Robotica (2017) volume 35, pp. 809–831. © Cambridge University Press 2015
doi:10.1017/S0263574715000831

3D SLAM in texture-less environments using rank order statistics

Khalid Yousif*, Alireza Bab-Hadiashar and Reza Hoseinnezhad

School of Aerospace, Mechanical and Manufacturing Engineering, RMIT University, Melbourne, VIC 3001, Australia

(Accepted September 21, 2015. First published online: October 21, 2015)

SUMMARY
We present a real time 3D SLAM system for texture-less scenes using only depth information provided by a low cost RGB-D sensor. The proposed method is based on a novel informative sampling scheme that extracts points carrying the most useful 3D information for registration. The aim of the proposed sampling technique is to informatively sample a point cloud into a subset of points based on their 3D information. The flatness of a point is measured by applying a rank order statistics based robust segmentation method to surface normals in its local vicinity. The extracted keypoints from sequential frames are then matched and a rank order statistics based robust estimator is employed to refine the matches and estimate a rigid-body transformation between the frames. Experimental evaluations show that the proposed keypoint extraction method is highly repeatable and outperforms the state of the art methods in terms of accuracy and repeatability. We show that the performance of the registration algorithm is also comparable to other well-known methods in texture-less environments.

KEYWORDS: SLAM; Localization; Mapping; RGB-D; Texture-less.

1. Introduction

The availability of affordable RGB-D sensors has generated intense interest in creating dense 3D models of the environment.[1–3] These sensors are able to record RGB images as well as pixel depth information. A popular family of this type of sensors is developed by Microsoft and its latest version, called Kinect 2.0, has a resolution of 1920×1080 pixels and can measure distance of around 5 m. The affordability of these sensors makes them a very attractive 3D sensor for different computer vision and robotics applications.

In particular, RGB-D sensors are very useful for autonomous navigation applications. Simultaneous Localization and Mapping (SLAM)[4] is the task of estimating the pose of a moving sensor and mapping its unknown environment, simultaneously. The SLAM problem has been intensely studied over the past couple of decades and many solutions have been developed for specific applications,[4] including generation of dense 3D maps. These maps are rich in information and are useful for object recognition and manipulation, collision avoidance and path planning. A 3D representation of the environment is also useful in augmented reality applications that facilitate interactions between the real and virtual environments.[1]

The use of RGB-D sensors for 3D SLAM poses a number of challenges. In particular, mobile robots are commonly required to navigate in texture-less areas such as offices, warehouses and residential buildings. The registration of frames in such texture-less environments is difficult as there are not readily available visual cues to align those frames. This issue is illustrated in Fig. 1 which shows a typical corridor in a university building. Furthermore, the size of data generated by RGB-D sensors makes it difficult to capture and process their data in real-time.

* Corresponding author. E-mail: s3362555@student.rmit.edu.au; abh@rmit.edu.au; rezah@rmit.edu.au

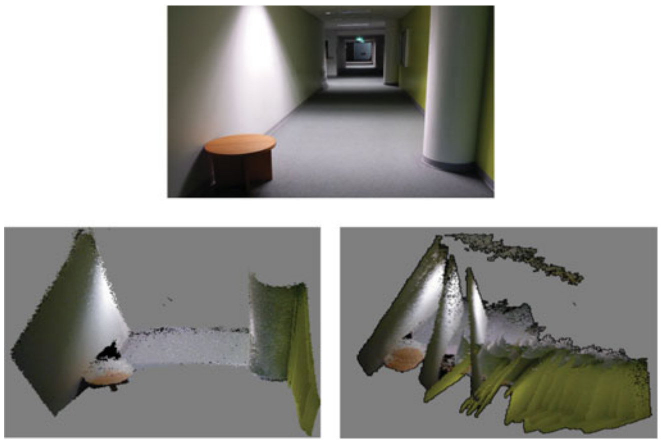


Fig. 1. (Top) A typical corridor in an office building with limited texture information. (Bottom) Alignment of multiple image using either (Left) only 3D features (Right) or only visual features.[19]

The paper outlines the problem of 3D SLAM in *texture-less* environments. Our aim is to develop a fast and accurate method that does not rely on the information provided by the RGB images. This enables us to study the limits of using structures for solving 3D SLAM. To fulfill this goal, we developed a sampling strategy to extract salient geometric 3D keypoints from sequential frames. We then assign a descriptor to each feature, match them using their descriptors and refine the matches and calculate a rigid-body transformation between the two frames using a robust estimator.[5] The relative transformations are then concatenated up to the current time resulting in a global pose. We finally employ a loop closure and pose graph optimization technique[6] in order to reduce the drift and obtain a globally consistent trajectory. Finally, a map is constructed by projecting and transforming the points according to the optimized trajectory. An example of a map that is obtained using the proposed method is shown in Fig. 2. Extraction and matching of 3D keypoints for a typical point cloud captured by an RGB-D sensor is very time consuming. The emphasis here is on the selection of a small subset of points that can be used to register two point clouds with similar registration accuracy to using the entire sets.

The main contribution is the development of an informative sampling based 3D feature extraction technique. The method is able to exploit the geometric information of the points and their neighbors to identify points that carry the most useful information. We call the points resulting from our informative sampling scheme: *Ranked Order Statistics* (ROS) keypoints. We show that the proposed keypoint extraction method is highly *repeatable* and can obtain a subset of points of the original point cloud that results in a very accurate registration compared to using a point cloud containing many more points ($\approx$15 times more points). The main advantage of using this sampling technique is that it would reduce computational time significantly. In fact, we will show that our method outperforms several state of the art registration methods both in accuracy and computational efficiency. This paper is an extended version of work published in ref. [20]. We extend our previous work as follows:

- Evaluating the repeatability of the proposed 3D keypoint extraction method in comparison to other well-known methods.
- Adding loop closure and global optimization steps for achieving global consistency. The proposed scheme stores only ROS keypoints at certain increments of motion (translation or rotation) and uses those to speed up loop closure detection.
- Performing additional experiments for the evaluation of the accuracy and efficiency of registration methods using a new rotation and translation sequence.
- Evaluating the effect of point cloud pre-processing.
- Evaluating 3D descriptors for point cloud based SLAM.
- Evaluating the key-frame selection criteria on the pose graph optimization
- Evaluating the performance of the proposed method using publicly available benchmark datasets.[8]
- Comparing the proposed method with the state of the art RGBD-SLAM v2[9] using both our own sequences and the public datasets.

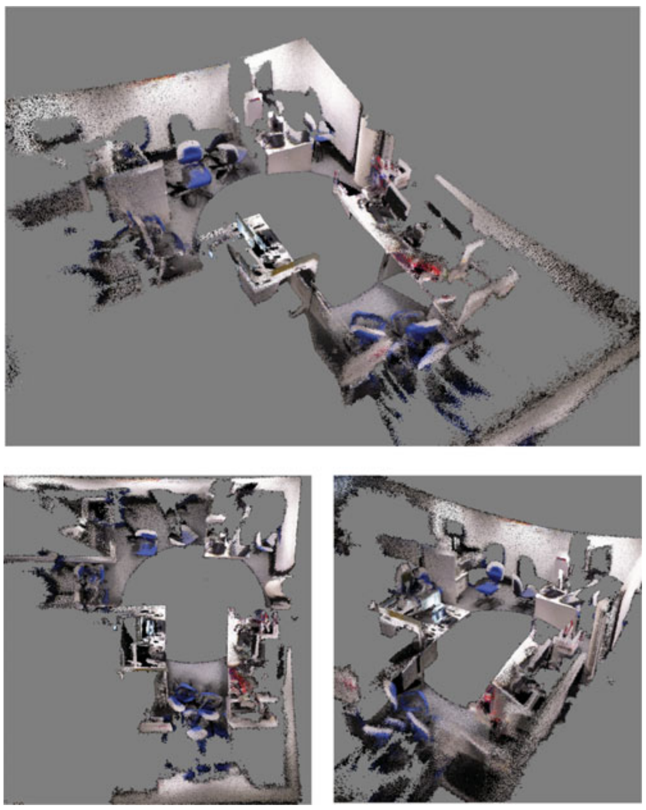


Fig. 2. Three views of a the constructed 3D map of an open office area using the proposed 3D registration and mapping algorithm.

The rest of the paper is organized as follows. In Section 2 the related work in this area are reviewed. We will present our informative sampling based feature extraction method in Section 3 and our 3D registration and mapping method in Section 4. Results are presented in Section 5 followed by a conclusion in Section 6.

2. Literature Review

2.1. RGB-D SLAM

The use of Microsoft Kinect for 3D SLAM was popularized by Henry *et al.*[2] They used a Microsoft Kinect to capture RGB-D data in sequential frames and estimated the camera pose and obtained a 3D reconstructed environment. In their method, Features from Accelerated Segment Test (FAST) keypoints[10] were extracted and matched between sequential RGB images and Random Sample Consensus (RANSAC) method was used to refine those matches and estimate an initial transformation. This transformation was refined using a generalized-version of the Iterative Closest Point (ICP)[11] algorithm. Global consistency was achieved by applying a Sparse Bundle Adjustment method and loop closure was detected by matching the current frame to previously collected key-frames. Endres *et al.*[3,9] proposed a similar method called "RGB-D-SLAM" in which Scale Invariant Feature Transform (SIFT),[12] Speeded Up Robust Features (SURF)[13] or Oriented FAST and Rotated BRIEF (ORB)[14] keypoints were used in place of FAST keypoints and the pose-graph optimization was used instead of Bundle Adjustment for global optimization. Du *et al.*[15] extended Henry *et al.*'s method by incorporating on-line user interaction and feedback during the map building step. User input enabled the system to recover from registration failures which may be caused by a fast moving camera. Audras *et al.*[16] outlined the problem of false feature matching and instead, used an appearance based optical flow method for estimating the camera pose and building a 3D map.

