Fast and Effective Loop Closure Detection to Improve SLAM Performance

Article *in* Journal of Intelligent & Robotic Systems · March 2019

DOI: 10.1007/s10846-017-0718-z

CITATIONS
29

READS
3,588

2 authors:

Oguzhan Guclu
Hacettepe University
10 PUBLICATIONS **187** CITATIONS
SEE PROFILE

Ahmet Burak Can
Hacettepe University
63 PUBLICATIONS **767** CITATIONS
SEE PROFILE

Noname manuscript No.
(will be inserted by the editor)

Fast and Effective Loop Closure Detection to Improve SLAM Performance

Oguzhan Guclu · Ahmet Burak Can

Received: 30 September 2016 / Accepted: 24 September 2017

Abstract A fundamental component of simultaneous localization and mapping systems is loop closure detection. For consistent mapping, accurate loop closure detection is crucial to reduce the drift of the estimated trajectory. As the map size increases, loop closure detection performance becomes more critical, but it gets harder and needs more computational time to find correct loop closure candidates. This paper presents an extension to a state-of-the-art RGB-D SLAM system to increase accuracy of large-scale mapping in real-time. The proposed extension uses a straightforward visual place recognition method to determine loop closure candidates. The method combines global and local image features through employing image histograms and keypoint matching. Four different place recognition techniques composed of complementary steps of the method are studied: histogram only, brute-force keypoint matching, hierarchical clustering, and adaptive thresholding. The extended RGB-D SLAM system is assessed on a popular dataset in terms of accuracy and speed. The quantitative results show that the proposed method improves accuracy up to ∼42% and works fast enough to meet real-time requirements. The method enables to perform real-time large-scale indoor mapping effectively on CPU.

Keywords SLAM · loop closure · place recognition · histogram · keypoint matching

A conference version of this paper is presented at ICARSC 2016.

O. Guclu
Department of Computer Engineering, Hacettepe University, Ankara, Turkey
E-mail: guclu.oguzhan@outlook.com

A. B. Can
Department of Computer Engineering, Hacettepe University, Ankara, Turkey
Tel.: +90-312-2977500
Fax: +90-312-2977502
E-mail: abc@hacettepe.edu.tr

1 Introduction

The ability to build a map of an unknown environment and locate itself in the map at the same time, known as Simultaneous Localization and Mapping (SLAM), is an essential requirement for an autonomous robot to perform various navigation and manipulation tasks. Creating a metrically accurate map in conjunction with performing precise localization is a nontrivial problem. In the robotics community, SLAM is one of the most actively studied problems, which has attracted an increasing attention in the recent decades.

Visual sensors are widely used as primary sensor to develop 3D SLAM methods due to their rich information capability, compactness, and low price. For monocular vision based approaches, undetermined absolute scale is a challenging factor since a single camera provides only 2D measurements of the environment [30]. In stereo vision based methods, the scale ambiguity problem can be addressed by directly computing depth from stereo pairs but texture is needed to calculate reliable depth information [30]. RGB-D sensors such as the Microsoft Kinect are able to provide dense depth information along with color images in real-time. Thus, the release of RGB-D sensors has led to great progress in SLAM by enabling researchers to exploit detailed depth maps and color images together with affordable prices.

Graph based formulation is a commonly used approach to solve the SLAM problem, in which nodes represent robot poses and edges connecting the nodes hold constraints between related poses [13]. In graph based SLAM approaches, the graph is constructed by two functions: odometry estimation and loop closure detection. The odometry estimation computes robot motion by registering data frames from the sensor. The loop closure detection is the task of recognizing previously visited areas of the environment. Accurate loop closure detection has critical importance for the map accuracy since it enables to reduce the accumulating trajectory drift caused by odometry errors [35]. For accurate loop closure detection, an effective place recognition method is needed to determine loop closure candidates. Besides the inherent difficulties of place recognition, another important challenge is the expansion of the search space by movement of the robot [23]. As the mapping environment expands, loop closure detection becomes even more important due to the increasing drift. However, it becomes more difficult and computationally expensive to recognize places correctly because more observations are needed to be searched. Furthermore, the place recognition method should be fast enough to allow real-time processing. Due to this challenging problem, most RGB-D based SLAM methods are unable to work in large-scale environments.

In this paper, we extend a state-of-the-art RGB-D SLAM system [9] with an effective place recognition method for loop closure detection in large-scale environments. The RGB-D SLAM system [9] produces successful results for small-sized environments. However, the system is not suitable for mapping larger environments since it tries to detect loop closures randomly. The proposed extension employs a visual place recognition method to determine loop closure candidates efficiently for more precise trajectory estimation. The method uses global and local image features together. While image histogram is employed as a global feature by its efficiency, keypoint matching is used to exploit local image features for robustness. The method firstly finds a set of candidates by using histogram similarity and then applies keypoint matching on the set to determine loop closure candidates. The method is highly applicable to the feature based SLAM systems since it uses the

local image features constructed in the odometry estimation process. This allows an important cost reduction.

The proposed method contains three complementary components that can be applied as different techniques: histogram only, brute-force or hierarchical clustering based keypoint matching, and adaptive thresholding. The first two techniques are studied in our previous work [15]. While histogram only place recognition offers a fast and computationally inexpensive approach, brute-force keypoint matching provides more accuracy with extra computational overhead. In this study, hierarchical clustering and adaptive thresholding based techniques are introduced to provide more computational efficiency while maintaining robustness. The hierarchical clustering technique reduces keypoint matching cost with close accuracy to the brute-force approach. The adaptive thresholding technique eliminates outlier candidates according to histogram similarity. Thus, it reduces computational cost and generally increases accuracy.

The extended RGB-D SLAM system with the proposed visual place recognition method is evaluated on a commonly used RGB-D benchmark dataset [33]. According to the quantitative results, the proposed method provides substantial accuracy improvement. The method allows to choose better candidates especially in large-scale mapping. In the experiments, the most effective results are obtained with the adaptive thresholding technique, where accuracy is increased up to $\sim$42%. Compared with the RGB-D SLAM system [9], the adaptive thresholding technique creates $\sim$8% and $\sim$11% extra computational costs for grayscale and RGB histograms respectively. In other words, the proposed method can still support real-time operation on CPU. The resulting SLAM system has the ability to work well in large-scale indoor environments with challenging conditions.

2 Related Work

In RGB-D sensor based SLAM systems, considering the frame registration technique, two main approaches are based on either utilizing sparse features or employing dense mechanisms. The feature based techniques make use of local image features by extracting and matching salient keypoints from the images. On the other hand, the dense approaches exploit all available information in the whole images directly by performing pixel-wise error minimization.

Instead of processing all dense information in the frames, the feature based methods provide significant computational speedup by applying keypoint extraction. The general approach is to match keypoints between frames and apply RANSAC [10] for registration. One of the first RGB-D sensor based SLAM methods was developed by Henry et al. [17]. The method relies on using sparse keypoints along with geometrical registration. SIFT keypoints [22] are extracted from color frames and located in 3D utilizing depth frames. Then the transformation is computed by performing RANSAC on the corresponding points. The RANSAC transformation is used to initialize ICP algorithm [31] and the transformation error is reduced with the ICP alignment step. Loop closures are determined according to the number of inliers obtained by applying RANSAC transformation estimation between the current keyframe and all previous keyframes. Graph optimization is performed with TORO [12] after each detected loop closure. An improved version of this system is presented in [18], which uses FAST detector [28] and Calon-

der descriptor [6] combination for keypoint extraction rather than SIFT. In order to reduce computational cost, ICP algorithm is not performed when the number of inliers obtained after RANSAC is greater than a threshold. For loop closure detection, bag of words approach is used to filter the previous keyframes before RANSAC alignment. Sparse bundle adjustment [20] is employed for optimizing the graph.

Besides performing RANSAC alignment with all the keyframes or a filtered subset for loop closure detection, another approach is sampling from previous frames. The feature based method proposed by Endres et al. [8] uses SIFT, ORB [29], and SURF [5] for extracting keypoints from color frames. 3 direct predecessors and 17 uniformly sampled frames are selected from earlier frames for registration with each incoming frame using RANSAC. The graph is optimized via g2o framework [21]. Maier et al. [24] employ RANSAC transformation over extracted keypoints for motion estimation similar to the work of Henry et al. [18]. Loop closure detection is performed by sampling 20 past frames and registering each sampled frame with the current one. Instead of optimizing the whole graph, they propose to partition the graph into submaps and optimize each submap individually. By this way, the optimization problem is handled through dividing it into subproblems.

Contrary to the feature based methods, the dense approaches use all the available frame data without extracting salient regions. They employ pixel-wise constraints for motion estimation. Some dense approaches create the map incrementally, without applying loop closure detection. The KinectFusion [26] system uses a truncated signed distance function (TSDF) to represent the scene by updating it continuously with the sensor movements. The volumetric model is used to predict a surface by raycasting and ICP registration is performed with the incoming depth frame to estimate the sensor pose. Then the TSDF is updated by integrating the surface measurement. The system is implemented on GPU. Also it is restricted to map small-scale environments since it does not have a loop closure detection component to reduce the increasing drift and the volumetric model needs too much memory. The KinectFusion is extended by Kintinuous [37], which dynamically moves the volumetric structure. The TSDF is shifted virtually with the sensor movements and new areas of the environment enter the volumetric model when the movement is bigger than a threshold. The region leaving the TSDF is used to extract a point cloud which is integrated into a mesh representation. The environment is mapped continuously through moving the TSDF along the trajectory. Gutierrez-Gomez et al. [16] propose to utilize constant uncertainty of inverse depth for parameterizing the geometric error.

Employing loop closure detection with graph optimization enables incremental dense mapping methods to adjust previous pose estimations and reduce the drift. Kerl et al. [19] register the frames through minimizing both photometric and geometric errors. For detecting loop closures, the candidates are selected from keyframes by metrically searching a sphere with a specified radius around the current keyframe. Then relative transformation is computed for each candidate. To validate a candidate, they propose to calculate inter-frame similarity by using an entropy based similarity metric. g2o is used to optimize the graph. Whelan et al. [36] extend the Kintinuous to a SLAM system with loop closure detection and graph optimization components. Despite using a dense technique in the odometry estimation component, the loop closure detection component employs a feature

based method. For the incoming frame, keypoint extraction is performed with SURF. Then loop closure candidates are searched in a bag of words database involving SURF descriptors of specified frames with a movement threshold. To determine a loop closure, different thresholds are used for; the count of keypoint correspondences, the number of inliers obtained with RANSAC, and the mean error of ICP registration respectively. More effective usage of the TSDF structure is proposed in [35] by locating the volume according to the sensor position dynamically. Moreover, graph subsampling is applied to increase processing speed.

