m	7.47 deg	**65** s
FR1 desk2	**0.102** m	**3.81** deg	176 s	0.115 m	5.87 deg	**92** s
FR1 plant	0.142 m	6.34 deg	424 s	**0.078** m	**5.01** deg	**187** s
FR1 room	0.219 m	9.04 deg	423 s	**0.198** m	**6.55** deg	**172** s
FR1 rpy	**0.042** m	**2.50** deg	243 s	0.059 m	8.79 deg	**95** s
FR1 xyz	0.021 m	**0.90** deg	365 s	0.021 m	1.62 deg	**111** s

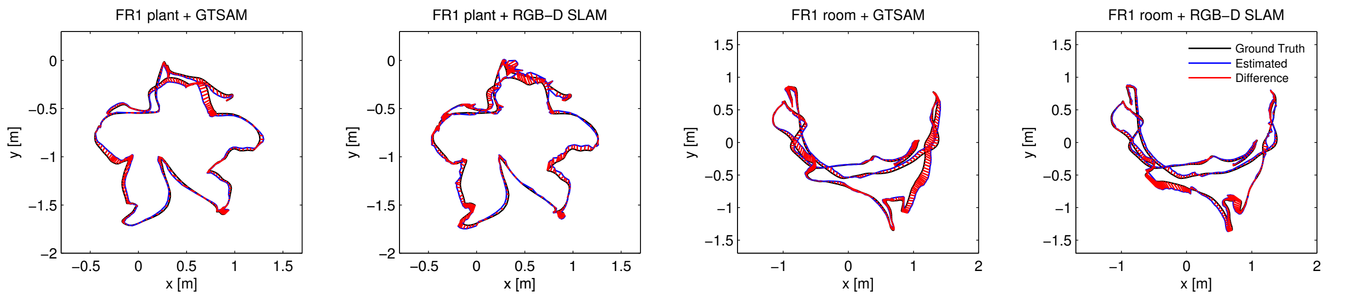

Fig. 4: **Plots of trajectories from two SLAM approaches.**

Table I represents RMSE of translation, rotation, and average run time of the seven approaches over the nine Freiburg 1 sequences. For each sequence, the first, second, and third lines show translational RMSE, rotational RMSE, and run time, respectively. The best results among the seven ICP approaches are shown in bold type. According to the reported RMSE, our edge-based ICP outperforms both SIFT keypoint-based and point-based ICP. ICP with occluding and RGB edges showed the best results. Interestingly, occluding edges report smaller translational errors, while RGB edges are slightly better for rotational errors. In terms of computational cost, the ICP with occluding edges was more efficient. We found high curvature edges to be slightly worse than the other edges, but this of course depends on the input sequences. Surprisingly, using all available points does not guarantee better performance, and actually reports nearly the worst performance over the nine sequences. This implies that some non-edge points are getting incorrectly matched, and are increasing the error, yielding less accurate results at high run times. One exception is "FR1 floor" sequence, in which using all points reports reasonable performance compared to other approaches. The sequence is mainly composed of wooden floor scenes in an office, so there are few occluding or high curvature edges. As one would expect, this sequence is challenging for occluding and high curvature edge-based ICP, but the texture from the wooden floor is well suited to RGB edge-based ICP, which yields the best result on this sequence. It is also worth noting that downsampling points greatly speeds up the runtime, but at the cost of a reduction in accuracy. It is even faster than the ICP with RGB edges, but the accuracy is the worst. And yet, it turns out that occluding edge-based ICP is the most efficient pair-wise registration method for the tested datasets.

It is also of interest to compare visual odometry style trajectories from the pair-wise registration to the ground truth trajectories. Although one large error in the middle of the sequence may seriously distort the shape of the trajectory, examining the accumulated errors over time can help us examine differences between these ICP methods. The estimated trajectories of "FR1 desk", "FR1 desk2", "FR1 plant", and "FR1 room" are plotted with their ground truth trajectories in Fig. 3. The plot data was generated via the benchmark tool of RGB-D SLAM dataset [5], in which the ground truth and estimated trajectories are aligned via [36] since their coordinate frames may differ. Based on the plots, the trajectories from points are the worst results and do not correlate with the ground truth trajectories. The trajectories from keypoints reasonably follow the ground truth but exhibits non trivial differences. As the best results in terms of RMSE, occluding and RGB edges result in clear trajectories which are close to the ground truth . These edges are quite powerful features for visual odometry style RGB-D point cloud registration and are also very promising if they are coupled with full SLAM approaches. The high curvature edge results show bigger translational errors, but the trajectories seem reasonable results which are comparable to those of keypoints and much better than those of points.

C. Edge-based SLAM

As described in section V, our edge-based registration can be applied to pose-graph SLAM. Some of the Freiburg 1 datasets used in Section VI-B were used for evaluation. For this experiment, we employed only occluding edges, since this type of edges is the most efficient yet effective as shown in Section VI-B. Table III presents RMSE and total computation times of both RGB-D SLAM [33] and our edge-based SLAM powered by GTSAM [23]. The results imply that our edge-based pose-graph SLAM is comparable to the state-of-the-art RGB-D SLAM in terms of accuracy. It even reported better results in "FR1 plant" and "FR1 room" sequences. For computational efficiency, the edge-based SLAM is quite efficient. Thanks to the faster pair-wise

ICP with the occluding edges, the total runtime of our SLAM system is about three times faster than the reported runtime of [33], though we could not directly compare the total runtime because both SLAM systems were run on different computers. Fig. 4 shows two trajectory plots of "FR1 plant" and "FR1 room" sequences where our SLAM reported better results. According to the plots, it is clear that trajectories of the edge-based SLAM are smoother as well as more accurate. These results indicate that the occluding edges are promising yet efficient measurement for SLAM problems.