To avoid using the RGB images, Newcombe *et al.*[1] proposed a GPU based RGB-D mapping algorithm called "KinectFusion". In their method, an ICP variant for registration in which the current measurement is matched with the full growing surface model was proposed as opposed to matching sequential frames. They also implemented a segmentation step which divided the scene into foreground and background and was able to detect moving humans that would be present in the scene. This allowed user interaction during the map building procedure without deteriorating the accuracy of the estimated transformations and the map. The major downside to their approach is that it is computationally expensive because of the requirement to store and process the full dense volumetric representation of the constructed model in GPU memory and as such, is not scalable for large environments. Whelan *et al.*[17] proposed an extension to KinectFusion which outlined those problems and allowed the region of the environment that is mapped by the KinectFusion algorithm to vary dynamically, allowing their approach to map larger scenes. Our approach is similar as we also do not rely on the availability of RGB keypoints for registration. However, since ICP only works when there is a good initial alignment, we advocate the use of geometric 3D keypoints for registration. In addition, both of the aforementioned approaches are GPU based methods, whereas our approach uses CPU only for processing.

Bachrach *et al.*[18] proposed a visual odometry and RGB-D mapping system for unmanned air vehicles (UAVs). Their method relied on pre-processing and extracting FAST keypoints from sequential RGB images at different pyramid levels, followed by a rotation estimation step that improved the efficiency of feature matching by limiting the size of the search window. An 80-byte descriptor was assigned to each feature, consisting of the brightness values of the 9×9 pixel patch around the feature and omitting the bottom right pixel. Feature matching was then performed by finding the mutual lowest sum of squared difference (SSD) score between the descriptor vectors. In order to refine the matches which were then used to estimate the transformation between frames, a greedy algorithm was employed. The authors suggested matching the current frame to a previously stored key-frame instead of matching consecutive frames for reducing the drift in the motion estimates.

Kerl *et al.*[20] recently proposed a 3D SLAM method that uses both photometric and geometric information for registration. In their implementation, they use all the points for registration and optimize both intensity and depth errors. Another recent method proposed by Keller *et al.*[21] allows for 3D reconstruction of dynamic environments by automatically detecting dynamic changes in the scene. Hu *et al.*[22] also proposed a 3D SLAM method that used a heuristic switching algorithm to choose between an RGB-D mapping approach and a 8-point RANSAC monocular SLAM, based on the availability of depth information. A 3D Iterative Sparse Local Submap Joining Filter (I-SLSJF) was then used to merge the two generated maps. In our previous work,[19] we outlined the problem of 3D registration in dark environments and proposed a method that switches between RGB and Infrared (IR) images for visual feature extraction and matching based on the brightness of the RGB image. A highly robust estimator[5] was then employed to estimate the transformation between frames. The estimate was then refined using an ICP algorithm. This method fails in cases where both the RGB and the IR images contain insufficient visual information, since ICP alone fails to provide an accurate estimate without being provided with a good initial guess. The method proposed in this paper is designed to overcome this issue solely based on using depth information.

2.2. Keypoint detection in 3D

Three dimensional keypoints are typically an extension of their 2D counterparts. For instance, the well-known Harris corner detector[23] has been implemented in 3D and is available in the point cloud library (PCL).[24] In the 3D version, surface normals were used in place of image gradients .[25] Similarly SIFT,[12] which is a blob detector to extract image patterns that differ from their immediate neighborhood in terms of intensity, color and texture, has also been extended to in 3D and is available in PCL. In the 3D implementation, instead of comparing a pixel with its neighboring pixels in an image, a 3D point is compared to its neighbors by using a kd-tree search based on the Euclidean distance. Gelfand *et al.*[26] proposed a sampling strategy that groups the point cloud data into stable points (constraining points) and non-stable points (that have no effect on the transformation estimation). This is achieved by analyzing the contribution of the force (translational) and torque (rotational) components by each pair of points using a point-to-plane error metric. The main shortcoming of this method is that it is

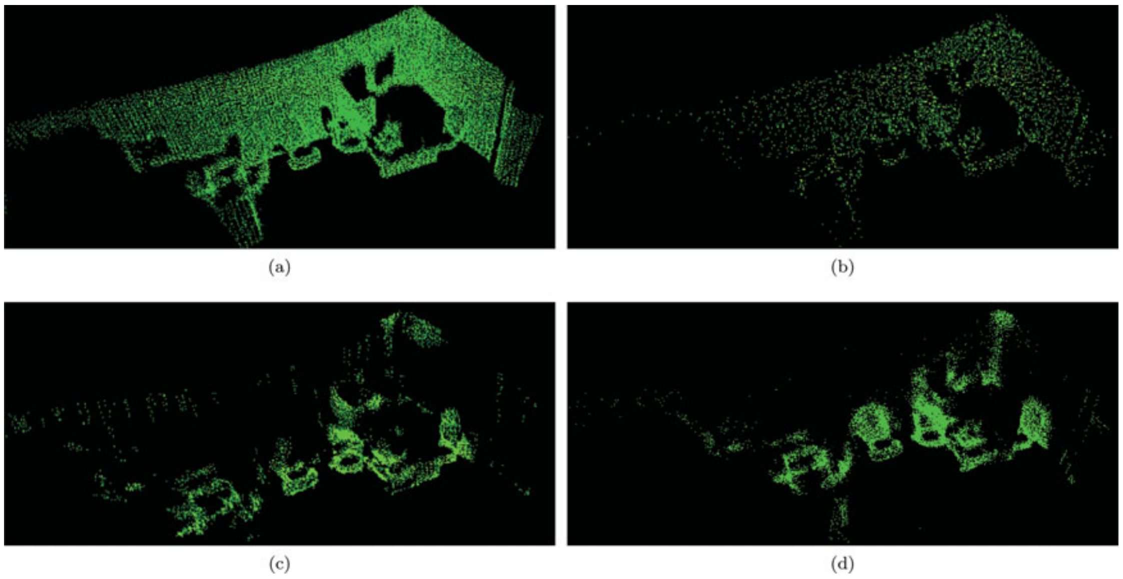


Fig. 3. A comparison between different sampling techniques. The point clouds after: (a) uniform sampling (b) ISS (c) covariance keypoints (d) the proposed keypoint extraction method (ROS).

optimized for a specific metric and it becomes suboptimal when other metrics (such as the squared Euclidean distance error metric employed here) are used. An example of this method is illustrated in Fig. 3 (c) which shows some points with significant local curvature being omitted from the point cloud (such as the top part of a chair). Zhong[27] proposed a 3D feature detector and descriptor called Intrinsic Shape Signatures (ISS). This method relies on exploiting the information provided by the covariance matrix that is constructed using surface normals estimates. The method extracts keypoints based on two measures: points that correspond to the smallest eigenvalues of the covariance matrix representing those with large variations, and the ratio between successive eigenvalues in order to discard redundant points that have a similar spread.

It's important to know that the above 3D feature extraction methods are mainly developed for general 3D modeling applications and those are not particularly optimized for SLAM purposes. For instance, our experiments showed that Zhong's method[27] finds many feature points on commonly encountered planar surfaces such as office walls and doors (see Fig. 3(b)). Some of those points are ill-conditioned for registration purposes and can deteriorate the estimation outcome.

This paper addresses the problem of aligning RGB-D images in an environment that may not contain sufficient texture information for 3D registration. Our proposed method extracts unique 3D keypoints from sequential frames using a novel ROS based informative sampling segmentation method. The extracted 3D keypoints are then matched using SHOT descriptors[28] and finally, the modified selective statistical estimator (MSSE)[5] is employed to segment the good matches and estimate a 6DOF transformation between the two frames. The estimated transformations are concatenated up to the current time in order to obtain a global transformation of the camera. Finally, we construct pose graph consisting of nodes (frames) and edges (constraints) and optimize the camera trajectory using a global optimization method.[6] The map in obtained by projecting and aligning the points into a global reference frame using the optimized trajectory. Figure 4 shows a system overview of the proposed SLAM method. The above steps are outlined in the following sections.

3. Extracting Geometric Keypoints Using Ranked Order Statistics

The aim of our proposed keypoint extraction method is to informatively down-sample a point cloud into a subset of points that geometrically differ from their immediate neighborhood. This is somewhat similar to the idea behind geometrically stable sampling for ICP.[26] However, we present an efficient method that finds the sample with the most information directly using the statistical analysis of their flatness. The main idea is to segment the points into two main groups: points that are locally flat

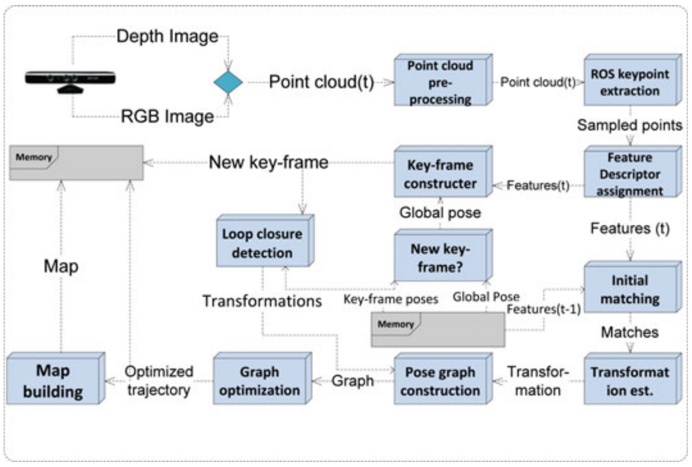


Fig. 4. A system overview of the proposed SLAM method.

and points that have significant local curvature. The details of the implementation of this method are presented in Algorithm 1. The input to this method is a point cloud with a normal vector calculated at every point. We then calculate the average of the squared angles between each point's normal and the normals of its N nearest neighbors which are found using a kd-tree search. The angles are calculated using the following equations:

$$\phi = \arccos(p_i.q_f) \tag{1}$$

$$\theta_i = \frac{\sum_{j=1}^{N} \phi_j^2}{N}; \quad i = 1 \ldots n, \tag{2}$$

where ϕ is the positive angle between two normal vectors p_i (a query point) and q_f (a neighboring point), n is the number of points in the point cloud and θ_i is the average angle between normals (of the query points and all its neighbors). The average squared angle θ_i which is calculated for every point in the point cloud, is considered as an error measure that specifies the similarity of a point's curvature to its neighbors. Therefore, for points associated with a θ_i value close to zero, the orientations of

Algorithm 1 Step-by-Step Algorithm of Proposed Feature Extraction Method

```
1:  Input: Point cloud with Normals $P_n$
2:  Output: Sampled Point cloud $P_d$
3:  Initialize and clear vector $\alpha$;
4:  for $i = 0$ to $length(P_n)$ do
5:      Find $N$ nearest neighbors to query point $i$;
6:      for $j = 1$ to $N$ do
7:          Calculate angle between normals ($\phi_j$) of query point and its neighbor $j$;
8:          Add $\phi_j^2$ to the vector $\alpha$;
9:      end for
10:     Calculate the average of the angles in vector $\alpha$ ($\theta$);
11:     Add $\theta$ to a vector;
12: end for
13: Sort $\theta$ values in an ascending order and store in ($r_\theta$);
14: Apply the MSSE (4) to find transition point ($k'$) that separates the two groups;
15: Store indices associated with points (from Group 2) that have sorted indices $> k'$ and $< k' + 400$
    in indices vector $K_{Indices}$;
16: Store points from $P_n$ associated with $K_{Indices}$ into $P_d$;
17: return $P_d$;
```

normals of the point and its neighbors are similar (such as points lying on a flat plane) regardless of the normal vector's orientation at those points (i.e. regardless of which plane they lay on). On the other hand, the points associated with higher residual values differ from their neighborhood and carry more useful information for registration purposes.