For loop closure detection, several approaches are employed by RGB-D SLAM researchers such as registration with all keyframes [17], sampling from past frames [8, 24], metrical nearest neighbor search [19], and the bag of words approach [18, 35, 36]. The incremental mapping approaches [16, 26, 37], which do not apply loop closure detection, cannot fix previous alignment errors to avoid accumulating drift. Thus, these approaches are unsuited to map large environments. Registration with all previous keyframes leads to excessive computational cost since the number of keyframes increases rapidly and continuously. Searching loop closure candidates in a certain size of neighborhood can reduce the computational cost but it is an inconvenient approach for detecting large loop closures. Sampling may produce successful results for small-sized environments because the keyframe count is relatively low. However, as the environment grows, the chance of choosing correct candidates decreases. Therefore, sampling is not a suitable method for large-scale mapping. The bag of words approach [32] requires a visual vocabulary structure to be built through an offline pre-training phase [7, 11] or online incrementally [3, 27]. The vocabulary is composed of visual words generated from visual image features. Images are represented using visual word occurrences. Because of using a static vocabulary built through a pre-training phase, the offline methods are dependent on the training environment and produce increasing inaccurate results in different environments. The online approaches construct and update the vocabulary incrementally for the mapping environment during the navigation. However, this process is costly and less appropriate for real-time large-scale operation.

From the registration aspect, the dense approaches are able to perform more precise odometry estimation than the feature based methods but they require high computational power for real-time operation. Thus, most of them (such as [16, 26, 35–37]) are implemented on GPU to perform the pixel-wise operations in parallel. On the other hand, feature based methods provide better motion estimation between the frames that are not sequential, as in loop closures. Therefore, the feature based technique is more suitable for large-scale mapping since it reduces computational requirements considerably and performs better for computing the motion between loop closing frames.

With the goal of being able to map large-scale environments in real-time, we extend a feature based RGB-D SLAM system [9], which applies random sampling for loop closure detection. We designed a method for feature based SLAM systems to find loop closure candidates in real-time. Unlike the loop closure detection approaches described above, we propose to utilize global and local image features together to obtain the most likely candidates. An initial group of candidates are selected by an efficient filtering mechanism using global image descriptors. Then the local image features extracted for odometry estimation are utilized to determine the final candidates from the initial group. In the RGB-D SLAM system [9], we implemented our method instead of random sampling approach for selecting

loop closure candidates. In the remainder of the paper, the method is described in detail and the experimental results are presented.

3 Method

In this section, the graph based approach to SLAM and the RGB-D SLAM system [9] are described firstly. Then the proposed extension, a visual place recognition method for selecting loop closure candidates in feature based SLAM, is presented.

3.1 Graph Based SLAM

The graph based formulation has become a common solution to the SLAM problem recently. Poses of the robot are held in the nodes of the graph while spatial constraints between connected poses are represented by the edges [13]. The aim of the approach is to obtain the robot trajectory that contains optimal configuration of the poses producing highest consistency with the constraints. Basically, a graph based SLAM system is composed of two main components: front-end and back-end. The front-end interprets sensor measurements and constructs the pose graph by odometry estimation and loop closure detection functions. The back-end performs graph optimization to find a globally consistent robot trajectory.

The front-end applies odometry estimation by processing incoming frames from the sensor and calculating frame-to-frame transformations between sequential frames. The edges holding odometry constraints are inserted into the graph between the related consecutive poses. The factors such as sensor noise, scene lighting conditions, fast sensor movement, detecting low number of keypoints, etc. affect odometry estimation negatively. Therefore, the odometry errors cause the estimated robot poses to drift increasingly over time and erroneous construction of the map.

The trajectory drift can be reduced by detecting loop closures [35], which is the process of recognizing whether an already mapped region of the environment is visited again. After each loop closure detection, the pairwise transformation between the related loop closure frames is computed and the edge representing the loop closure constraint is added to the graph. In the adjustment phase, the back-end uses the loop closure constraints to optimize the pose graph and provides global consistency of the map. Place recognition is a key requirement for determining loop closure candidates. The place recognition module searches for potential loop closures in previous observations and provides the matches found as loop closure candidates. Then, a validity check is performed on the candidates to detect loop closures. Especially large-scale mapping requires effective place recognition to obtain the loop closure candidates fast and robustly.

3.2 RGB-D SLAM System

We extend the state-of-the-art RGB-D SLAM system developed by Endres et al. [9]. This graph based system utilizes sparse keypoints for estimating the robot motion between frames. The system detects keypoints and extracts keypoint descriptors from RGB frames and calculates 3D coordinates of the keypoints using the

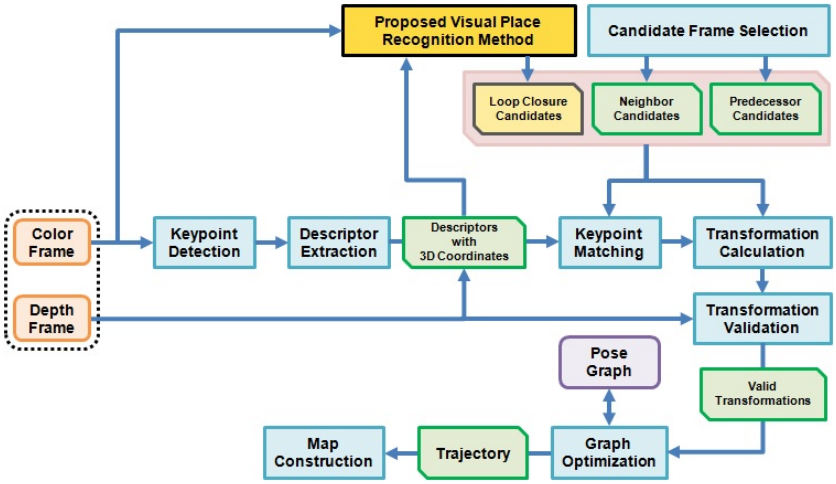


Fig. 1 Schematic overview of the RGB-D SLAM system [9] with integration of the proposed visual place recognition method

depth frames. The transformation between frames is computed by firstly finding 3D point correspondences with keypoint matching, and then applying RANSAC on the correspondences by using Mahalanobis distance to determine inliers. Then the transformation is verified with an environment measurement model, and the motion of the robot is estimated.

The front-end selects three groups of candidates from past frames to compute a transformation with each incoming frame:

- n direct predecessor frames of the current frame,
- k neighbor frames which are randomly selected from graph neighborhood of the previous frame,
- l randomly sampled frames from the keyframe set, which are used for loop closure detection. The keyframe set is a subset of observations containing full robot trajectory. The current frame is added to the keyframe set if it could not be successfully registered to the latest keyframe.

The graph is expanded by the front-end through adding a new node for each incoming frame and new edges for the valid transformations computed. Pruning is performed to eliminate the edges that have large error values and the graph is optimized by the back-end using the g2o framework [21]. Finally, map of the environment is generated by using the computed robot trajectory through projecting observations at each robot pose into a common coordinate frame.

The RGB-D SLAM system [9] randomly selects l loop closure candidates. Despite using a visual sensor, the system does not utilize any visual information to obtain candidates. Therefore, the chance of similarity between the current frame and arbitrarily chosen frames has important effect on the system accuracy. For small-scale environments, this situation does not cause problems because keyframe set is small and it is more likely to choose similar keyframes to the current frame. As the mapping environment expands and the robot navigates without frequent loop closures, the number of keyframes grows linearly and the probability of choosing similar keyframes decreases significantly. Thus, this situation causes unsteady

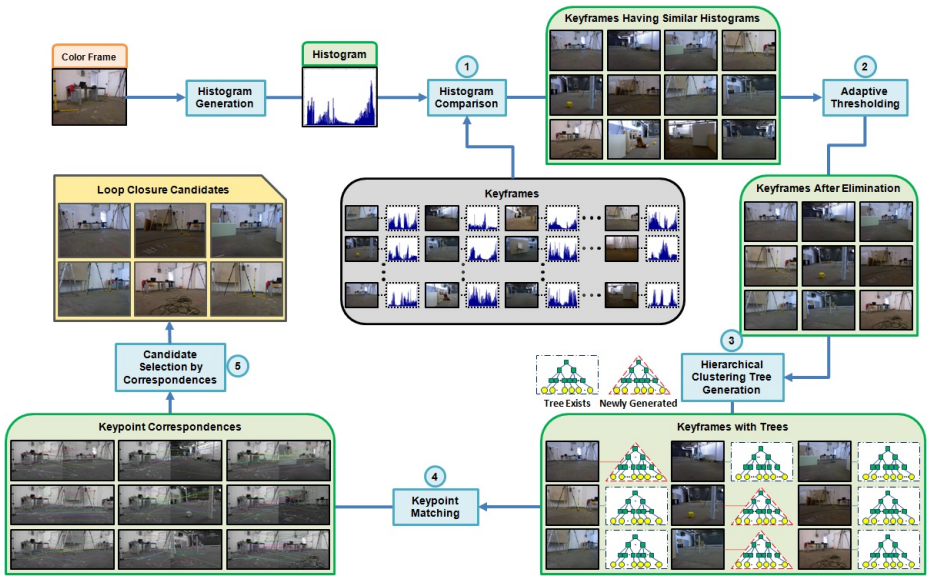


Fig. 2 Schematic overview of the proposed method. A group of similar keyframes are selected according to histogram comparison results (1). Then an adaptive thresholding mechanism is used to eliminate outliers (2). After generating the missing clustering trees (3), keypoint matching is performed between the incoming frame and each of the keyframes in the group (4). The loop closure candidates are selected according to the number of keypoint correspondences (5)

mapping results as the explored area grows. To overcome this problem, we integrate our place recognition method in this system to select loop closure candidates from keyframes. A schematic representation of the system with integration of our method is shown in Figure 1.

3.3 Proposed Visual Place Recognition Method for Feature Based SLAM Systems

We propose a visual place recognition method to obtain loop closure candidates in real-time. The proposed method is straightforward and easily applicable in feature based SLAM systems. The method determines loop closure candidates by using global image descriptors in conjunction with local image features. Image histogram is employed as a global descriptor to filter the search space with low memory and computational cost. Firstly, an initial group of candidates are retrieved according to histogram similarity. Then local features of the filtered candidates are utilized via keypoint correspondences to select loop closure candidates. In this phase, the method uses keypoint descriptors extracted for odometry estimation of the feature based SLAM system. Figure 2 shows a schematic representation of the proposed method.

We define four different techniques that are complementary parts of the method explained in the following sections. The first two techniques were used in our previous study [15]. The first technique, which employs only histogram comparison, has lower computational complexity and accuracy than the other techniques. More-

Table 1 Distance metrics used for comparing histograms

Distance Metric	**Formula**
Euclidean Distance	$\sqrt{\sum_{i=1}^{N}(H_i - H'_i)^2}$
Hellinger Distance	$\sqrt{1 - \frac{1}{\sqrt{\bar{H}\bar{H}'N^2}}\sum_{i=1}^{N}\sqrt{H_i H'_i}}$
Intersection Distance	$\sum_{i=1}^{N}\min(H_i, H'_i)$
Manhattan Distance	$\sum_{i=1}^{N}\lvert H_i - H'_i\rvert$

over, the other techniques employ keypoint matching approach and have better accuracy but with more computational complexity.

3.3.1 Histogram Only Place Recognition

The first technique depends on only image histograms to select loop closure candidates. Image histograms contain global intensity distribution of images. They represent pixel counts for each intensity level in the images. Histograms are easy to compute and insensitive to small viewpoint changes.