VII. Conclusions

We presented an efficient edge detection approach for RGB-D point clouds which detects several types of edges, including depth discontinuities, high surface curvature, and photometric texture. Because some of these edge features are based on geometry rather than image features, they are applicable to textureless domains that are challenging for keypoints. These edges were applied to pair-wise registration, and it was found that edge-based ICP outperformed both SIFT keypoint-based and downsampled point cloud ICP. We further investigated the application of these edge features in a pose-graph SLAM problem, which showed comparable performance to a state-of-the-art SLAM system while providing efficient performance.

VIII. Acknowledgments

This work was mainly funded and initiated as a project of Google Summer of Code 2012 and the Point Cloud Library (PCL). The work has further been sponsored by the Boeing Corporation. The support is gratefully acknowledged.

References

[1] K. Lai, L. Bo, X. Ren, and D. Fox, "Sparse distance learning for object recognition combining RGB and depth information," in *Proc. IEEE Int'l Conf. Robotics Automation (ICRA)*, 2011, pp. 4007–4013.

[2] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. Rusu, and G. Bradski, "CAD-model recognition and 6DOF pose estimation using 3D cues," in *ICCV Workshops*, 2011, pp. 585–592.

[3] C. Choi and H. I. Christensen, "3D pose estimation of daily objects using an RGB-D camera," in *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots Systems (IROS)*, 2012, pp. 3342–3349.

[4] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments," in *Proc. Int'l Symposium on Experimental Robotics (ISER)*, 2010.

[5] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots Systems (IROS)*, Oct. 2012.

[6] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 239–256, 1992.

[7] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, Nov. 1986.

[8] H. Barrow, J. Tenenbaum, R. Bolles, and H. Wolf, "Parametric correspondence and chamfer matching: Two new techniques for image matching," in *Proc. Int'l Joint Conf. Artificial Intelligence (IJCAI)*, 1977, pp. 659–663.

[9] C. Olson and D. Huttenlocher, "Automatic target recognition by matching oriented edge pixels," *IEEE Trans. Image Proc.*, vol. 6, no. 1, pp. 103–113, 1997.

[10] C. Harris, *Tracking with Rigid Objects*. MIT Press, 1992.

[11] M. Isard and A. Blake, "Condensation–conditional density propagation for visual tracking," *Int'l J. Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[12] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid, "Groups of adjacent contour segments for object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 1, pp. 36–51, 2008.

[13] J. Shotton, A. Blake, and R. Cipolla, "Multiscale categorical object recognition using contour fragments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 7, pp. 1270–1281, 2008.

[14] Y. Ohtake, A. Belyaev, and H. P. Seidel, "Ridge-valley lines on meshes via implicit surface fitting," in *ACM Trans. Graphics*, vol. 23, 2004, pp. 609–612.

[15] S. Gumhold, X. Wang, and R. MacLeod, "Feature extraction from point clouds," in *Proc. 10th Int. Meshing Roundtable*, 2001, pp. 293–305.

[16] M. Pauly, R. Keiser, and M. Gross, "Multi-scale feature extraction on point-sampled surfaces," in *Computer Graphics Forum*, vol. 22, 2003, pp. 281–289.

[17] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2D range scans," *Journal of intelligent & robotic systems*, vol. 18, no. 3, pp. 249–275, 1997.

[18] J. Minguez, F. Lamiraux, and L. Montesano, "Metric-based scan matching algorithms for mobile robot displacement estimation," in *Proc. IEEE Int'l Conf. Robotics Automation (ICRA)*, vol. 4, 2005, pp. 3557–3563.

[19] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proc. Robotics: Science and Systems (RSS)*, 2009.

[20] K. Pulli, "Multiview registration for large data sets," in *Proc. Int'l Conf. 3-D Digital Imaging and Modeling (3DIM)*, 1999, pp. 160–168.

[21] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proc. IEEE Int'l Conf. Robotics Automation (ICRA)*, 2011, pp. 3607–3613.

[22] S. Agarwal and K. Mierle, *Ceres Solver: Tutorial & Reference*, Google Inc.

[23] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *Int'l J. Robotics Research*, vol. 25, no. 12, pp. 1181–1204, 2006.

[24] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6D SLAM—3D mapping outdoor environments," *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 699–722, 2007.

[25] K. Pathak, A. Birk, N. Vaskevicius, and J. Poppinga, "Fast registration based on noisy planes with unknown correspondences for 3-D mapping," *IEEE Trans. Robotics*, vol. 26, no. 3, pp. 424–441, 2010.

[26] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, "Real-time 3D visual SLAM with a hand-held camera," in *Proceedings of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*, Vasteras, Sweden, April 2011.

[27] G. Grisetti, R. Kummerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2D and 3D mapping," in *Proc. IEEE Int'l Conf. Robotics Automation (ICRA)*, 2010, pp. 273–278.

[28] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. Int'l Symposium on Mixed and Augmented Reality (ISMAR)*, 2011, pp. 127–136.

[29] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *ACM Transactions on Graphics (SIGGRAPH)*, 1996.

[30] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Int'l J. Robotics Research*, 2012.

[31] B. Steder, R. Rusu, K. Konolige, and W. Burgard, "Point feature extraction on 3D range scans taking into account object boundaries," in *Proc. IEEE Int'l Conf. Robotics Automation (ICRA)*, 2011, pp. 2601–2608.

[32] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int'l J. Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[33] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *Proc. IEEE Int'l Conf. Robotics Automation (ICRA)*, May 2012, pp. 1691–1696.

[34] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Proc. IEEE Int'l Conf. Robotics Automation (ICRA)*, 2011, pp. 1–4.

[35] C. Wu, *SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)*, 2007.

[36] B. K. Horn *et al.*, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.