To further clarify this point, let us assume that there are two planes perpendicular to each other. Those points with the smallest residuals correspond to points lying on either one of the two planes. However, we are mainly looking for the points that lay close to the intersection between the two planes. In this scheme, those correspond to points with higher residual values.

In the next step, we sort all the θ_i values (those are all positive) in ascending order and store those in the vector r_θ. The final step is to find the transition point that separates the two aforementioned groups. A ROS based segmentation technique (MSSE)[5] is employed for this segmentation. This method is outlined as follows.

3.1. Robust segmentation

Although there are many techniques to conduct robust segmentation, we chose MSSE because its implementation is straight forward and runs efficiently.[29] In MSSE, the segmentation is conducted iteratively by first calculating the standard deviation of sorted data using the first k sorted values (initial value of k corresponds to the assumed minimum percentage of points included in the first segment) as follows:

$$\sigma_k^2 = \frac{\sum_{i=1}^{k} r_{\theta_i}^2}{n-p}, \tag{3}$$

where $r_{\theta i}$ is ith data in the sorted vector r_θ and p is the dimension of the model. The transition point k' is found by iteratively incrementing k until the following condition is met:

$$r_{\theta_{k'+1}} > T\sigma_{k'}, \tag{4}$$

where T is a constant factor which is set to 2.5 and includes around 99% population of a normal distribution.

3.2. Keypoint selection

The points associated with values in r_θ vector beyond the transition point (GROUP 2) are deemed to have statistically significant local curvature. Figure 5 shows the classification of different points

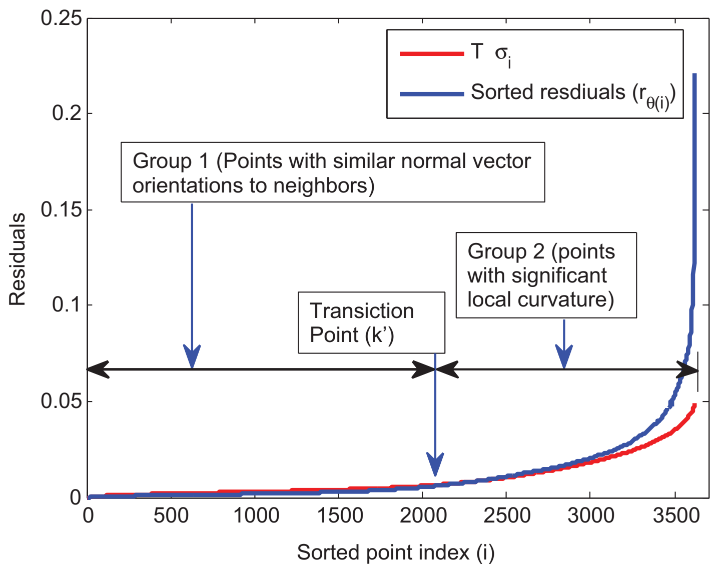


Fig. 5. Classification of points based on the calculated residuals using the MSSE. The extracted keypoints are those chosen from Group 2 (points with significant local curvature).

based on the calculated θ values using the MSSE constraint (4) starting at a k associated with 20% of all data. Since these are sorted values, group of points with the highest values (e.g. last 5% of sorted values) are likely to include points with gross measurement errors (outliers such as outside range values returned by the sensor). The points associated with the remaining members of GROUP 2 are then chosen as keypoints (called ROS keypoints).

Our extensive experiments showed that although the exact number of chosen points does not affect the registration accuracy, having around 400 keypoints provides a good balance between registration accuracy and speed. Figure 3 shows a comparison between the proposed method and other feature extraction techniques. In our experiments, ROS keypoints are always chosen as ones associated with the first 400 sorted values of GROUP 2.

4. 3D Registration and Mapping

In the following sections, we will discuss the steps of our proposed 3D registration and mapping method which are outlined in Algorithm 2.

Algorithm 2 Step-by-Step Algorithm of Proposed 3D Registration and Mapping Method

```
 1: Input: Point cloud (previous frame) $P_t$, point cloud (current frame) $P_s$)
 2: Output: Transformation between frames $T$ , Global pose of camera $G_n$, Map $M$
 3: Filter point clouds;
 4: Uniformly down-sample $P_t$ and $P_s$ to approximately $\approx 4000$;
 5: Calculate normal vectors at each point in $P_t$ and $P_s$;
 6: Extract ROS keypoints from $P_t$ and $P_s$ as described in Algorithm 1;
 7: Assign SHOT descriptors to each feature point in $P_t$ and $P_s$;
 8: Initially match the extracted keypoints between $P_t$ and $P_s$ using their descriptors;
 9: Obtain the inliers (good matches) using MSSE and estimate the 6DOF transformation $T$ between
    the two consecutive frames;
10: Concatenate the estimated transformations to obtain a global pose of the camera $G_n$.
11: Transform and map ($M$) the points with respect to a global reference frame;
12: return $T$, $G_n$ and $M$;
```

4.1. *Pre-processing steps*

To improve the efficiency and accuracy of the registration, we apply a pass-through filter to exclude points that are more than 4 m away from the sensor (due to the decrease in depth precision the further the points are from the camera). We then uniformly sample the point cloud (containing around 300, 000 points) using a voxel grid in which the 3D space is divided into many small 3D boxes (voxels). All points lying inside a box (with pre-defined dimensions) are represented by their centroid. As a result, the minimum distances between points in the point cloud are constrained and the total number of points are reduced. The point cloud spatial resolution decreases the further the scene is from the RGB-D sensor. As such, using a pre-defined voxel grid size results in point clouds with varying number of points in each frame, depending on the depth information in the scene (e.g. in our experiments, point clouds consisted of around 2000 to 7000 points). To insure that the sampled points are distributed fairly uniformly in different frames with varying depth, we assign a variable voxel leaf size to every frame based on computing the point cloud resolution which is calculated by finding the average distance between each point in the point cloud and its nearest neighbors. The voxel leaf size can be calculated using the following equation:

$$\varrho = C \times \varphi, \tag{5}$$

where ϱ is the voxel leaf size, φ is the calculated point cloud resolution and C is a pre-defined constant. This constant is defined based on approximately how many points the user wishes to obtain. In our experiments, the above procedure reduces the number of points to around 4000 points (using a constant factor of 11).

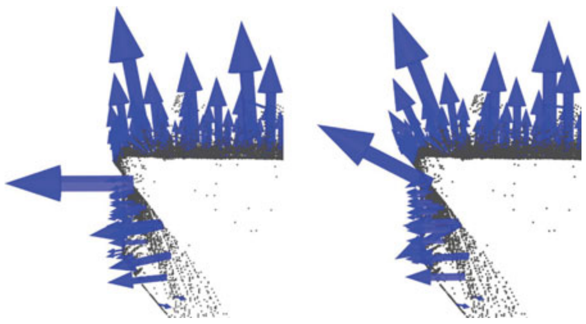


Fig. 6. Example of estimated surface normals for a subset of points using a small search radius (left) large search radius (right) (Courtesy of and permission granted by Radu B. Rasu[30]).

4.2. Normal vector estimation

We estimate the normal to a surface at a given point by fitting a plane to the point and its neighbors in a variable search area that is calculated in a similar way to the method explained in Section 4.1. The search area is intentionally large to include points from adjacent surfaces, therefore affecting the surface normals near edges and corners. By doing so, we differentiate between the angles of normals lying entirely on a plane and those near corners and edges. This concept is illustrated in Fig. 6 in which the effect of using a large search area is demonstrated in the right image.

4.3. 3D keypoint extraction and matching

The process of constructing a descriptor vector, even for around 4, 000 points, and matching those with their corresponding points from the previous frame is still computationally expensive. To improve the computation efficiency, we need to find a way to reduce the number of points without losing the information required for registration. To achieve this, we developed an informative sampling scheme based on using ROS (described in Section 3). After applying this method, the resulting point cloud would contain around 400 points. A feature descriptor is then computed for each extracted keypoint. Our experiments showed that SHOT descriptors[28] were the most accurate and efficient descriptors in comparison to other state of the art descriptors (see Section 5.3). Matching was performed, using a mutual consistency check, by finding the nearest neighbors in the descriptor vector space from the source point cloud (at time t) to the target point cloud (at time $t-1$) and vice versa. Only pairs of corresponding points that were mutually matched to each other were considered as the initial correspondences.