We use four distance metrics shown in Table 1 to compare histograms. As shown in the table, histograms to be compared are represented by H and H'. N indicates bin counts of the histograms. Hence, i-th bins of the histograms are denoted by H_i and H'_i respectively. The distance metrics have different characteristics. While large bins dominate the Euclidean distance, Hellinger handles smaller bin values more sensitively [4]. Intersection distance can consider partial overlaps and robustly handle occlusion [34]. Manhattan distance is a popular metric for comparing histograms with its simple characteristic and efficiency.

Algorithm 1: Candidate selection using histogram similarity

Input : current frame C
set K containing previous keyframes
Output: loop closure candidates
Parameters: *distance_metric*, *candidate_count*
$H_c \leftarrow$ generate histogram of the current frame C
for *each keyframe* $F_k \in K$ **do**
  $S_k \leftarrow$ similarity between histogram H_{F_k} of the keyframe and H_c in terms of *distance_metric*
  $L \leftarrow$ add the keyframe F_k to the list according to S_k in similarity order
end
return the first *candidate_count* keyframes from L

In the first step of the proposed method, two types of histograms are generated for each new color frame from the sensor: a three-channel RGB color histogram

and a single-channel grayscale histogram (after converting color image to grayscale format). The generated histograms are stored for long term to avoid recalculation of previous histograms for each incoming frame. The method subsequently compares histograms of all previous keyframes with the current frame by using one of the distance metrics in Table 1. The similar keyframes by color/gray level intensity distribution are acquired by this way. Then *candidate_count* keyframes having the most similar histograms to the current frame are chosen as loop closure candidates (see Algorithm 1). Histogram generation and comparison are fast operations. Thus, this process has negligible effect on frame processing speed.

3.3.2 Brute-Force Keypoint Matching Based Place Recognition

The global information obtained from histograms may cause mismatches due to similarity in color/gray levels of unrelated keyframes. Therefore, local image features are utilized via keypoints in keyframe comparison. Local features represent information about the regions around distinctive points (keypoints) in the images. Keypoint descriptors contain spatial information of the scene. Hence, they can tolerate large viewpoint changes.

We use keypoint descriptors to describe the scene and perform brute-force keypoint matching for recognizing places, where distances for all pairs of keypoints in the compared frames are calculated. In keypoint matching, the ratio approach proposed by Lowe [22] is applied to increase matching accuracy.

$$\frac{distance\ to\ the\ closest\ keypoint}{distance\ to\ the\ second\ closest\ keypoint} > ratio\_threshold \tag{1}$$

To accept a match of a keypoint in the source frame with a keypoint in the target as reliable, the ratio between distances of the nearest and the second nearest keypoint descriptors in the target frame must exceed a threshold. Considering that a keypoint in the source frame can be matched to one keypoint in the target, the nearest keypoint in the target must be much closer than the second nearest for a correct match. Because of the high dimensional feature space, ambiguous matches probably produce similar distances to the nearest and the second nearest target keypoints.

Keypoint matching is computationally expensive compared with histogram comparison. Instead of applying keypoint matching to every previous keyframe, the most similar $grouping\_factor \times candidate\_count$ keyframes are selected as an initial group by using histogram comparison. Then keypoint matching is applied between the current frame and each keyframe in the initial group. The keyframes are sorted by the number of keypoint matches with the current frame. The *candidate_count* keyframes producing the largest number of keypoint correspondences are selected as loop closure candidates (see Algorithm 2).

In a feature based SLAM system such as RGB-D SLAM [9], detection of keypoints and extraction of descriptors are already performed for each new frame in the odometry estimation phase. Thus, the existing descriptors are used in this step. The extra cost of this step comes from only matching of keypoints.

Algorithm 2: Candidate selection with brute-force keypoint matching

```
Input  : current frame C
         the initial group G of keyframes obtained by histogram comparison
         (size of G = grouping_factor × candidate_count)
Output: loop closure candidates
Parameters: ratio_threshold, candidate_count
K ← keypoint descriptors of the current frame C
for each keyframe G_f ∈ G do
    D_{G_f} ← keypoint descriptors of the keyframe G_f
    for each keypoint descriptor k ∈ K do
        for each keypoint descriptor d ∈ D_{G_f} do
            H_{k-d} ← calculate Hamming distance between k and d
        end
        c ← distance to the closest keypoint in G_f
        sc ← distance to the second closest keypoint in G_f
        if c/sc > ratio_threshold then
            M_{G_f} ← add the match to the accepted match set for G_f
        end
    end
    L ← add the keyframe G_f to the list according to size of M_{G_f} in descending order
end
return the first candidate_count keyframes from L
```

3.3.3 Hierarchical Clustering Based Place Recognition

Brute-force keypoint matching is computationally expensive especially when the frames are rich in keypoints. As another alternative, we use the matching algorithm proposed by Muja and Lowe [25], which decomposes the search space hierarchically, to perform keypoint matching more efficiently. The points are clustered successively to build a *hierarchical clustering tree*, in which the cluster centers are represented by the non-leaf nodes and the points are hold in the leaf nodes. The number of clusters is determined by the *branching_factor* parameter and the cluster centers are randomly selected among the points. Another parameter of the algorithm is *maximum_leaf_size*, which specifies the maximum count of points that can be contained in a cluster.

Algorithm 3: Building hierarchical clustering tree

```
Input  : set D containing all points
Output: hierarchical clustering tree
Parameters: branching_factor, maximum_leaf_size
if size of D < maximum_leaf_size then
    create leaf node with the points in D
else
    R ← choose branching_factor points as cluster centers from D randomly
    C ← cluster the points in D around the closest centers
    for each cluster C_i ∈ C do
        create non-leaf node with the center R_i
        recursively apply the algorithm to the points in C_i
    end
end
return hierarchical clustering tree
```

Algorithm 4: Searching hierarchical clustering tree

```
Input  : hierarchical clustering tree T, query point Q
Output: K approximate nearest neighbors of the query point
Parameters: maximum number of points to check N_max
N ← 0 {holds count of the checked points}
PQ ← empty priority queue for holding unexplored nodes
S ← empty priority queue for holding neighbors found
call TraverseTree(T, PQ, S)
while PQ is not empty and N < N_max do
    F ← top of PQ
    call TraverseTree(F, PQ, S)
end
return K top points from S
Procedure TraverseTree(F, PQ, S)
    if node F is a leaf node then
        search all the points in F and add to S
        N ← N + |F|
    else
        C ← child nodes of F
        C_q ← the closest node of C to the query point Q
        C_p ← C − C_q
        add all nodes in C_p to PQ
        call TraverseTree(C_q, PQ, S)
    end
end
```

To build the hierarchical clustering tree (see Algorithm 3), the algorithm begins with all points, chooses cluster centers randomly, creates non-leaf nodes for these centers, and assigns each point to the closest cluster. For each cluster having less number of points than *maximum_leaf_size*, a leaf node is created with the points in that cluster. The other clusters are divided again by the same procedure. This operation is recursively applied until each cluster contains less than *maximum_leaf_size* points. After building the tree in this way, the matching is performed by recursively traversing the tree (see Algorithm 4). To find a match for a query point, the closest node is selected in each step of the tree traversal. After obtaining the leaf node, the points in that cluster are checked for the match. A priority queue is used to hold unexplored nodes. The search can be expanded by picking the nearest node from the queue and continuing the traversal. The algorithm allows constructing multiple trees and searching them in parallel. Using multiple trees can increase search precision. However, it brings more computational cost due to increasing build time and memory requirements.

Algorithm 5 shows candidate selection by using the hierarchical clustering technique. In this technique, a single hierarchical clustering tree is used for each keyframe. First of all, as in the second technique, an initial group of candidates are selected by using histogram similarity. Then all keyframes in the initial group are checked for existence of the clustering tree. If a keyframe does not have its own tree, a clustering tree is built for that keyframe using its previously detected keypoints. Once a tree is constructed for a keyframe, it can be used for the later matching operations. Therefore, tree construction is performed only once for each keyframe in the whole mapping process. After building the clustering trees, keypoint matching is applied between the current frame and each of the keyframes in the initial group by considering the keypoints of the current frame as query points.

Algorithm 5: Candidate selection with hierarchical clustering

```
Input  : current frame C
         the initial group G of keyframes obtained by histogram comparison
         (size of G = grouping_factor × candidate_count)
Output: loop closure candidates
Parameters: ratio_threshold, candidate_count
for each keyframe G_f ∈ G do
    if G_f does not have its clustering tree then
        D_{G_f} ← keypoint descriptors of the keyframe G_f
        T_{G_f} ← build clustering tree for the keyframe G_f using D_{G_f}
    end
end
K ← keypoint descriptors of the current frame C
for each keyframe G_f ∈ G do
    for each keypoint descriptor k ∈ K do
        search T_{G_f} and find the two nearest neighbors of k by using Hamming distance
        c ← distance to the nearest keypoint in G_f
        sc ← distance to the second nearest keypoint in G_f
        if c/sc > ratio_threshold then
            M_{G_f} ← add the match to the accepted match set for G_f
        end
    end
    L ← add the keyframe G_f to the list according to size of M_{G_f} in descending order
end
return the first candidate_count keyframes from L
```

As in the brute-force keypoint matching technique, the distance ratio approach of Lowe [22] is used to eliminate false matches. According to the matching results, the *candidate_count* keyframes having the greatest number of correspondences are determined as loop closure candidates.

3.3.4 Adaptive Thresholding Based Place Recognition

In the second and the third techniques, the initial group of candidates are selected as the first *grouping_factor* × *candidate_count* keyframes according to histogram similarity scores. In some cases, this initial group may contain dissimilar frames to the current frame. Instead of selecting fixed number of candidates for the initial group, the candidates similar enough to the current frame can be selected. In this way, more outliers can be eliminated from the initial group before applying keypoint matching. To achieve this, each candidate frame in the group is checked for its similarity to the current frame and eliminated if it is less similar than a threshold. The threshold value is dynamically computed by multiplying the similarity score of the most similar candidate in the group by a factor. More precisely, the similarity threshold can be expressed as *best_score_in_the_group* × *thresholding_factor*. As explained in Section 4.5, a different *thresholding_factor* parameter is calculated for each similarity metric by analysing the general distribution of similarity scores for all frames. In this way, value of the similarity threshold is dynamically determined in each loop closure candidate search and it changes adaptively for each incoming frame from the sensor. Therefore, we call this outlier elimination technique as adaptive thresholding (see Figure 3).

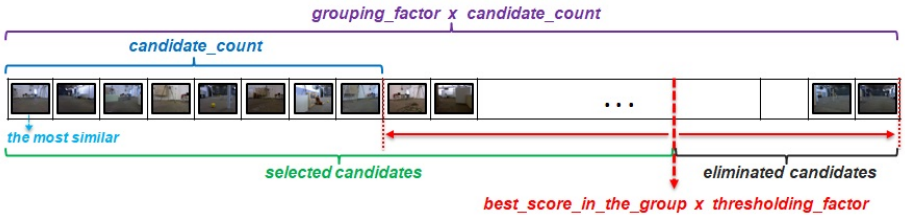


Fig. 3 Adaptive thresholding mechanism with an example loop closure candidate search

After eliminating outliers with adaptive thresholding, the selected initial group is passed to the keypoint matching step. Due to its accuracy and computational efficiency, we use the hierarchical clustering approach for this step. After matching keypoints, the *candidate_count* keyframes are chosen as loop closure candidates according to the number of keypoint correspondences.

We integrate our proposed method into the RGB-D SLAM system [9] to determine l loop closure candidates (see Figure 1). The next section presents the results of applying the proposed method on this SLAM system through the defined techniques.

4 Experimental Results

We carried out extensive experiments to evaluate the system with the proposed place recognition techniques. We assess contribution of each technique in terms of accuracy and computational efficiency by analysing the quantitative results of the experiments.

4.1 Experimental Setup

To evaluate the extended system with our proposed method, we use the *TUM RGB-D benchmark* [33] dataset which is widely used to evaluate SLAM systems. There are a variety of sequences in the dataset which comprise RGB and depth frames acquired by an RGB-D sensor along with the ground truth sensor trajectory. We employ *fr2* data sequences recorded in a large industrial hall. The sequences involve challenges such as motion blur, illumination changes, repetitive structures, less number of distinctive keypoints, and partially missing depth information due to the sensor range limit. The *fr2* sequences are suitable for evaluating the system performance in large and challenging indoor environments. Detailed information about the sequences are shown in Table 2. We use the *Absolute Trajectory Error* (ATE) metric provided by the benchmark to analyse the performance. The ATE is a global consistency metric that measures the translational drift between the estimated trajectory and the ground truth by computing absolute distances between related poses of the trajectories. The root mean square of the ATE (RMS-ATE) is computed as proposed by the benchmark to obtain the trajectory drift. It is a widely used metric for SLAM evaluation.

For place recognition and odometry estimation, we employ CenSurE keypoint detector [1] with FREAK descriptor [2]. CenSurE+FREAK performed best in

Table 2 Detailed information about the data sequences used in the experiments. The last column represents average number of keypoints per frame with CenSurE keypoint detector

Sequence	**Duration (s)**	**Average Translational Velocity (m/s)**	**Average Angular Velocity (deg/s)**	**Average Keypoint Count per Frame**
fr2/360_hemisphere	91.48	0.16	20.57	290.2
fr2/coke	84.55	0.14	9.43	248.2
fr2/desk	99.36	0.19	6.34	580.2
fr2/dishes	100.55	0.15	9.67	165.9
fr2/flowerbouquet	99.40	0.11	8.46	323.0
fr2/flowerbouquet_br.	76.89	0.16	10.60	235.4
fr2/large_no_loop	112.37	0.24	15.09	318.1
fr2/large_with_loop	173.19	0.23	17.21	309.6
fr2/metallic_sphere	75.60	0.15	10.42	214.3
fr2/metallic_sphere2	62.33	0.19	12.95	207.2
fr2/pioneer_360	72.75	0.23	12.05	249.5
fr2/pioneer_slam	155.72	0.26	13.38	279.8
fr2/pioneer_slam2	115.63	0.19	12.21	331.7
fr2/pioneer_slam3	111.91	0.16	12.34	289.1

Table 3 Parameter values used in the experiments

Parameter	**Value**
Transformation Parameters	
Maximum RANSAC iterations	250
Maximum keypoints	700
ratio_threshold	0.8
Place Recognition Parameters	
Histogram bin count	32
branching_factor	32
maximum_leaf_size	100
grouping_factor	4
Candidate Parameters	
Predecessor candidates (n)	8
Neighbor candidates (k)	20
Loop closure candidates (l)	8

terms of accuracy and speed in our previous work [14], in which we evaluated effects of various keypoint detectors and descriptors on RGB-D SLAM performance. In the experiments, we limit the maximum count of keypoints extracted from an image as 700. For matching keypoints, Hamming distance is used to measure distances between keypoint descriptors since FREAK is a binary descriptor. *ratio_threshold* is set to 0.8 for the distance ratio approach of Lowe [22]. For transformation estimation, the number of RANSAC iterations is limited to 250 at maximum. In loop closure detection, image histograms are computed using 32 bins for each channel. The hierarchical clustering trees are constructed by applying *branching_factor* as 32 and *maximum_leaf_size* as 100 (see Table 3).

The sizes of the candidate groups are determined as $n = 8$ predecessor candidates, $k = 20$ neighbor candidates, and $l = 8$ loop closure candidates. The initial group of candidates are determined by applying the *grouping_factor* parameter as 4. Thus, in the second and the third place recognition techniques, $4 \times 8 = 32$ keyframes are selected as the initial group by histogram comparison. The keypoint

matching step enables to reduce this number to *candidate_count* = 8 keyframes as loop closure candidates.

The parameter values are chosen according to the results of different experiments performed by using various parameter combinations. We are not able to give results of all experiments because of space limitations. A desktop PC running Ubuntu 12.04 with Intel Core i7-2600 CPU at 3.40GHz and 8GB of RAM is used to carry out the experiments. Each experiment is performed 5 times and average result for the 5 runs is presented as the final result. The following sections present the experimental results for each proposed place recognition technique.

Table 4 Accuracy (RMS-ATE in meters) results for histogram similarity technique and comparison with RGB-D SLAM [9]. (Euclidean=Euc, Hellinger=Hel, Intersection=Int, Manhattan=Man)

Histogram Type	**RGB**				**GRAYSCALE**				**RGB-D SLAM**
Distance Metric	**Euc.**	**Hel.**	**Int.**	**Man.**	**Euc.**	**Hel.**	**Int.**	**Man.**	
fr2/360_hemisphere	0.622	0.626	0.623	0.650	0.622	0.777	0.604	0.610	**0.592**
fr2/coke	0.203	**0.186**	0.204	0.192	0.224	0.198	0.192	0.195	0.205
fr2/desk	0.089	**0.087**	**0.087**	**0.087**	0.089	0.089	0.089	0.089	0.090
fr2/dishes	**0.784**	1.158	2.956	0.809	1.790	1.325	2.408	2.813	1.126
fr2/flowerbouquet	0.133	0.137	**0.120**	0.141	0.142	0.151	0.133	0.139	0.131
fr2/flowerbouquet_br.	0.733	0.732	0.739	0.744	0.753	0.753	0.720	0.751	**0.703**
fr2/large_no_loop	0.916	**0.450**	0.954	0.891	0.670	0.737	0.885	0.877	0.887
fr2/large_with_loop	0.408	0.400	**0.360**	0.408	0.540	0.456	0.406	0.388	3.598
fr2/metallic_sphere	1.817	2.051	1.818	2.441	2.012	1.903	1.805	1.875	**1.099**
fr2/metallic_sphere2	0.783	**0.702**	0.866	0.790	0.815	0.887	0.785	0.768	0.772
fr2/pioneer_360	0.209	0.218	0.204	0.207	0.204	**0.198**	0.219	0.212	0.213
fr2/pioneer_slam	0.387	**0.339**	0.342	0.351	0.348	0.354	0.390	0.357	0.367
fr2/pioneer_slam2	0.385	0.404	0.401	0.396	0.690	1.085	0.645	0.788	**0.381**
fr2/pioneer_slam3	0.581	0.584	0.407	0.375	0.832	0.324	**0.323**	0.330	0.511
Average	**0.575**	0.577	0.720	0.606	0.695	0.660	0.686	0.728	0.762

4.2 Histogram Only Experiments

We firstly assess selecting loop closure candidates by using only histogram similarity. The most similar 8 keyframes according to histogram similarity are selected as loop closure candidates. Table 4 contains accuracy results (RMS-ATE) for using this technique and makes comparison with RGB-D SLAM [9]. Final accuracy result, which is average of RMS-ATE obtained in 5 runs, is presented for each sequence. Average values for the columns are shown in the last row. According to the accuracy results, the lowest average drift is observed with RGB histogram and Euclidean distance (RGB+Euclidean) combination with 57.5 cm. The second best average result is produced by RGB+Hellinger with 57.7 cm RMS-ATE. Usage of RGB histogram performs better than grayscale in general, except for Intersection distance. For all combinations, the overall average error (see the last row) is decreased by employing histogram similarity technique to select loop closure candidates, comparing with random loop closure candidate selection approach of RGB-D SLAM [9]. Grayscale+Manhattan combination produces the maximum average drift with 72.8 cm RMS-ATE, that is still lower than 76.2 cm average drift

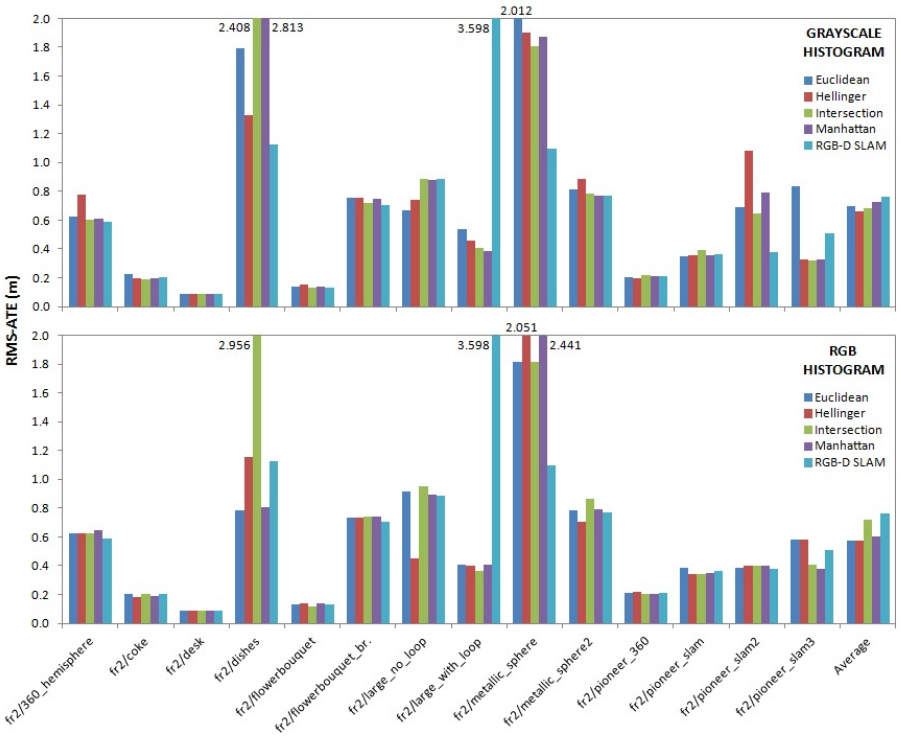


Fig. 4 Accuracy (RMS-ATE in meters) comparison between RGB-D SLAM [9] and the histogram similarity technique

of RGB-D SLAM [9]. Figure 4 shows the same data of Table 4 as a bar graph. As it can be seen from this graph, the most successful combination (RGB+Euclidean) is not better than RGB-D SLAM [9] in every data sequence, especially for small maps. The difference between RGB+Euclidean combination and RGB-D SLAM [9] in average error is mostly caused by *fr2/large_with_loop* sequence. This shows the effectiveness of the proposed approach in large maps even without using local image features.