4.4. Inlier detection and initial transformation estimation

Matching 3D keypoints between consecutive images using their descriptors usually results in a number of false matches (outliers). To remove the effect of these false matches, we refine the matches using a high breakdown point robust estimator (i.e. MSSE[5]). The MSSE is an extension of the robust least *kth* order statistical estimator and can tolerate much higher ratio of outliers compared to RANSAC (we previously showed that MSSE outperforms RANSAC in terms of registration accuracy[19]). MSSE is comprised of two main steps, first the MSSE fits a given model to a set of data that contains outliers. In the second step, the scale of noise from this data is estimated. In contrast to RANSAC, MSSE is flexible and does not rely on the strong fixed error threshold assumption (which forms the bases of the RANSAC algorithm).[19] Instead, MSSE assumes that the minimum number of inliers that are required to define a structure is known a priori. After setting the minimum number of inliers (k), the implementation of MSSE is very straightforward and is as follows. One thousand sets of 3-tuples of corresponding initial matches are randomly chosen and for each set, the transformation is estimated using the following equation:

$$\mathbf{T}^* = \underset{\boldsymbol{T}}{\operatorname{argmin}} \left(\sum_{i=1}^{A_d} |\mathbf{T}(p_s^i) - p_t^i|^2 \right) \tag{6}$$

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}, \tag{7}$$

where $\mathbf{T}^*$ is the estimated 6DOF transformation which consists off a translational component $\mathbf{t} \in \Re^{3\times 1}$ and a rotational component $\mathbf{R} \in SO(3)$, p_s and p_t are the 3D coordinates of the matched feature points of the source and target frames respectively and A_d is the number of correspondences (here $A_d = 3$). The estimated transformation is applied to all the keypoints in the first frame and the distances between the transformed keypoints and the corresponding points in the target frame are calculated (called residuals - r) and their squared values are sorted in ascending order. Having set the k, then the transformation associated with the least kth order residual (from the 1000 hypotheses) is chosen as the best initial transformation. To segment these residual values, we use the robust segmentation part of MSSE as outlined in Section 3.1. Using the residuals of the chosen transformation, the inlier group members are chosen by applying the condition (4) iteratively starting at $i = k$ and incrementing this value at each iteration until the condition is met. Finally, the transformation is re-calculated using all the obtained inliers.

4.5. *Global pose estimation*

The transformation that was calculated in the previous section describes the motion between two RGB-D frames. In order to obtain a global pose of the camera with respect to a fixed reference frame (the initial frame), we concatenate all the transformations up to the current time using:

$$\mathbf{G_n} = \mathbf{G_{n-1}}\mathbf{T_{n,n-1}}, \tag{8}$$

where $\mathbf{G_{n-1}}$ is the previous global transformation with respect to the pose of the initial reference frame $\mathbf{G_0}$ at $n = 1$ and $\mathbf{T_{n,n-1}}$ is the 6DOF transformation between the current frame and the previous frame.

4.6. *Global optimization*

The global pose estimation procedure described above concatenates the relative transformations between frames and provides visual odometry information. However, these frame to frame estimates contain a small error due to noise and inaccuracies in the camera model. Accumulation of those small errors increase the motion estimation error over time. In order to reduce the accumulated drift and obtain a globally consistent trajectory and map, we employ an open-source pose graph optimization framework $\mathbf{g^2o}$.[6] The graph model in this framework consists of nodes that correspond to the camera poses and edges which represent the constraints between the nodes, described by rigid-body transformations. The pose graph model allows the formation of constraints between non-adjacent poses by re-observing the same keypoints from different locations or re-visiting a previously seen region of the scene (loop closures). Our implementation of the global optimization procedure is described in detail below.

4.6.1. Node addition. In this step, a node is constructed and added to the graph. Each node consists of the global camera pose of this frame with respect to a global reference frame and its index. To minimize the computational burden, we do not store points/features for every node. We only store the required information for a subset of nodes (key-frames). The key-frame selection procedure is described in Section 4.6.3.

4.6.2. Edge addition. Each time a new node is added to the graph, an edge that connects this node with the previous one is added to the graph. The edge describes the estimated rigid-body transformation (7) between the two consecutive nodes. In addition, a matrix containing information about the number of matches between the nodes is also stored.

4.6.3. Key-frame selection. The detection of loop closures and addition of constraints between non-adjacent nodes is necessary for reducing the accumulated drift. In order to do so, the current frame is required to be matched with previous frames. However, matching the current frame with all the previous frames is computationally expensive especially as the map gets larger (computational expense increases linearly with the number of estimates[9]). As such, in order to reduce the computational burden, we select a subset of frames ("key-frames") which will be used for the detection of loop closures and the addition of non-adjacent constraints. A number of methods exist for the selection of key-frames. The simplest form of key-frame selection is to select every *nth* frame. Kerl *et al.*[20] proposed a key-frame selection method based on a differential entropy measure. Other solutions such as the one used by Henry *et al.*[2] and Endres *et al.*[9] are to match the current frame with the previous key-frame. A new key-frame is selected only when the number of inliers detected is below a pre-defined threshold (making it a new scene). In order to improve the computational efficiency and avoid the matching step between each frame and the key frame, we add a key-frame when the accumulated rotation or translation from the last key-frame exceeds a pre-defined threshold (15 degrees in rotation or 25 cm in translation). For each key-frame, we extract the ROS keypoints as described in Section 3 and assign SHOT descriptors to those points. The keypoints and their descriptors are stored with each key-frame.

4.6.4. Loop closure detection. Different methods have been proposed for loop closure detection. The simplest form is to match the current frame with all the previously selected key-frames. Other more sophisticated approaches such as the one introduced by Henry *et al.*[2] employ the place recognition method based on the vocabulary tree described by Nister and Stewenius[31] in which the feature descriptors of the candidate key-frames are hierarchically quantized and are represented by a "bag of visual words". In our approach, in order to increase the computational efficiency, we only match the newly selected key-frame (since a key-frame is a representation of a number of adjacent frames) with the stored key-frames instead of matching every frame to all the stored key-frames. A loop closure is detected as follows. First, when a new key-frame is stored, we check if there are previous key-frames with a global pose that is adjacent to the pose of the this key-frame (accumulated rotation and accumulated translation difference between the key-frames is less than a pre-defined threshold). In the next step, only those adjacent key-frames resulting from the previous filtering step are matched with the new key-frame using the mutual consistency matching and robust inlier detection procedures described in Sections 4.4 and 4.3. Note that we have already extracted and stored the ROS keypoints (around 400 points) for each key-frame and their associated descriptors. As such, we can directly match the key-frames without performing the keypoint selection and the computationally expensive descriptor assignment each time. Thus, increasing the computational efficiency significantly. Finally, when a loop closure is detected (number of good matches identified by MSSE during the transformation estimation between new key-frame and neighboring key-frames is greater than a threshold) an edge containing the estimated transformation between the key-frames is added to the graph.

4.6.5. Graph optimization. The goal of this optimization is to find the arrangement of poses that best satisfies those constraints. Generally, this is a non-linear least squares optimization problem which can be described as:

$$\mathbf{C}^* = \operatorname*{argmin}_{C} \sum_{\mathbf{i,j}} \mathbf{w_i}||\mathbf{C_i} - \mathbf{T_{i,j}}\mathbf{C_j}||^{\mathbf{2}}, \tag{9}$$

where $\mathbf{C}^* = \mathbf{C_1^*} \ldots \mathbf{C_n^*}$ is a vector containing all the optimized camera poses (consisting off x, y and z values) and $\mathbf{T_{i,j}}$ is the 6DOF rigid-body transformation between a node pair and $\mathbf{w_i}$ is a weighting factor based on the information matrix described in 4.6 step 2. This minimization problem is solved by a non-linear optimization algorithm (i.e. Levenberg–Marquardt). Figure 7 demonstrates the positive effect of the global optimization on the global consistency of the mapping.

4.7. Map representation

The previous global optimization step produces a globally consistent trajectory (i.e. each node/frame is associated with an optimized global pose). In order to construct the map, first we project the points (from the image frames) to 3D using the following equations:

$$X = (u - c_x) * Z/f_x \tag{10}$$

$$Y = (v - c_y) * Z/f_y, \tag{11}$$

where (u, v) is the image coordinate of an extracted visual feature and (X, Y, Z) is its projected 3D coordinate in the camera optical frame. Z is obtained from the depth image which is provided by the Microsoft Kinect. f_x and f_y are the focal lengths in the horizontal and vertical axes, respectively and (c_x, c_y) is the 2D coordinate of the camera optical center. This is followed by transforming the projected points to a common reference frame using the optimized trajectory. In order to reduce the

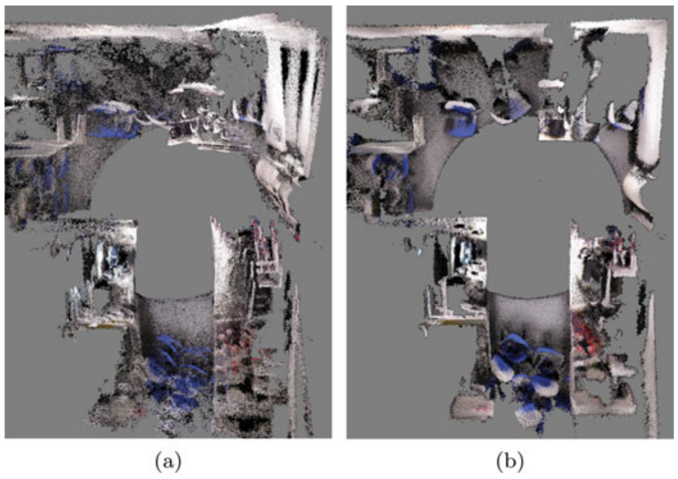


Fig. 7. Example of a reconstructed map of an office scene (a) with (b) without loop closure and global optimization. The loop closure and global optimization step has been able to correct the drifts in the reconstructed map.

computational burden of storing all points provided by the Microsoft Kinect (307,000 points for each frame), we randomly down-sample each point cloud into 5000 points. Figure 2 shows a reconstructed 3D map of an office environment using the proposed method while Fig. 7 compares a map of the same area with and without loop closure and global optimization.

5. Experimental Results

We evaluated the performance of the proposed 3D registration and mapping method by comparing it with the state of the art techniques. All methods were implemented using a Dell Precision M3800, powered by an Intel i7-4702HQ processor, 16 GB of RAM and running on Ubuntu 12.04. The Robot Operating System (ROS Hydro)[32] and PCL 1.7[24] were used for perception and 3D geometry processing. We used a Microsoft Kinect for capturing RGB-D data which operates at a frame-rate of 30 fps in VGA resolution mode. All of the experiments were conducted at typical university offices, which contain many texure-less objects (e.g. plain walls). We performed seven trials for each method and averaged the calculated results and errors. Since we do not have a ground truth trajectory of the camera when using our own datasets, we calculated the rotational and translational errors by returning the camera to the initial (known) location. This procedure is described in detail in Section 5.4. For the evaluation using the public datasets, we used the provided ground truth information which is recorded using a high accuracy motion capture system.