Runtime results for the histogram similarity technique (see Table 8) show that average time spent for processing a frame is 172.8 ms for RGB histogram combinations and 172.2 ms for grayscale combinations. In comparison with 171.3 ms average processing time of RGB-D SLAM [9], we see that the histogram similarity technique brings negligible computational cost since generating and comparing histograms are fast processes. Even for some sequences such as *fr2/coke* and *fr2/metallic_sphere*, it is observed that processing time decreases with this technique compared with RGB-D SLAM [9]. The reason behind this situation is optimization time fluctuation.

Table 5 Accuracy (RMS-ATE in meters) results for brute-force keypoint matching technique and comparison with RGB-D SLAM [9]

Histogram Type	**RGB**				**GRAYSCALE**				**RGB-D SLAM**
Distance Metric	**Euc.**	**Hel.**	**Int.**	**Man.**	**Euc.**	**Hel.**	**Int.**	**Man.**	
fr2/360_hemisphere	0.642	0.654	0.591	0.591	0.614	**0.580**	0.645	0.591	0.592
fr2/coke	0.203	0.193	**0.191**	0.207	0.218	0.195	0.207	0.198	0.205
fr2/desk	**0.090**	**0.090**	**0.090**	**0.090**	**0.090**	**0.090**	**0.090**	**0.090**	**0.090**
fr2/dishes	**0.719**	0.785	1.766	0.825	0.846	0.802	0.795	0.809	1.126
fr2/flowerbouquet	0.126	0.138	0.127	0.129	0.130	**0.122**	0.138	0.126	0.131
fr2/flowerbouquet_br.	0.664	0.639	0.641	0.638	**0.637**	0.663	0.652	0.652	0.703
fr2/large_no_loop	1.085	**0.396**	0.602	0.494	0.564	0.773	0.640	0.857	0.887
fr2/large_with_loop	0.391	0.379	0.379	0.403	0.377	0.382	0.378	**0.370**	3.598
fr2/metallic_sphere	1.593	1.627	0.547	0.702	1.849	1.469	0.570	**0.533**	1.099
fr2/metallic_sphere2	0.726	0.769	0.898	0.747	0.791	0.743	0.722	**0.698**	0.772
fr2/pioneer_360	0.206	0.208	0.213	0.205	0.215	0.204	0.212	**0.199**	0.213
fr2/pioneer_slam	0.337	0.375	0.363	0.335	0.332	**0.329**	0.343	0.337	0.367
fr2/pioneer_slam2	0.398	0.387	0.391	0.386	0.406	0.406	0.396	0.391	**0.381**
fr2/pioneer_slam3	0.753	0.372	0.614	0.379	0.736	**0.322**	0.331	0.339	0.511
Average	0.567	0.501	0.529	0.438	0.557	0.506	**0.437**	0.442	0.762

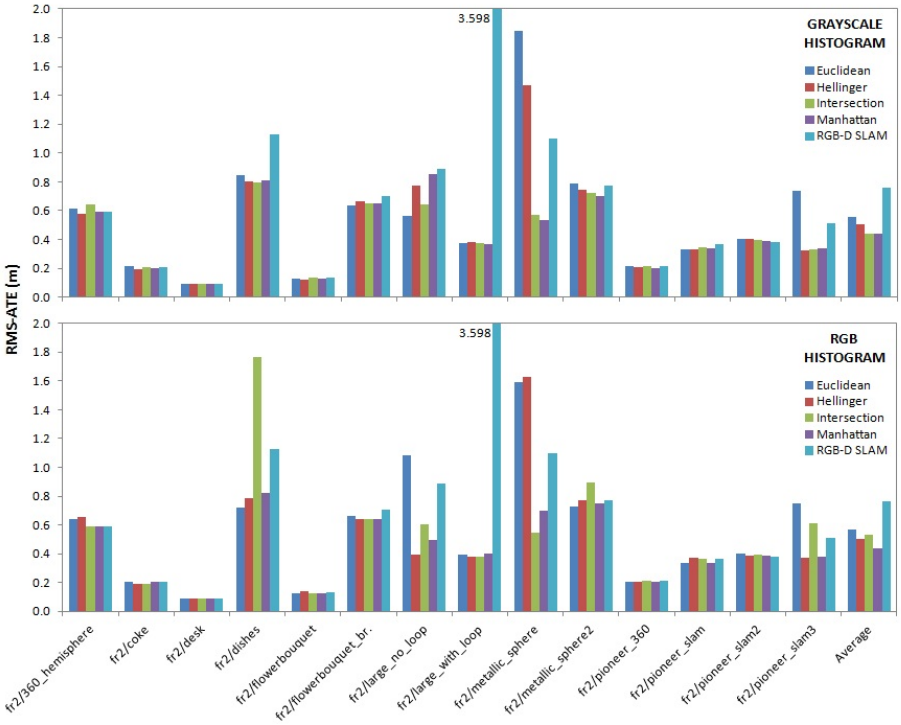


Fig. 5 Accuracy (RMS-ATE in meters) comparison between RGB-D SLAM [9] and the brute-force keypoint matching technique

4.3 Brute-Force Keypoint Matching Experiments

In the second type of experiments, loop closure candidates are selected by using brute-force keypoint matching after histogram comparison. Brute-force keypoint

matching is applied on 4×8 keyframes selected by using histogram similarity and then 8 loop closure candidates are determined according to the number of keypoint correspondences. The accuracy results are presented in Table 5. For all combinations, overall average drift error is reduced by performing brute-force keypoint matching after histogram comparison. The most successful result in average is 43.7 cm drift observed with Grayscale+Intersection combination, which is nearly 43% reduction of average drift comparing with RGB-D SLAM [9]. Additionally, RGB+Manhattan and Grayscale+Manhattan combinations have close performances to the best. Considering the results in the previous section, it can be seen that Intersection and Manhattan metrics perform better in finding more likely candidates as a larger initial group, but poorer in choosing the most relevant candidates near the top. Keypoint matching enables to select better candidates from the initial group. Figure 5 represents the same data of Table 5 as a graph. Comparing with histogram similarity technique, the major effect on accuracy improvement belongs to *fr2/dishes* and *fr2/metallic_sphere* sequences. Overall, the results demonstrate that employing local features via keypoint matching increases robustness.

The runtime results in Table 8 demonstrate that brute-force keypoint matching increases processing time around 33% in average. This extra cost is introduced by the matching process because keypoint detection and descriptor extraction are already performed for each incoming frame in the odometry estimation step as explained in Section 3.3.2. The increase in the processing time for *fr2/desk* sequence is much larger than the other sequences because the scenes in this sequence contain more keypoints than the others (see Table 2). Therefore, the brute-force approach needs to perform more comparisons for matching keypoints.

Table 6 Accuracy (RMS-ATE in meters) results for hierarchical clustering based technique and comparison with RGB-D SLAM [9]

Histogram Type	**RGB**				**GRAYSCALE**				**RGB-D SLAM**
Distance Metric	**Euc.**	**Hel.**	**Int.**	**Man.**	**Euc.**	**Hel.**	**Int.**	**Man.**	
fr2/360_hemisphere	0.604	0.630	0.700	**0.566**	0.599	0.657	0.621	0.625	0.592
fr2/coke	0.200	0.200	0.208	0.213	0.202	0.194	**0.183**	0.184	0.205
fr2/desk	**0.090**	**0.090**	**0.090**	**0.090**	**0.090**	**0.090**	**0.090**	**0.090**	**0.090**
fr2/dishes	0.837	0.809	1.845	**0.731**	0.832	0.799	0.806	0.761	1.126
fr2/flowerbouquet	0.138	0.132	0.137	0.146	0.141	0.145	0.141	0.135	**0.131**
fr2/flowerbouquet_br.	0.669	0.655	0.641	**0.636**	0.661	0.642	0.654	0.660	0.703
fr2/large_no_loop	0.826	0.522	0.744	0.530	0.664	0.879	**0.332**	0.653	0.887
fr2/large_with_loop	0.391	0.380	0.380	0.397	0.399	**0.369**	0.381	0.379	3.598
fr2/metallic_sphere	1.586	1.810	0.602	**0.546**	1.486	1.298	0.599	0.585	1.099
fr2/metallic_sphere2	**0.712**	0.774	0.762	0.720	0.856	0.789	0.802	0.730	0.772
fr2/pioneer_360	0.210	0.215	0.210	0.211	**0.204**	0.205	0.205	0.205	0.213
fr2/pioneer_slam	**0.332**	0.343	0.352	0.343	0.355	0.349	0.333	0.349	0.367
fr2/pioneer_slam2	0.392	0.391	0.386	0.411	0.385	**0.379**	0.400	0.401	0.381
fr2/pioneer_slam3	0.485	0.369	0.514	0.360	0.501	0.337	**0.331**	0.335	0.511
Average	0.534	0.523	0.541	0.421	0.527	0.509	**0.420**	0.435	0.762

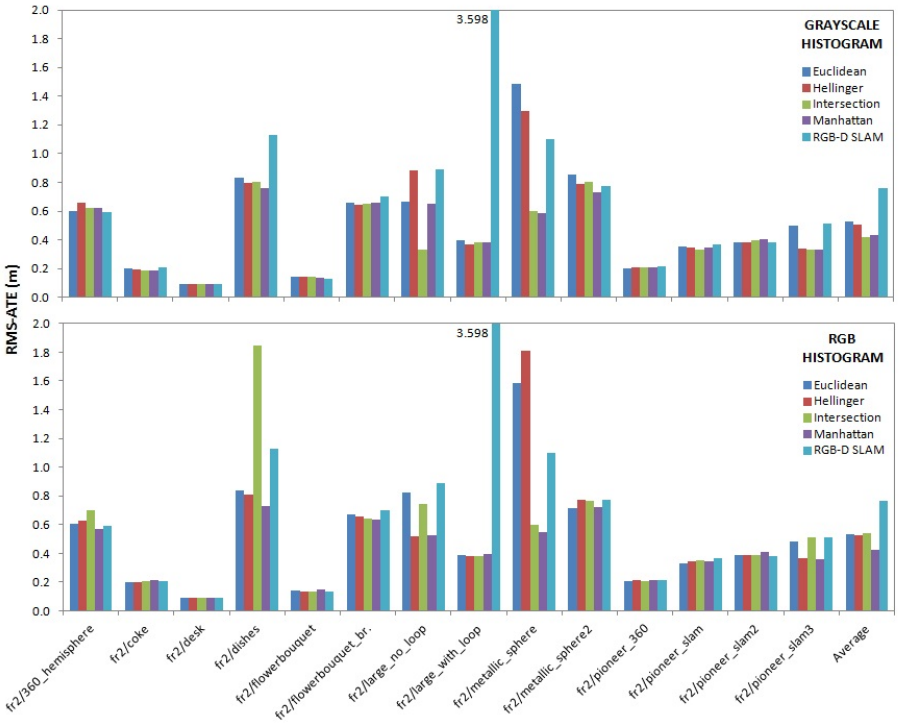
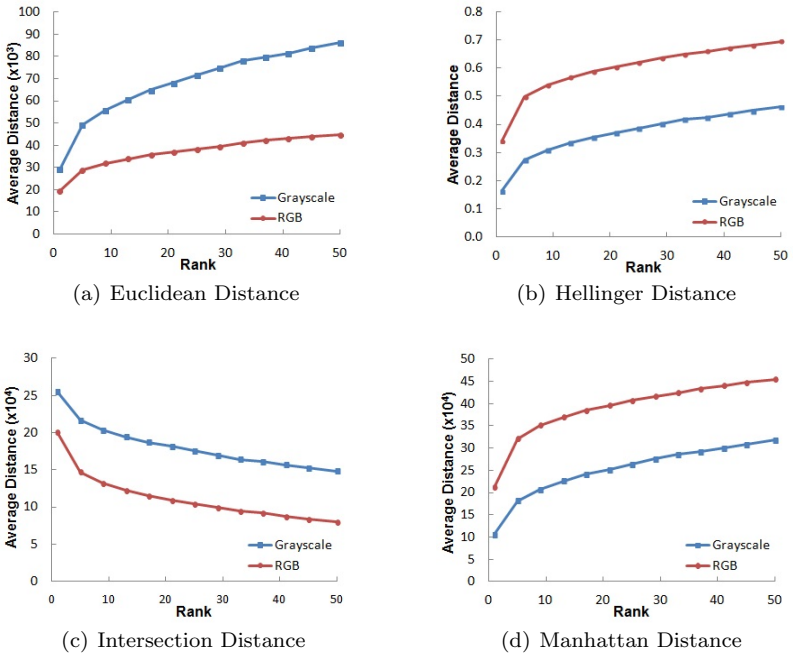


Fig. 6 Accuracy (RMS-ATE in meters) comparison between RGB-D SLAM [9] and the hierarchical clustering based technique

4.4 Hierarchical Clustering Experiments

In the third group of experiments, we utilize hierarchical clustering trees as explained in Section 3.3.3 to perform keypoint matching between the current frame and each of 4×8 keyframes chosen by histogram comparison. Table 6 presents RMS-ATE results obtained by using this technique. Similar to the brute-force matching results, Grayscale+Intersection combination is the best option again with a slightly lower average drift of 42 cm, which is approximately 45% accuracy improvement comparing with RGB-D SLAM [9]. RGB+Manhattan and Grayscale +Manhattan combinations produce close results to the best with 42.1 cm and 43.5 cm average drift errors respectively. Generally, usage of grayscale or RGB histograms performs closely for all metrics except Intersection, which produces ∼12 cm lower average error with grayscale histogram. Applying hierarchical clustering based keypoint matching as a second step after histogram comparison provides a notable drift reduction similarly to the brute-force approach. Hierarchical clustering and brute-force approaches perform similarly in terms of accuracy, which shows that using a single clustering tree provides enough search precision for our method. This effect can be observed in Figure 6 and Figure 5.

The main effect of employing hierarchical clustering instead of brute-force approach is observed in the processing time results. When we analyse the runtime results in Table 8, we see that the extra computational cost introduced by hierar-



(a) Euclidean Distance (b) Hellinger Distance

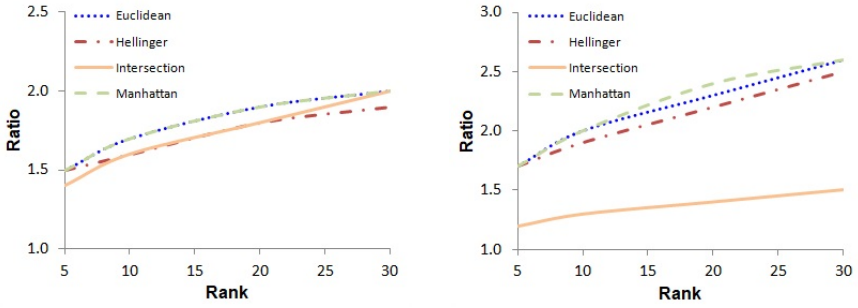
(c) Intersection Distance (d) Manhattan Distance

Fig. 7 Average distance distribution on all sequences according to the rank of the candidate. Unlike the other metrics, higher distance value means more similarity for Intersection metric

chical clustering based keypoint matching is about 12% in average. Considering the average cost of 33% for brute-force matching, it is obvious that keypoint matching step is performed much faster by employing hierarchical clustering. Besides the efficiency of the tree based search algorithm, building a tree for each keyframe only once and using this tree in every matching operation are important factors on this speedup. The speed improvement for *fr2/desk* sequence is more evident since it is much richer in keypoints than the other sequences (see Table 2).

4.5 Adaptive Thresholding Experiments

In the final group of experiments, we activate the adaptive thresholding module to eliminate outlier candidates before the hierarchical clustering based keypoint matching step, as explained in Section 3.3.4. Firstly, we perform an analysis to determine the value of the *thresholding_factor* parameter. We analyse the histogram similarity results using each distance metric on all sequences. Figure 7 shows variation of the average distance values according to the candidate rank for each metric. We observe a rapid decrease in similarity between the most similar and the 5^{th} similar candidate and then the similarity falls slowly. We calculate the ratio of histogram similarity between the first rank and the other specified ranks. According to the ratio values presented in Figure 8, it can be seen that close results are obtained for all distance metrics. The exception is Inter-



(a) RGB Histogram (b) Grayscale Histogram

Fig. 8 Histogram similarity ratio variation between the most similar candidate and other candidates in specified positions

section metric with grayscale histogram because of different characteristic property of the metric. Considering that 4×8 keyframes are selected as the initial group in the histogram comparison step, we use the similarity ratio between 1^{st} and 30^{th} ranks to determine a threshold. Therefore, using the values in Figure 8, the *thresholding_factor* parameter is applied as 2.0 for RGB histogram usage and 2.5 for grayscale histogram for all metrics except Intersection. For Intersection metric with grayscale histogram, the threshold is chosen as 1.5. Thus, for RGB histogram usage, the candidates having less similar histograms to the current frame than *best_score_in_the_group* $\times$ 2 are eliminated. Similarly, the frames having less similarity than *best_score_in_the_group* $\times$ 2.5 are excluded for grayscale histogram with the metrics other than Intersection. The elimination is performed as *best_score_in_the_group* $\times$ 1.5 for Grayscale+Intersection usage. For Intersection metric, the *thresholding_factor* is applied with inversion due to the characteristic of the metric as mentioned.

We test our overall method as shown in Figure 2. After histogram comparison, 4×8 keyframes having the most similar histograms to the current frame are selected as the initial group. Then adaptive thresholding is applied on the initial group for outlier elimination. After applying hierarchical clustering based keypoint matching, 8 keyframes are determined as loop closure candidates according to the number of keypoint correspondences. The accuracy results in Table 7 shows that Grayscale+Intersection is the most successful combination again with the average drift error of 44.1 cm. Grayscale+Manhattan and RGB+Intersection combinations produce the second and the third lowest average drifts as 44.3 cm and 45.3 cm respectively. Grayscale or RGB histograms generally produce close results (see Figure 9). In comparison with the histogram similarity technique (see Table 4 for results), adaptive thresholding in conjunction with hierarchical clustering provides extra drift reduction about 30% ($\sim$21 cm) for grayscale combinations and 20% ($\sim$13 cm) for RGB histogram usage in average. Furthermore, the proposed method increases accuracy of RGB-D SLAM [9] around 42%. This is a slightly lower accuracy improvement than hierarchical clustering technique ($\sim$45%). However, the decrease in computational cost makes it the most effective option as explained later.

Table 7 Accuracy (RMS-ATE in meters) results for adaptive thresholding technique and comparison with RGB-D SLAM [9]

Histogram Type	**RGB**				**GRAYSCALE**				**RGB-D SLAM**
Distance Metric	**Euc.**	**Hel.**	**Int.**	**Man.**	**Euc.**	**Hel.**	**Int.**	**Man.**	
fr2/360_hemisphere	0.746	0.623	**0.550**	0.627	0.682	0.601	0.569	0.577	0.592
fr2/coke	0.195	0.192	0.190	0.213	0.206	**0.185**	0.194	0.198	0.205
fr2/desk	0.089	0.089	0.090	0.089	**0.088**	0.089	0.090	0.089	0.090
fr2/dishes	0.829	0.781	0.807	0.765	0.808	**0.718**	0.797	0.880	1.126
fr2/flowerbouquet	0.151	0.142	**0.131**	0.133	0.140	0.137	0.137	0.144	**0.131**
fr2/flowerbouquet_br.	0.779	**0.680**	0.728	0.762	0.783	0.730	0.723	0.733	0.703
fr2/large_no_loop	0.919	0.501	0.724	**0.348**	0.604	0.510	0.395	0.430	0.887
fr2/large_with_loop	0.409	0.389	0.375	0.364	0.385	**0.348**	0.367	0.381	3.598
fr2/metallic_sphere	1.392	1.662	0.765	0.967	1.614	1.661	0.914	**0.512**	1.099
fr2/metallic_sphere2	0.765	0.810	0.696	0.716	0.755	0.859	**0.688**	0.781	0.772
fr2/pioneer_360	0.232	0.210	**0.202**	0.212	0.215	0.216	0.213	0.207	0.213
fr2/pioneer_slam	**0.319**	0.344	0.344	0.378	0.347	0.357	0.349	0.345	0.367
fr2/pioneer_slam2	**0.368**	0.388	0.386	0.374	0.391	0.392	0.400	0.390	0.381
fr2/pioneer_slam3	0.515	0.346	0.355	0.460	0.521	0.357	**0.341**	0.527	0.511
Average	0.551	0.511	0.453	0.458	0.539	0.511	**0.441**	0.443	0.762

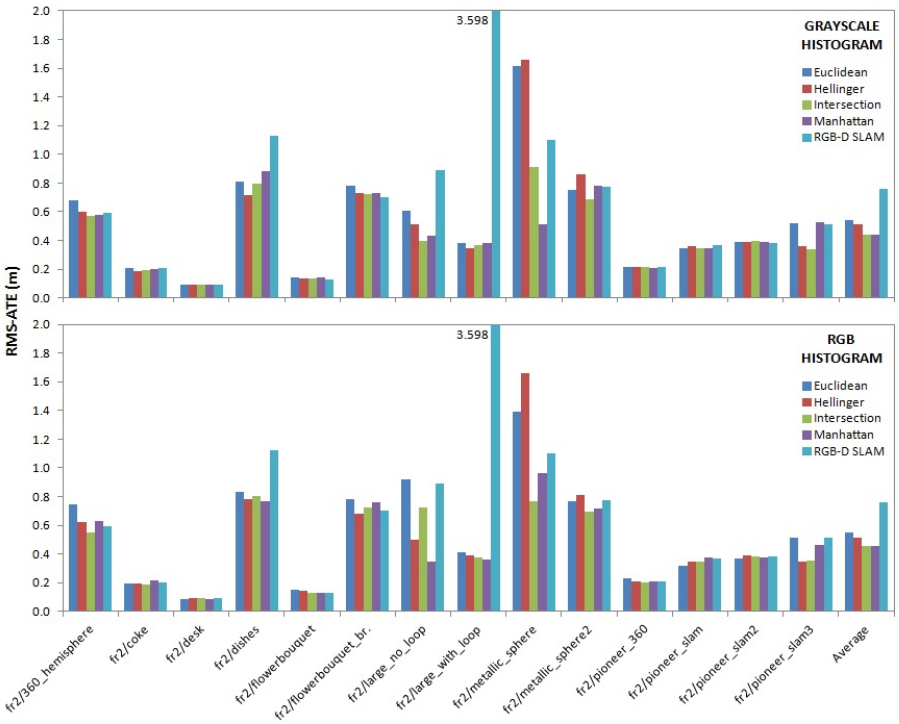


Fig. 9 Accuracy (RMS-ATE in meters) comparison between RGB-D SLAM [9] and the adaptive thresholding technique

The major benefit of adaptive thresholding comes in processing time. Eliminating outliers and processing less number of keyframes in keypoint matching reduce the frame processing time. Considering the results in Table 8, the hierarchical clus-

tering technique brings an extra cost around 25 ms per frame for RGB histogram and 19 ms per frame for grayscale comparing with RGB-D SLAM [9]. However, with adaptive thresholding, the extra costs are reduced to approximately 18 ms and 13 ms for RGB and grayscale histograms respectively. These results show that the extra computational cost of hierarchical clustering based keypoint matching is approximately decreased to around 30%.