5.1. ROS keypoint repeatability evaluation

The most crucial characteristic of a keypoint extraction method is its repeatability, which is defined as the ability to extract the same set of corresponding points from different point clouds (differences may be due to noise, view point change, occlusion or a combination of the previous factors[33]) of an overlapping scene. The repeatability of the proposed keypoint extraction method is evaluated using the relative repeatability measure described in ref. [33] and outlined below.

- In the first step, we transform the set of keypoints k_s extracted from the source point cloud P_s according to the ground-truth rotation and translation (R_{st}, T_{st}) that aligns the source point cloud P_s and the target point cloud P_t. We then check for the presence of points in P_t in a pre-defined neighborhood ($1\times$ source cloud resolution) of the transformed keypoints. If at least one point is present in P_t in this a neighborhood, the keypoint extracted from P_s is added to a set O_{st}. Finally, the cardinality of this set is calculated.
- In the second step, we calculate the absolute repeatability by first transforming the set of keypoints k_s according to the ground-truth rotation and translation (R_{st}, T_{st}). The transformed keypoint k_s^i is said to be repeatable if the distance from its nearest neighbor, k_t^j, in the set of keypoints extracted

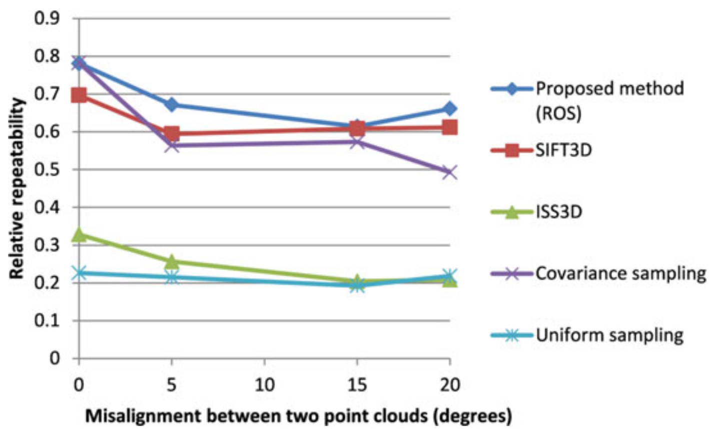


Fig. 8. A comparison between the relative repeatability of different keypoint extraction methods measured using misaligned point clouds. The misalignment angles = 0°, 5°, 15° and 20°.

from the target pointcloud P_t is less than a threshold ρ (2× source cloud resolution) as follows:

$$|R_{st}k_s + T_{st} - k_t| < \rho. \tag{12}$$

The absolute repeatability is obtained by calculating the cardinality of the set of repeatable points RE_{ks} .

- Finally, in order to take into account the amount of overlap between P_s and P_t in the repeatability measure, the relative repeatability r is calculated using the following equation:

$$r = \frac{|RE_{ks}|}{|O_{st}|}. \tag{13}$$

In this evaluation, we compared the repeatability of our method to the state of the art keypoint extraction methods. In all of the experiments, we first measured the relative repeatability by extracting keypoints from two different (aligned) point clouds of the same office scene. We then calculated the relative repeatability using two misaligned point clouds (angle difference between point clouds = 5°, 15° and 20°). The results of this comparison are shown in Fig. 8. The results show that our method obtained the highest relative repeatability scores when compared to other state of the art methods.

5.2. Point cloud preprocessing evaluation

5.2.1. Evaluating the registration using different pass-through filter depth limits . In order to evaluate the effect of point cloud pre-processing on the registration, we compared the accuracy of the registration using different pre-processing parameters. We first evaluate the effect of applying a pass-through filter (removing points that are further than a certain distance from the camera) at different depth limits. In these experiments, we rotated the Microsoft Kinect counter clockwise until it was back to its original position (360°). During this process, the point clouds were acquired, registered, mapped and a global pose of the camera was estimated online. For this evaluation, we used the registration procedure explained in Section 4 and set the number of ROS keypoints to 450. We set the camera's initial pose to the identity matrix. Since the camera is returned back to the exact original position (using markers), we evaluated the accuracy of all methods by calculating the average translational error (in the x, y and z directions) in meters and average rotational error (roll, pitch and yaw angles) in degrees with reference to the initial pose. We also calculated the average time of the registration process in seconds and compared the point cloud sizes before and after uniform sampling. The depth limits that we compared were: no limit (no filtering), 6 m, 5 m, 4 m, 3 m and 2 m. The rotation and translation accuracy for different depth limits are illustrated in Fig. 9 and the full results are outlined in Table I. The results show that the accuracy of the registration increases when far away points are excluded. We found that filtering points that are further than 4 m generally provides the best accuracy. Notice that both the rotation and translation accuracy begin to decrease as we reduce

Table I. Comparison of the accuracy of registration using different pass-through filtering depths. The results are averaged over seven trials.

Depth limit	Trans. error(m)	Rot. error(°)	Registration time(s)	Cloud size	Sample size
No filter	0.176 ± 0.063	7.064 ± 2.898	0.127	307000	5660
6 m	0.141 ± 0.066	7.180 ± 1.888	0.120	283168	5578
5 m	0.115 ± 0.034	6.878 ± 3.137	0.117	279228	4753
4 m	0.118 ± 0.043	3.311 ± 1.988	0.117	267692	4078
3 m	0.149 ± 0.042	6.994 ± 1.878	0.115	256526	3809
2 m	0.434 ± 0.0297	24.294 ± 9.014	0.075	169030	1908

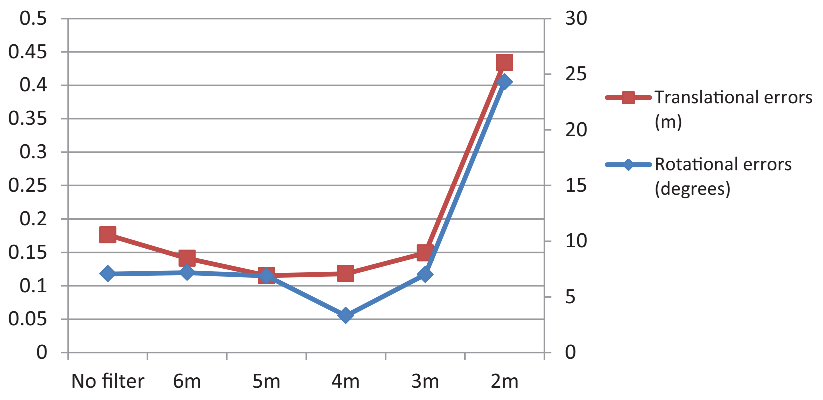
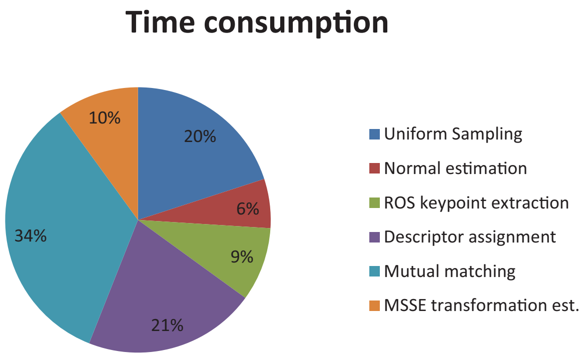


Fig. 9. Accuracy of the registration using different depth limits.



(a)

Fig. 10. Percentage of time consumed by each individual process in the registration procedure.

the depth limit to 3 m and 2 m respectively. This could be a result of loosing important information that are useful for registration (e.g. using a 2 m depth filter results in a point cloud containing only 1908 points after uniform sampling). Also, there is no significant computational time reduction when reducing the depth limit (except for 2m), the main reason is that the resultant point cloud after this operation is sampled to 450, and then only those points are used for feature descriptor assignment and matching, which are the most computationally expensive procedures in the registration process (combined, these procedures account for approximately 55% of the total registration process time). Figure 10 shows the percentage of time consumption by each procedure. Note that uniform sampling accounts for approximately 20%, which explains why the process time was significantly reduced when using the 2 m depth filter (the point cloud size was only 169,030, which is significantly less than the point cloud size after applying the 3 m depth filter).

5.2.2. Evaluating the effect of using an variable vs. fixed voxel size on the point cloud. As it was mentioned in Section 4.1, we use a variable voxel size for the uniform sampling step. In order

Table II. Comparison of the point clouds using a variable vs. fixed voxel size.

Method	Average No. of points	Min No. of points	Max No. of points	Standard deviation	Reg. time (ROS)	Reg. time (Uniform Sampling)
Variable size	3999	2831	5274	523	$0.117s$	$1.989s$
Fixed size	4144	2130	7031	1320	$0.118s$	$2.409s$

to evaluate the effect of using a variable voxel size compared to a fixed one, we performed the proposed registration method on the rotation sequence described in the previous section and set the pass-through filter depth limit to 5m. In this experiment, we are mainly interested in comparing the variance of the point cloud sizes (after uniform sampling) captured by different frames in this sequence and comparing the registration process time. The results of this experiment are outlined in Table II. The results show that when using a variable size, the standard deviation of the point cloud sizes after uniform sampling is reduced by about 60%. In other words, the point cloud sizes did not vary as much in different frames (which contain different point cloud resolutions depending on how far the points are from the camera) when compared to using a fixed voxel size. Also note that since we are extracting ROS keypoints after uniform sampling, there is no significant process time difference between using a variable vs. fixed size. However, if we use all the uniform sampled points for registration (instead of the ROS keypoints), the process time is reduced by approximately 21% when using a variable voxel size .

5.3. 3D descriptors evaluation for point cloud based SLAM

In this experiment, we compared the accuracy of registration using the state of the art 3D descriptors. In all of the experiments, we used the proposed registration method described in Section 4. We fixed all parts of the system and only changed the descriptor method. We evaluated the performance of the methods using the rotation sequence described in Section 5.2.1 and used the recommended default descriptor parameters. The descriptor methods were Point Feature Histogram (PFH),[30] Fast Point Feature Histogram (FPFH),[34] 3D Shape Context (3DSC),[35] Unique Shape Context (USC)[28] and Signatures of Histograms of Orientations (SHOT).[28] The PFH captures information of the geometry surrounding each point by analyzing the difference between the directions of the normals in its vicinity. The PFH does not only pair the query keypoint with its neighbors, but also the neighbors with themselves. As such, the PFH is computationally expensive and would be not suitable for real-time applications. The FPFH extends the PFH by only considering the direct connections between the query keypoint and its surrounding neighbors. To make up for the loss of extra connections between neighbors, an additional step after all histograms have been computed is added: the sub-PFHs of a point's neighbors are merged with its own and is weighted according to the distance between those. This provides point surface information of points as far away as two times the radius used. The 3DSC is a descriptor that works by constructing a structure (sphere) centered at the each point, using the given search radius. The "north pole" of that sphere is pointed in the same direction as the normal at that point. Then, the sphere is divided in 3D regions (regions vary in volume in the radial direction as they are logarithmically spaced so they are smaller towards the center). A descriptor is computed by counting the number of points in each 3D region. The count is weighted by the volume of the bin and the local point density (number of points around the current neighbor). As such, the descriptor is resolution invariant to some extent. In order to account for rotation variances, the support sphere is rotated around the normal N times and the process is repeated for each, giving a total of N descriptors for a point. This procedure is computationally expensive, as such, the USC descriptor extends the 3DSC by defining a local reference frame that provides a unique orientation at each point. This procedure reduces the size of the descriptor when compared to 3DSC, since computing multiple descriptors to account for orientations is not required. SHOT descriptor is similar to 3DSC and USC in that it encodes surface information within a spherical support structure. The sphere around each keypoint is divided into 32 bins, and for each bin, a descriptor containing one variable is computed. This variable is the cosine of the angle between the normal of the keypoint and the neighboring point inside the bin. Finally, the descriptor is obtained by augmenting local histograms. Similar to USC, SHOT descriptors are also rotation invariant.

Table III. Accuracy comparison between offline 3D registration methods using different 3D descriptors for the rotation only sequence. The results are averaged over seven trials.

Method	Trans. error(m)	Rot. error(°)	Registration time(s)
PFH	0.375 ± 0.031	11.388 ± 0.5521	10.939
FPFH	0.189 ± 0.045	10.347 ± 2.899	0.346
3DSC	0.417 ± 0.051	12.467 ± 1.178	5.409
USC	0.221 ± 0.129	8.011 ± 2.960	2.168
SHOT	0.129 ± 0.044	5.880 ± 1.041	0.154

Table IV. Accuracy comparison between online 3D registration methods using different 3D descriptors for the rotation only sequence. The results are averaged over seven trials.

Method	Trans. error(m)	Rot. error(°)	Registration time(s)
FPFH	0.182 ± 0.076	3.949 ± 1.019	0.331
SHOT	0.122 ± 0.057	1.968 ± 1.048	0.154

Some of the above descriptors are computationally expensive, and as such, would not be suitable for cases with fast camera movement. Thus, we performed two experiments: offline and online. The results of the offline evaluation are outlined in Table III. The results of this experiment show that both FPFH and USC are indeed an improvement over PFH and 3DSC, respectively. FPFH significantly improves the computational efficiency by a factor of 31 when compared to PFH. Whereas USC improves the efficiency by a factor of 2.6. FPFH improves the translational and rotational registration accuracy by 49.6% and 9.1% respectively in comparison to PFH. USC also improves the translational and rotational registration accuracy by 47% and 35.7% respectively when compared to 3DSC. Overall, SHOT descriptors significantly outperformed all of the other descriptor both in terms of accuracy and computational efficiency. As can be seen in Table III, PFH, 3DSC and USC have relatively high processing times. As such, it was not possible to perform online registration with these methods and we only compared the methods that were able to perform the online registration process. Those methods were FPFH and SHOT descriptors. The results of the online registration comparison are outlined in Table IV. The results show that SHOT descriptors outperforms FPFH both in terms of registration accuracy and efficiency. SHOT improves the translational and rotational accuracy by 32.9% and 50% respectively and improves the computational efficiency by 53.4% when compared to FPFH.

5.4. Registration accuracy and efficiency comparison using different keypoint extraction methods

In this experiment, we compared the accuracy of registration using our proposed keypoint extraction and some of the best available methods. In the first experiment, we used the rotation sequence described in Section 5.2.1. In the second experiment, the Micosoft Kinect was rotated 180°, then translated for 1.5 m, followed by another 180° rotation and finally translated back to its original position. The compared methods are: SIFT3D,[24] ISS keypoints,[27] geometrically stable points[26] (covariance sampling) and uniform sampling (we set a large search radius to obtain around 400 points). We compare the average translational error, average rotational error and average time of the registration process. In addition to those measures, we also compared the average keypoint extraction time (in seconds) and average number of extracted keypoints. The results of both experiments are outlined in Tables V and VI. The results show that our proposed method significantly outperforms the other methods in terms of both the translational and rotational accuracy. In terms of computational efficiency, our method was slightly slower than covariance sampling and uniform sampling. However, this does not significantly affect the registration efficiency, as those methods required similar times for registration. Our experiments show that SIFT3D was the most computationally expensive keypoint extraction algorithm.

Table V. Comparison of the accuracy of 3D registration using different keypoint extraction methods for the rotation only sequence. Translational errors are in meters, rotational errors are in degrees and times are in seconds. The results are averaged over seven trials.

Method	Trans. error	Rot. error	Extraction time	Registration time	Number of points
ROS	0.086 ± 0.01	2.2 ± 0.76	0.0091	0.12	400
SIFT3D	0.118 ± 0.02	3.12 ± 1.73	0.295	0.45	402
ISS keypoints	0.135 ± 0.04	8.9 ± 3.11	0.018	0.09	160
Covariance samp.	0.304 ± 0.07	17.4 ± 6.46	0.0065	0.113	453
Uniform samp.	0.107 ± 0.02	5.65 ± 2.51	0.0015	0.102	365

Table VI. Comparison of the accuracy of 3D registration using different keypoint extraction methods for the rotation + translation sequence. Translational errors are in meters, rotational errors are in degrees and times are in seconds. The results are averaged over seven trials.

Method	Trans. error	Rot. error	Extraction time	Registration time	Number of points
ROS	0.134 ± 0.03	3.6 ± 1.41	0.0078	0.112	400
SIFT3D	0.147 ± 0.03	4.82 ± 2.12	0.28	0.404	393
ISS keypoints	0.26 ± 0.06	7.6 ± 3.73	0.018	0.09	157
Covariance samp.	0.22 ± 0.08	10.83 ± 2.92	0.0056	0.112	450
Uniform samp.	0.19 ± 0.07	4.5 ± 1.41	0.0014	0.102	367

5.5. *Registration comparison using sparse keypoints vs. a denser point cloud*

In this experiment, we demonstrate the effectiveness of our informative sampling keypoint extraction method by comparing its registration accuracy and computational efficiency with another method that uses a much denser point cloud for registration using the rotation sequence described above. The point cloud was processed as it was described in Section 4.1. Table VII shows the results of this experiment. The results show that our method was able to achieve a very similar accuracy, despite using only 6.4% of the processed point cloud points and 0.13% of the original point cloud. The computational time for registration was significantly improved by a factor of 24.

5.6. *Comparison of different registration methods*

In this experiment, we evaluated the performance of our registration method by comparing it to two state of the art registration techniques using the two sequences mentioned above. In the first of the compared methods, SIFT3D keypoints were extracted and matched using SHOT descriptors (simply because those were shown to outperform other methods[36]). Then SAC-IA[34] (a sampling consensus method that is similar to RANSAC but instead of selecting random samples between the source and target clouds, samples are selected based on points with the most similar feature histograms) was applied for rejecting false matches and initial transformation estimation, followed by ICP (in order to achieve real time performance, we performed ICP on the uniformly sampled point cloud instead of the full point cloud) for refining the initial transformation. The second compared method uses SIFT3D keypoint extraction and matching, followed by a pre-rejection RANSAC algorithm.[37] This method adds an additional verification step to the standard RANSAC algorithm which eliminates some false matches by analyzing their geometry. Tables VIII and IX show the results of those comparisons. Our proposed registration method significantly outperforms the other methods both in terms of accuracy and computational efficiency.

5.7. *Evaluation of global optimization*

In this experiment, we compared the performance of the proposed mapping method with and without global optimization in terms of accuracy and efficiency. In the first comparison, we performed the mapping method on the translation and rotation sequence described in Section 5.4. In the second experiment, we used a sequence that is similar to the rotation only sequence described in Section 5.4. However, the Microsoft Kinect is rotated 1080° and back to its original position (full rotation $\times 3$). The reason behind rotating the Kinect multiple times is that we wanted the drift to be significant for

Table VII. Registration comparison using a sparse point cloud vs. a denser point cloud. The results are averaged over seven trials.

Method	Trans. error(m)	Rot. error(°)	Registration time(s)	Number of points
Proposed registration	0.086 ± 0.01	2.2 ± 0.76	0.12	400
Uniform sampling	0.006 ± 0.004	2.2 ± 0.54	2.88	6205

Table VIII. Accuracy comparison of 3D registration methods using the rotation only sequence. The results are averaged over seven trials.

Method	Trans. error(m)	Rot. error(°)	Registration time(s)
Proposed registration	0.086 ± 0.01	2.2 ± 0.76	0.12
SIFT3D+SAC-IA+ICP	0.122 ± 0.03	4.667 ± 1.63	0.68
SIFT3D+pre-rejective RANSAC	0.305 ± 0.14	13.92 ± 5.72	0.65

Table IX. Accuracy comparison of 3D registration methods using the rotation + translation sequence. The results are averaged over seven trials.

Method	Trans. error(m)	Rot. error(°)	Registration time(s)
Proposed registration	0.134 ± 0.03	3.6 ± 1.41	0.112
SIFT3D+SAC-IA+ICP	0.151 ± 0.04	6.5 ± 2.52	0.68
SIFT3D+pre-rejective RANSAC	1.44 ± 0.27	29.8 ± 22.68	0.65

Table X. Comparison of the accuracy of registration with and without global optimization and loop closure using the rotation + translation scene. The duration of this sequence is 37.71s. The results are averaged over seven trials.