Table 8 Comparison of computational performance (processing time per frame in milliseconds) between RGB-D SLAM [9] and the proposed techniques for all distance metrics. (The graph optimization time is considered)

Dataset	Histogram		Brute-Force Matching		Hierarchical Clustering		Adaptive Thresholding		RGB-D SLAM
	RGB	Gray.	RGB	Gray.	RGB	Gray.	RGB	Gray.	
fr2/360_hemisphere	156.6	152.5	199.7	188.9	184.0	175.8	178.4	170.1	154.1
fr2/coke	152.8	152.4	206.2	198.2	176.0	176.3	172.5	177.2	167.3
fr2/desk	329.7	369.2	590.6	596.6	426.1	399.8	410.1	396.0	348.6
fr2/dishes	142.9	142.1	148.1	145.0	140.4	143.7	138.5	141.0	135.4
fr2/flowerbouquet	205.3	212.5	291.5	279.7	235.6	229.3	226.8	219.4	202.6
fr2/flowerbouquet_br.	167.9	167.6	181.1	188.7	174.8	170.4	152.8	157.4	149.8
fr2/large_no_loop	146.3	144.8	221.5	213.9	172.3	163.9	168.5	159.7	143.9
fr2/large_with_loop	171.8	155.7	255.5	234.8	202.0	181.8	200.0	178.1	156.6
fr2/metallic_sphere	151.7	145.8	180.0	176.4	167.3	163.0	164.1	155.7	155.7
fr2/metallic_sphere2	135.2	121.5	159.7	156.2	148.9	145.2	140.9	135.8	136.4
fr2/pioneer_360	143.0	139.7	164.4	161.6	156.5	153.1	147.9	144.2	139.8
fr2/pioneer_slam	158.6	148.9	205.6	201.1	174.9	172.0	170.4	170.6	155.4
fr2/pioneer_slam2	202.0	204.2	245.0	243.8	225.0	218.3	216.9	213.0	198.9
fr2/pioneer_slam3	155.6	153.4	195.6	190.7	170.9	166.8	165.2	160.0	153.4
Average	172.8	172.2	231.8	226.8	196.8	190.0	189.5	184.2	171.3

4.6 Discussion

Figure 10 clearly emphasises the comparison between RGB-D SLAM [9] and the extended system with our most effective combination of Grayscale+Intersection in terms of accuracy. Additionally, Figure 11 compares minimum, maximum, and average RMS-ATE values obtained in 5 runs. The proposed method provides an improvement in almost all data sequences. For RGB-D SLAM [9], the average standard deviation of the drift error is 39.5 cm, while it is 7.8 cm for the extended system. This shows that the extended system performs more robustly than RGB-D SLAM [9]. The *fr2/large_with_loop* sequence shows the influence of our method most clearly since it contains a large loop in a long trajectory, where the RGB-D sensor returns back to the starting position at the end. This is the only loop in the sequence and is crucial to be detected for map consistency. It can be seen in the figures that our method works effectively and reduces the trajectory drift considerably. *fr2/large_no_loop* is also another sequence with a large trajectory but it contains no loop closures contrary to all the other sequences. Thus, avoiding false positive loop closures is more important for this sequence than the others.

The number of detected keypoints is critical for accurate trajectory estimation. *fr2/dishes*, *fr2/metallic_sphere*, and *fr2/metallic_sphere2* have lower average number of keypoints than the other sequences (see Table 2). Despite containing short

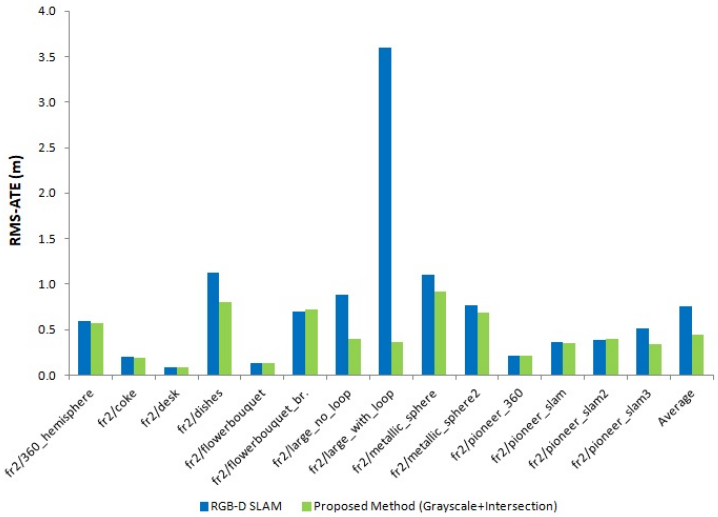
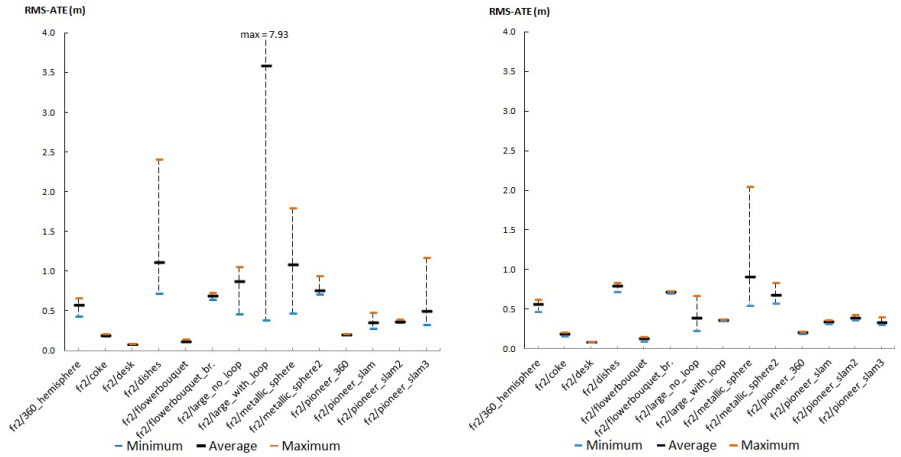


Fig. 10 Accuracy (RMS-ATE in meters) comparison between RGB-D SLAM [9] and the proposed extension (using the most effective combination of Grayscale+Intersection with adaptive thresholding)

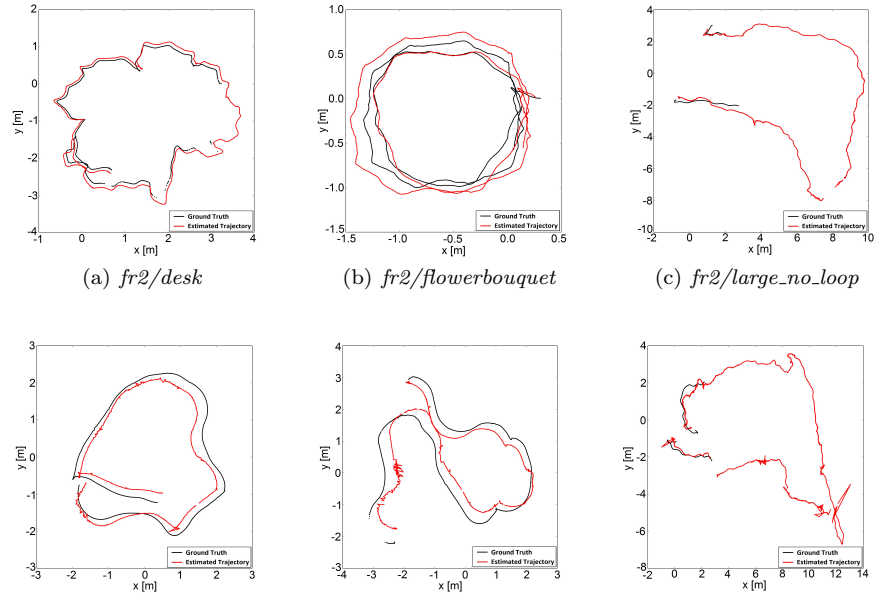


(a) RGB-D SLAM

(b) Grayscale+Intersection with Adaptive Thresholding

Fig. 11 Detailed accuracy results for the most effective combination of Grayscale+Intersection with adaptive thresholding and RGB-D SLAM [9]

trajectories, relatively high drift errors are observed for these sequences because the computed transformations have low quality. Therefore, effective loop closure detection becomes more important in this situation. *fr2/desk* is the sequence with the highest number of average detected keypoints per frame (see Table 2) and was recorded with relatively low angular velocity. Thus, the number of valid transformations to the earlier frames grows faster, and so does the number of edges in



(a) *fr2/desk* (b) *fr2/flowerbouquet* (c) *fr2/large_no_loop*

(d) *fr2/pioneer_360* (e) *fr2/pioneer_slam3* (f) *fr2/large_with_loop*

Fig. 12 2D projections of ground truth and estimated trajectory for sample sequences with the proposed extension (using the most effective combination of Grayscale+Intersection with adaptive thresholding)

the graph. This situation increases accuracy and decreases the processing speed. A similar effect is also observed for *fr2/flowerbouquet* sequence. While recording the *fr2/flowerbouquet_brownbackground* sequence, blocking the background with a brown carton prevents utilizing the keypoints in the background and increases the drift error. *fr2/pioneer_360*, *fr2/pioneer_slam*, *fr2/pioneer_slam2*, and *fr2/pioneer_slam3* sequences were recorded with stable movements of the RGB-D sensor along the horizontal plane since the sensor was mounted on a wheeled robot. This factor increases the reliability of transformation estimation and affects the accuracy positively.