Method	Trans. error(m)	Rot. error(°)	Registration time(s)
Method without global optimization	0.134 ± 0.03	3.6 ± 1.41	0.112
Method with global optimization	0.02 ± 0.007	1.4 ± 0.29	0.157
RGBD-SLAM v2	0.102 ± 0.03	2.24 ± 1.23	0.4

this evaluation (since 1 rotation does not result in a very large drift) in order to showcase the effect of global optimization and loop closure detection. The results of those comparisons are outlined in Tables X and XI. The results show that the average registration error was reduced by a around 85.7% and 92.7% for the first and second experiments, respectively. However, the average time of registration was increased by 40.1% in the first experiment and 136% in the second experiment. In the first experiment, the average number of nodes and keynodes that were added to the pose graph was 180 and 40 respectively, and the number of detected loop closures was 78. In the second experiment, the average number of nodes and keynodes were 144 and 89 respectively. The average number of loop closures detected was 389, which was much higher than the number of loop closures detected in the first sequence. The reason is that the Microsoft Kinect was rotated multiple times and re-observed the same scene many times. As a result, 389 additional constraints were added to the pose graph, which explains the 136% increase in registration time when mapping this sequence. Figure 7 shows a top view of the reconstructed maps with and without employing global optimization. The figure clearly shows that the addition of global optimization reduces the drift and produces a more globally consistent map. Finally, the above results were compared with the state of the art RGBD-SLAM v2[9] method (using the default parameters) which also utilizes the **g²o** framework. The results are outlined in Tables X and XI. The results show that the proposed method (with global optimization) outperforms RGBD-SLAM in both accuracy and efficiency using the two aforementioned sequences.

Table XI. Comparison of the accuracy of registration with and without global optimization and loop closure using the multiple rotation scene. The duration of this sequence is 48.4s. The results are averaged over seven trials.

Method	Trans. error(m)	Rot. error(°)	Registration time(s)
Method without global optimization	0.138 ± 0.05	8.5 ± 1.84	0.122
Method with global optimization	0.01 ± 0.003	0.65 ± 1.41	0.288
RGBD-SLAM v2	0.01 ± 0.006	3.396 ± 1.12	0.44

Table XII. Evaluation of different key-frame selection criteria on the registration procedure.

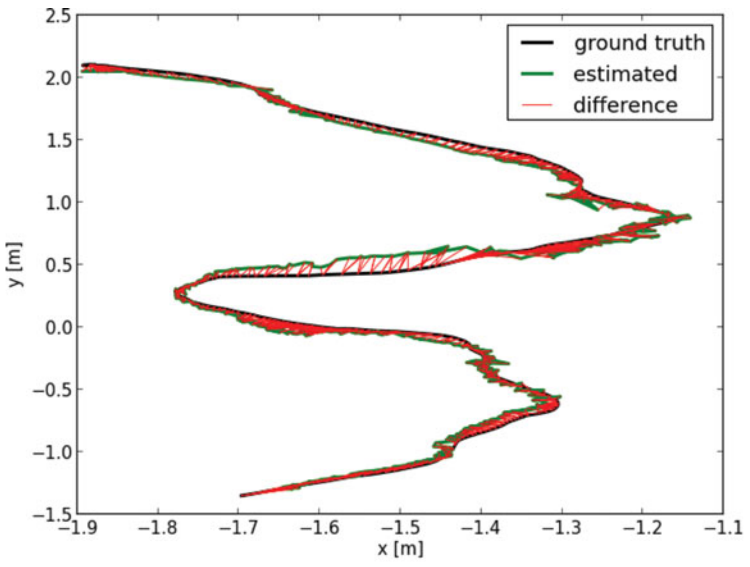
Trans.-Rot. threshold	Trans. error(m)	Rot. error(°)	No. of key-frames	Total No. of frames	Key-frames to frames ratio	Registration time(s)
No threshold	0.011 ± 0.001	0.423 ± 0.172	116	116	1	1.325
0.07 m – 5°	0.025 ± 0.001	0.929 ± 0.413	62	116	0.534	0.648
0.1 m – 7°	0.022 ± 0.002	0.586 ± 0.264	53	130	0.407	0.553
0.2 m – 10°	0.021 ± 0.007	0.560 ± 0.172	54	150	0.360	0.489
0.25 m – 15°	0.026 ± 0.008	0.760 ± 0.243	32	160	0.200	0.312
0.3 m – 25°	0.056 ± 0.047	2.212 ± 1.862	27	164	0.164	0.300

Table XIII. The median, mean and standard deviation of the absolute trajectory error (ATE) between the estimated trajectory and the ground truth data for the proposed and RGBD-SLAM v2 methods. The results are averaged over seven trials.

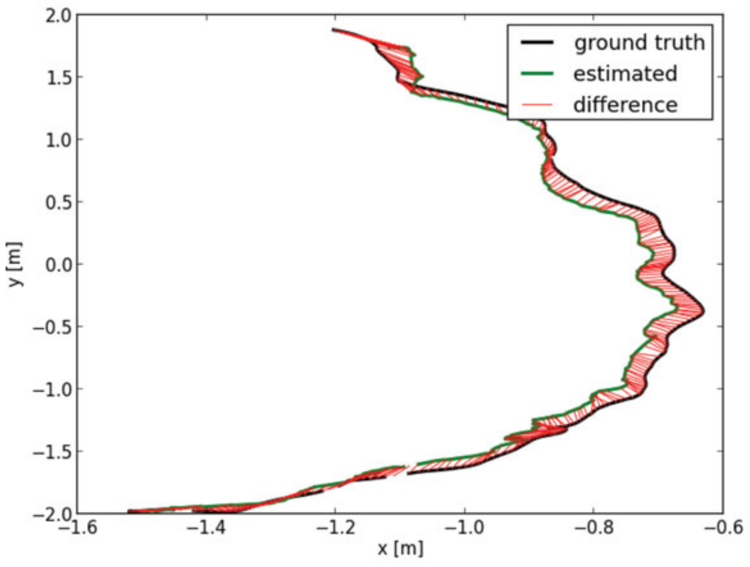
	Proposed Method			RGBD-SLAM v2		
Dataset	Median ATE	Mean ATE	SD	Median ATE	Mean ATE	SD
fr1/xyz	0.0250 m	0.0306 m	0.0184 m	0.0412 m	0.0545 m	0.0468 m
fr1/desk	0.0695 m	0.0815 m	0.047 m	0.0548 m	0.0700 m	0.048 m
fr1/desk2	0.1032 m	0.1165 m	0.0679 m	0.0911 m	0.1013 m	0.057 m
fr3/structure_notexture_near	0.0726 m	0.0927 m	0.07 m	0.1679 m	0.1893 m	0.155 m
fr3/structure_notexture_far	0.0413 m	0.0482 m	0.029 m	0.1244 m	0.1513 m	0.119 m
fr3/structure_texture_near	0.0648 m	0.0859 m	0.059 m	0.0607 m	0.0787 m	0.05 m
fr3/structure_texture_far	0.0664 m	0.0768 m	0.038 m	0.3823 m	0.5387 m	0.291 m
fr3/nostructure_texture_near_ with_loop	1.8890 m	1.9038 m	0.349 m	0.0360 m	0.0523 m	0.04 m

5.8. Evaluation of key-frame selection criteria

As described in Section 4.6.3, a key-frame is selected when the accumulated translation or rotation exceeds a certain threshold. We evaluated the performance of the registration algorithm using different translation/rotation combinational thresholds using the rotation only sequence. In the first experiment, no threshold was used, in other words, all frames were used in the global optimization. We then evaluated the following translation/rotation thresholds: 0.07 m/5°, 0.1m/7°, 0.2 m/10°, 0.25m/15° and 0.3 m/25°. The results of this experiment are outlined in Table XII. The most accurate results are unsurprisingly the ones obtained by using all the frames in the optimization. However, this approach increases the computational complexity of the system significantly and makes it unsuitable for real-time applications (computational time is 1.325 s - less than a frame/second). The following four thresholds combinations (up to 0.25 m/15°) provided comparable registration accuracy. However, the computational time decreased as we increased the thresholds. For instance, the computational time when using the 0.25 m/15° threshold combination is 51.8% lower than using the 0.07 m/5° threshold combination, since only 20% of all frames are selected as key-frames, as opposed to 64.8% of all frames. Also note that when selecting the 0.3 m/25° threshold combination, the translational



(a)



(b)

Fig. 11. Visualization of the absolute trajectory error (ATE) which was calculated after performing our proposed method on the: (a) "freiburg3_structure_texture_near" sequence (b) "freiburg3_structure_no_texture_far_".

and rotational accuracy is reduced by 53% and 65% respectively when compared to the 0.25 m/15° threshold. This is due to the key-frames being too far apart and as such, the matching between key-frames was reduced (which results in less constraints in the pose graph). We also experimented with larger thresholds, but we found that anything higher than the 0.3 m/25° threshold combination results in no additional non-adjacent constraints being added to the pose graphs. This results in a system with similar registration accuracy to the registration method that does not employ global optimization, but with the added computational expenses of the global optimization procedure.

5.9. Proposed method evaluated using public RGB-D dataset

We evaluated the performance of the proposed 3D SLAM method using the publicly available RGB-D benchmark provided by the Technical University of Munich.[8] The datasets contain various scenes captured by an RGB-D camera and provide very accurate ground truth information that were obtained by an external motion capture system. For the evaluation of the compared methods, we utilize the absolute trajectory error (ATE) metric provided by[8] in which the absolute distances between the estimated and ground truth data are calculated. Given a sequence of camera poses from the estimated trajectory $\mathbf{P1}, \ldots, \mathbf{Pn} \in SE(3)$ and from the ground truth trajectory $\mathbf{Q1}, \ldots, \mathbf{Qn} \in SE(3)$, the ATE at time-step i can be obtained using the following equation:

$$ATE_i = Q_i^{-1} S P_i, \tag{14}$$

where S denotes the rigid body transformation that aligns the coordinate frames of P_i and Q_i. We then calculated the mean and median errors over all time indices n of the translational components of the ATE. An example of the ATE between the estimated trajectory (using the proposed method) and ground truth data is illustrated in Fig. 11. For this evaluation, we performed the proposed SLAM method using the scenes available from the "freiburg3 structure vs. texture" category and compared it with the RGBD-SLAM v2 (using the default SURF keypoints and descriptors). Table XIII shows a summary of the results of this experiment. We first evaluated both methods using the "freiburg3_structure_notexture_near" and "freiburg3_structure_notexture_far" sequences. Both sequences are captured by moving the sensor along a zig-zag structure that is built from wooden panels and contains very limited visual information. As we expected, our method clearly outperforms the RGBD-SLAM v2 method using those sequences since RGBD-SLAM only uses visual information for registration. We then evaluated both methods using the "freiburg3_nostructure_texture_near_withloop" sequence which is captured by moving the Microsoft Kinect over a highly textured planar surface. Due to the very limited structure information in this scene, our method struggles registering this sequence and the RGBD-SLAM v2 method clearly outperformed our method. We then used the "freiburg3_structure_texture_near" and "freiburg3_structure_texture_far" sequences, which consist of moving the camera along a zig-zag structure that is fully wrapped in a colorful plastic foil. In the first of the aforementioned sequences, the performance of both methods was very comparable, with the RGBD-SLAM v2 method having the slight advantage. For the second sequence, our method significantly outperformed the RGBD-SLAM method since RGBD-SLAM seems to fail around the 15s mark due to a false loop closure which disrupts and breaks the registration process. The RGBD-SLAM v2 fails to recover from this, which results in a very high ATE. We repeated this experiment many times and also tried using RGBD-SLAM's other recommended keypoints (SIFT and ORB), resulting with the same failure each time. We then evaluated both methods using the "freiburg1_xyz", "freiburg1_desk" and "freiburg1_desk2" sequences. All of those sequences are captured by moving the camera in a postgraduate office environment and contain varying amounts of structure and texture information. In addition, the "freiburg1" sequence is challenging due to fast camera motions. The performance of both the compared methods using these sequences is comparable, with our method having the advantage in first sequence and the RGBD-SLAM v2 having the slight advantage in the last two. Also note that at around 9 seconds in the "freiburg1_desk2", the camera is moved over a planar table containing very little structure, which results in a slight misalignment. This explains the relatively high ATE of the proposed method when using this sequence. In such cases (where there is a lack of either texture of structure information), it might be better to use a combination of texture and structure information. This is something we would like to add to our method in the future.

6. Conclusions

In this paper, we presented a novel real-time 3D SLAM system that uses only the depth information provided by RGB-D sensors, without relying on texture information. As such, our method is well suited to perform localization and mapping in texture-less environments commonly encountered in both office and industrial buildings. Since registration using the dense point cloud is a very computationally expensive operation, we propose a novel sampling scheme that informatively selects the points carrying the most useful information using the statistical analysis of their flatness. We showed that the proposed keypoint extraction method outperforms other state of the art methods in terms of accuracy and repeatability and performs comparably in terms of efficiency. We also showed that our proposed registration method is faster and more accurate compared to other well-known registration methods.

Having said that, depth only registration methods struggle when registering scenes with very little structure. As such, we plan on aiding the depth information with visual information when available. Another aspect that our method struggled with was the registration in the presence of multiple motions (such as people walking in the scene). Since we use a high-breakdown estimator in our method, we think it should be possible to segment different motions explicitly and register those separately. This is part of our future work.

Acknowledgment

The first author would like to acknowledge the financial support provided by the Australian Postgraduate Award (APA) scholarship. This research was in part supported under Australian Research Council's Linkage Projects funding scheme (LP130100521).

References

1. 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," *Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Basel, Switzerland (2011) pp. 127–136.
2. P. Henry, M. Krainin, E. Herbst, X. Ren and D. Fox, "Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments," *Int. J. Robot. Res.* **31**(5), 647–663 (2012).
3. F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers and W. Burgard, "An Evaluation of the rgb-d Slam System," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, St - Paul, Minnosota, USA (2012) pp. 1691–1696.
4. H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part i," *Robot. Autom. Mag. IEEE* **13**(2), 99–110 (2006).
5. A. Bab-Hadiashar and D. Suter, "Robust segmentation of visual data using ranked unbiased scale estimate," *Robotica* **17**(6), 649–660 (1999).
6. R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige and W. Burgard, "g^2o: A General Framework for Graph Optimization," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China (2011) pp. 3607–3613.
7. K. Yousif, A. Bab-Hadiashar and R. Hoseinnezhad, "Real-Time rgb-d Registration and Mapping in Texture-Less Environments using Ranked Order Statistics," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, IEEE, Chicago, Illinois, USA (2014) pp. 2654–2660.
8. J. Sturm, N. Engelhard, F. Endres, W. Burgard and D. Cremers, "A Benchmark for the Evaluation of rgb-d Slam Systems," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Vilamoura - Algarve, Portugal (2012) pp. 573–580.
9. F. Endres, J. Hess, J. Sturm, D. Cremers and W. Burgard, "3-d mapping with an rgb-d camera," *IEEE Trans. Robot.* **30**, 177–187 (2014).
10. F. Fraundorfer and D. Scaramuzza, "Visual odometry: Part ii: Matching, robustness, optimization, and applications," *Robot. Autom. Mag. IEEE* **19**(2), 78–90 (2012).
11. A. Segal, D. Haehnel and S. Thrun, "Generalized-icp," *Robot.: Sci. Syst.* **2**, 4 (2009).
12. D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.* **60**(2), 91–110 (2004).
13. H. Bay, T. Tuytelaars and L. Van Gool, "Surf: Speeded Up Robust Features," *Computer Vision–ECCV* **3951**, 404–417 (Graz, Austria, 2006).
14. E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "Orb: An Efficient Alternative to Sift or Surf," *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Barcelona, Spain (2011) pp. 2564–2571.
15. H. Du, P. Henry, X. Ren, M. Cheng, D. Goldman, S. Seitz and D. Fox, "Interactive 3d Modeling of Indoor Environments with a Consumer Depth Camera," *Proceedings of the 13th International Conference on Ubiquitous Computing*, ACM, Beijing, China (2011) pp. 75–84.
16. C. Audras, A. Comport, M. Meilland and P. Rives, "Real-Time Dense Appearance-Based Slam for rgb-d Sensors," *Australasian Conference on Robotics and Automation*, Melbourne, Australia (2011).
17. T. Whelan, M. Kaess, M.F. Fallon, H. Johannsson, J.J. Leonard and J.B. McDonald, "Kintinuous: Spatially Extended Kinectfusion," *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, Australia (July 2012).
18. A. Bachrach, S. Prentice, R. He, P. Henry, A. Huang, M. Krainin, D. Maturana, D. Fox and N. Roy, "Estimation, planning, and mapping for autonomous flight using an rgb-d camera in gps-denied environments," *Int. J. Robot. Res.* **31**(11), 1320–1343 (2012).

19. K. Yousif, A. Bab-Hadiashar and R. Hoseinnezhad, “3d Registration in Dark Environments using rgb-d Cameras,” *Proceedings of the International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, Hobart, Tasmania, Australia (2013) pp. 1–8.
20. C. Kerl, J. Sturm and D. Cremers, “Dense Visual Slam for rgb-d Cameras,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan (2013) pp. 2100–2106.
21. M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich and A. Kolb, “Real-Time 3d Reconstruction in Dynamic Scenes using Point-Based Fusion,” *Proceedings of the International Conference on 3DTV-Conference*, Tokyo, Japan (2013) pp. 1–8.
22. G. Hu, S. Huang, L. Zhao, A. Alempijevic and G. Dissanayake, “A Robust rgb-d Slam Algorithm,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Vilamoura - Algarve, Portugal (2012) pp. 1714–1719.
23. C. Harris and M. Stephens, “A Combined Corner and Edge Detector,” *Alvey Vision Conference*, vol. 15, Manchester, UK (1988), p. 50.
24. R. B. Rusu and S. Cousins, “3d is Here: Point Cloud Library (pcl),” *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Shanghai, China (2011) pp. 1–4.
25. S. Filipe and L. A. Alexandre, “A Comparative Evaluation of 3d Keypoint Detectors,” *Proceedings of the 9th Conference on Telecommunications, Conftele*, Castelo Branco, Portugal (2013) pp. 145–148.
26. N. Gelfand, L. Ikemoto, S. Rusinkiewicz and M. Levoy, “Geometrically Stable Sampling for the icp Algorithm,” *Proceedings of the 4th International Conference on 3-D Digital Imaging and Modeling*, Banff, Canada (2003) pp. 260–267.
27. Y. Zhong, “Intrinsic Shape Signatures: A Shape Descriptor for 3d Object Recognition,” *Proceedings of the IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)*, Kyoto, Japan (2009) pp. 689–696.
28. F. Tombari, S. Salti and L. Di Stefano, “Unique Signatures of Histograms for Local Surface Description,” *Computer Vision–ECCV*, Springer, Haraklion, Crete, Greece (2010) pp. 356–369.
29. R. Hoseinnezhad, A. Bab-Hadiashar and D. Suter, “Finite sample bias of robust estimators in segmentation of closely spaced structures: A comparative study,” *J. Math. Imaging Vis.* **37**(1), 66–84 (2010).
30. R. B. Rusu, “Semantic 3d object maps for everyday manipulation in human living environments,” *KI-Künstliche Intelligenz* **24**(4), 345–348 (2010).
31. D. Nister and H. Stewenius, “Scalable Recognition with a Vocabulary Tree,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, New York, USA (2006) pp. 2161–2168.
32. M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng, “Ros: An Open-Source Robot Operating System,” *ICRA Workshop on Open Source Software*, Kobe, Japan (2009) p. 5.
33. S. Salti, F. Tombari and L. Di Stefano, “A Performance Evaluation of 3d Keypoint Detectors,” *Proceedings of the International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, Hangzhou, China (2011) pp. 236–243.
34. R. B. Rusu, N. Blodow and M. Beetz, “Fast Point Feature Histograms (fpfh) for 3d Registration,” *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA’09*, Kobe, Japan (2009) pp. 3212–3217.
35. A. Frome, D. Huber, R. Kolluri, T. Bülow and J. Malik, “Recognizing Objects in Range Data using Regional Point Descriptors,” *Computer Vision-ECCV*, Springer, Prague, Czech Republic (2004) pp. 224–237.
36. L. A. Alexandre, “3d Descriptors for Object and Category Recognition: A Comparative Evaluation,” *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, no. 2, Vilamoura, Portugal (2012).
37. A. G. Buch, D. Kraft, J.-K. Kämäräinen, H. G. Petersen and N. Krüger, “Pose Estimation using Local Structure-Specific Shape and Appearance Context,” *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA*, Karlsruhe, Germany (2013) pp. 2080–2087.