Figure 12 presents estimated trajectories with the extended system (using the most effective combination of Grayscale+Intersection) and ground truth trajectories for the sequences *fr2/desk*, *fr2/flowerbouquet*, *fr2/large_no_loop*, *fr2/pioneer_360*, *fr2/pioneer_slam3*, and *fr2/large_with_loop*. It is clearly seen that close trajectories to ground truth are estimated. Particularly for *fr2/large_with_loop* sequence, the estimated trajectory demonstrates that the large loop is closed correctly (Figure 12(f)).

The map accuracy is not increased always by detection of more loop closures since the quality of calculated transformation is an important factor. The transformation quality may be affected adversely if the inlier count is low or detected keypoints do not spread on the frames. Thus, the accuracy may be reduced. We observe this situation in our experiments. For example, in comparison with RGB-D SLAM [9], the drift error is increased slightly by our most effective combination

Table 9 Comparison with other state-of-the-art approaches according to RMS-ATE in meters

Sequence	Extended System	RGB-D SLAM [9]	Whelan et. al [35]	DVO SLAM [19]	RGBiD-SLAM [16]
fr2/360_hemisphere	**0.569**	0.592	-	-	-
fr2/coke	**0.194**	0.205	-	-	-
fr2/desk	0.090	0.090	0.034	**0.017**	0.075
fr2/dishes	**0.797**	1.126	-	-	-
fr2/flowerbouquet	0.137	**0.131**	-	-	-
fr2/flowerbouquet_br.	0.723	**0.703**	-	-	-
fr2/large_no_loop	**0.395**	0.887	-	-	-
fr2/large_with_loop	**0.367**	3.598	-	-	-
fr2/metallic_sphere	**0.914**	1.099	-	-	-
fr2/metallic_sphere2	**0.688**	0.772	-	-	-
fr2/pioneer_360	**0.213**	**0.213**	-	-	-
fr2/pioneer_slam	**0.349**	0.367	-	-	-
fr2/pioneer_slam2	0.400	**0.381**	-	-	-
fr2/pioneer_slam3	**0.341**	0.511	-	-	-

Table 10 Comparison of ATE Median and ATE Max errors for the *fr2/large_no_loop* sequence with other state-of-the-art approaches. The results are given in meters

Method	Extended System	RGB-D SLAM [9]	Whelan et. al [35]
ATE Median	**0.19**	0.83	0.26
ATE Max	**0.41**	1.42	0.88

(Grayscale+Intersection) for *fr2/flowerbouquet*, *fr2/flowerbouquet_brownbackground*, and *fr2/pioneer_slam2* sequences (see Table 7). Thus, detecting accurate loop closures which produce high quality transformations with high number of inliers is the most crucial point. This becomes an important challenge as the map grows, since it gets more difficult to obtain the correct candidates in real-time. Our method aims to achieve this by firstly filtering the search space rapidly and then selecting the candidates according to the keypoint count for high quality loop closing transformations.

4.7 Comparative Evaluation

We compare the extended system with other state-of-the-art approaches in terms of trajectory accuracy (RMS-ATE) in Table 9. In the *TUM RGB-D benchmark* [33] dataset, the *fr1* data sequences are used by most SLAM systems since these sequences are rich in features and contain scenes of a small office. However, the *fr1* sequences are not convenient to evaluate large-scale mapping performance with loop closure detection. The *fr2* data sequences contain large trajectories, loop closures, and important challenges such as missing depth information, less features, and illumination changes. As it can be seen in the table, the *fr2* sequences are not generally preferred by researchers to evaluate the SLAM performance.

The extended system produces lower drift for the majority of the sequences. Especially it performs well for the *fr2/large_no_loop* and *fr2/large_with_loop* sequences, which contain the two largest trajectories in the dataset. Close results

are obtained for the sequences having shorter trajectories. *fr2/desk* can be categorized differently because this sequence is much richer in features than other sequences and was recorded with slower sensor movements. Thus, odometry estimation performance dominates the results for this sequence. In Table 10, ATE Median and ATE Max error results for *fr2/large_no_loop* are compared with the SLAM systems of Endres et. al [9] and Whelan et. al [35]. The extended system outperforms both SLAM systems in terms of both metrics.

5 Conclusion

In this paper, we presented an extension to a state-of-the-art RGB-D SLAM system [9] to enable real-time large-scale mapping. Our extension is a visual information based place recognition method to select loop closure candidates in feature based SLAM systems. The proposed method generates histograms of images and retrieves similar keyframes by histogram comparison using various distance metrics. As a second phase of histogram processing, a newly developed technique called adaptive thresholding eliminates outlier keyframes. Then an approximate matching approach based on searching hierarchical clustering trees is employed to match keypoints between the current frame and the keyframes. Finally, loop closure candidates are chosen according to the number of keypoint correspondences.

We integrated the proposed method into the RGB-D SLAM system [9] for detecting loop closures. We comprehensively evaluated the proposed method by performing various experiments on a commonly used dataset. While the dataset contains highly challenging conditions, the proposed visual place recognition method determines loop closure candidates robustly and reduces the trajectory drift substantially, especially for large sequences. The experimental results demonstrate that the extended system is able to map larger environments effectively. The best performing Grayscale histogram+Intersection distance combination with adaptive thresholding improves accuracy $\sim$42% comparing with RGB-D SLAM [9]. This improvement is obtained with approximately 13 ms extra processing time per frame, which can still enable real-time operation. To increase accuracy and speed more, we plan to study efficient search mechanisms based on location of previous keyframes in a future work. Utilizing location information may help to reduce number of keyframes to be processed and discover better candidates.

References

1. Agrawal, M., Konolige, K., Blas, M.: Censure: Center surround extremas for realtime feature detection and matching. In: Computer Vision–ECCV 2008, pp. 102–115. Springer (2008)
2. Alahi, A., Ortiz, R., Vandergheynst, P.: Freak: Fast retina keypoint. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pp. 510–517. IEEE (2012)
3. Angeli, A., Filliat, D., Doncieux, S., Meyer, J.A.: Fast and incremental method for loop-closure detection using bags of visual words. Robotics, IEEE Transactions on **24**(5), 1027–1037 (2008)
4. Arandjelović, R., Zisserman, A.: Three things everyone should know to improve object retrieval. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pp. 2911–2918. IEEE (2012)
5. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). Computer vision and image understanding **110**(3), 346–359 (2008)

6. Calonder, M., Lepetit, V., Fua, P.: Keypoint signatures for fast learning and recognition. In: Computer Vision–ECCV 2008, pp. 58–71. Springer (2008)
7. Cummins, M., Newman, P.: Appearance-only slam at large scale with fab-map 2.0. The International Journal of Robotics Research **30**(9), 1100–1123 (2011)
8. Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., Burgard, W.: An evaluation of the rgb-d slam system. In: Robotics and Automation (ICRA), 2012 IEEE International Conference on, pp. 1691–1696. IEEE (2012)
9. Endres, F., Hess, J., Sturm, J., Cremers, D., Burgard, W.: 3-d mapping with an rgb-d camera. Robotics, IEEE Transactions on **30**(1), 177–187 (2014)
10. Fischler, M., Bolles, R.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6), 381–395 (1981)
11. Gálvez-López, D., Tardos, J.D.: Bags of binary words for fast place recognition in image sequences. IEEE Transactions on Robotics **28**(5), 1188–1197 (2012)
12. Grisetti, G., Grzonka, S., Stachniss, C., Pfaff, P., Burgard, W.: Efficient estimation of accurate maximum likelihood maps in 3d. In: Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on, pp. 3472–3478. IEEE (2007)
13. Grisetti, G., Kümmerle, R., Stachniss, C., Burgard, W.: A tutorial on graph-based slam. Intelligent Transportation Systems Magazine, IEEE **2**(4), 31–43 (2010)
14. Guclu, O., Can, A.: A comparison of feature detectors and descriptors in rgb-d slam methods. In: Image Analysis and Recognition, pp. 297–305. Springer (2015)
15. Guclu, O., Can, A.: Histogram based visual place recognition for improving slam performance. In: Autonomous Robot Systems and Competitions (ICARSC), 2016 IEEE International Conference on. IEEE (2016)
16. Gutierrez-Gomez, D., Mayol-Cuevas, W., Guerrero, J.: Dense rgb-d visual odometry using inverse depth. Robotics and Autonomous Systems **75**, 571–583 (2016)
17. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In: In the 12th International Symposium on Experimental Robotics (ISER) (2010)
18. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. The International Journal of Robotics Research **31**(5), 647–663 (2012)
19. Kerl, C., Sturm, J., Cremers, D.: Dense visual slam for rgb-d cameras. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pp. 2100–2106. IEEE (2013)
20. Konolige, K.: Sparse sparse bundle adjustment. In: BMVC, pp. 1–11 (2010)
21. Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: g 2 o: A general framework for graph optimization. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp. 3607–3613. IEEE (2011)
22. Lowe, D.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60**(2), 91–110 (2004)
23. Lowry, S., Snderhauf, N., Newman, P., Leonard, J.J., Cox, D., Corke, P., Milford, M.J.: Visual place recognition: A survey. IEEE Transactions on Robotics **32**(1), 1–19 (2016)
24. Maier, R., Sturm, J., Cremers, D.: Submap-based bundle adjustment for 3d reconstruction from rgb-d data. In: Pattern Recognition, pp. 54–65. Springer (2014)
25. Muja, M., Lowe, D.G.: Fast matching of binary features. In: Computer and Robot Vision (CRV), 2012 Ninth Conference on, pp. 404–410. IEEE (2012)
26. Newcombe, R., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: Real-time dense surface mapping and tracking. In: Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on, pp. 127–136. IEEE (2011)
27. Nicosevici, T., Garcia, R.: Automatic visual bag-of-words for online robot navigation and mapping. IEEE Transactions on Robotics **28**(4), 886–898 (2012)
28. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: Computer Vision–ECCV 2006, pp. 430–443. Springer (2006)
29. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: an efficient alternative to sift or surf. In: Computer Vision (ICCV), 2011 IEEE International Conference on, pp. 2564–2571. IEEE (2011)
30. Scaramuzza, D., Fraundorfer, F.: Visual odometry, part i: The first 30 years and fundamentals. IEEE robotics & automation magazine **18**(4), 80–92 (2011)

31. Segal, A., Haehnel, D., Thrun, S.: Generalized-icp. In: Robotics: Science and Systems, vol. 2 (2009)
32. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on, pp. 1470–1477. IEEE (2003)
33. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of rgb-d slam systems. In: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pp. 573–580. IEEE (2012)
34. Swain, M.J., Ballard, D.H.: Color indexing. International journal of computer vision **7**(1), 11–32 (1991)
35. Whelan, T., Kaess, M., Johannsson, H., Fallon, M., Leonard, J., McDonald, J.: Real-time large-scale dense rgb-d slam with volumetric fusion. The International Journal of Robotics Research **34**(4-5), 598–626 (2015)
36. Whelan, T., Kaess, M., Leonard, J., McDonald, J.: Deformation-based loop closure for large scale dense rgb-d slam. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pp. 548–555. IEEE (2013)
37. Whelan, T., McDonald, J., Kaess, M., Fallon, M., Johannsson, H., Leonard, J.: Kintinuous: Spatially extended KinectFusion. In: RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras (2012)