Improved RGB-D Camera-based SLAM System for Mobil Robots

László Somlyai, Zoltán Vámossy

Óbuda University, John von Neumann Faculty of Informatics
Bécsi út 96/b, H-1034 Budapest, Hungary
somlyai.laszlo@nik.uni-obuda.hu; vamossy.zoltan@nik.uni-obuda.hu

Abstract: The paper presents an improved Simultaneous Localization and Mapping (iSLAM) and 3D reconstruction system for a mobile robot carrying an RGB-D camera. The developed system generates a three-dimensional point cloud from the color and depth camera data of the RGB-D camera. The matching is performed on successive point clouds that partially overlap. After feature detection on the color camera images, the method selects the 3D points in the successive point clouds that belong together. During the iterative multi-step matching algorithm based on Singular Value Decomposition (ISVD), it minimizes the matching error between the selected point clouds and deletes non-real point pairs in the process. Our previous SLAM method has been improved in several ways. On the one hand, a conditional averaging filter (Distance Image Filter: DIF) was created for the depth camera data to reduce the noise. The matching algorithm iteratively determines the matching transformation and the estimated displacement from the feature points of several recent point cloud segments. It defines a parameter for the accuracy and quality of each matching and includes the sub-results of pairs in the final displacement estimate by weighting these accuracy parameters. In this way, the algorithm yields significantly improved accuracy values, which in all cases are of comparable magnitude to those of methods in the literature, and for some published test sets exceed their characteristics. Since parallel programming methods are used to run the fits to previous states, the operation runtime remains fast. If the robot returns to its previous location, the improved loop closure detection method detects this fact, refines the estimates, and improves the accuracy. Finally, the proposed system also produces a 3D point cloud of the environment.

Keywords: mobile robot navigation; simultaneous localization and mapping; loop-closure detection; 3D reconstruction; RGB-D sensor

1 Introduction

Today, autonomous mobile robots are becoming more and more common and can be found in more areas of life. Depending on their task, they need to recognize their environment, detect obstacles, and determine their position. Research on robot

navigation uses different sensors. Machine vision is a common approach for sensing the environment. The mobile robot may not have a map of the environment beforehand, so it has to create one on-the-fly based on the information gathered from the area explored. When navigation involves both learning about the environment and determining one's position, the navigation procedure is called Simultaneous Localization and Mapping (SLAM) [1].

Indoor navigation does not allow the use of absolute positioning systems such as GPS, which significantly facilitates positioning over a larger area. However, building an absolute localization system indoors is possible, but this requires using other sensors. Recently, more and more sensors have become available for machine vision systems, such as the Xtion, PrimeSense, and Kinect sensors. Despite their simpler design, they have sufficient accuracy to detect the environment of a small robot. They can be used to create a detailed 3D map of the vehicle's environment. RGB-D and stereo cameras or LIDAR are primary sensors in many research projects [1]. This paper focuses on methods for indoor localization and presents an improved SLAM system for indoor use.

In the case of the SLAM system and 3D reconstruction procedure described in this article, no pre-installed external sensor system is required. The system determines the motion estimation using data from an RGB-D camera mounted on the mobile robot. In previous works [2, 3], we compared our in-house developed structured light-based sensor to the MS Kinect device investigating their accuracy. The here discussed method is an improved version of our previous SLAM system [4, 5], for which a dedicated framework has been developed [6]. The framework allows the algorithm to be tested on other public datasets in addition to the data recorded in our environment. In our tests, we used the datasets TUM [7] and POZNAN [8], which are commonly used in the literature. Our current system provides a more accurate estimation of the displacement, and its operation is more robust than the previous one due to the improved matching procedure and the real-time loop-closure algorithm. Using the TUM datasets the proposed method was compared with the work of Guclu et al. [9], Whelan et al. [10], Liu et al. [11], Endres et al. (RGB-D SLAM) [12], and Stückler et al. MRSMap system [13], and in more cases we obtained better values than similar previous works.

The paper is organized as follows: Section 2 presents similar approaches and the methods used therein concisely. Section 3 presents the architecture of our SLAM system, 3D point cloud indexing, feature point determination, and as a novelty, the algorithm of an improved position estimation procedure and an efficient method for loop-closure detection. Section 4 describes the evaluation of the implemented system, both on our test database and on two external databases that provide a comparison with similar systems. Finally, the conclusion summarizes the results of the development.

2 Related Work

Stand-alone SLAM systems, where there is no need to install external sensors, use different sensor types. The simpler two-dimensional LIDAR systems have lower power requirements and can process sensor data at high-speed. 2D SLAM algorithms usually use iterative Closest Point algorithm (ICP) [14] or point-to-line [15] methods for displacement estimation. However, more resource-intensive 3D LIDAR or machine vision systems can achieve higher accuracy and build a more detailed map [9, 10, 11, 12, 13, 16-20, 21], but these systems require higher computing power.

One of the first SLAM systems based purely on machine vision was developed by Henry et al. [22, 23]. The system was based on RGB-D camera data and feature point detection. An ICP-based algorithm was used to estimate the displacement and construct the pose graph. The high computational demand of the ICP algorithm was reduced to avoid the need to match the entire point clouds. A few pre-selected points and reduced point clouds are required to be matched. Instead of the pose graph optimization [24], the SBA method [25] was used to minimize the estimated path errors.

Another system uses a similar solution [26], where a drone is controlled utilizing an RGB-D camera. A FAST feature point detector is used, which is a less robust detector, i.e. less invariant to transformations. It has the advantage that the algorithm has a short processing time. It uses data from an Inertial Measurement Unit (IMU) sensor to fit the point clouds and estimate the initial displacement. Subsequently, several researchers [12, 27] have improved the accuracy of the algorithm by tracking feature points. The use of robust feature detectors can further improve the system.

The operation of each feature point detector may be different in various environments. One study compares three robust feature search methods [27]: Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), and (Oriented FAST and Rotated Brief) ORB feature detectors. Feature point detectors are used for both successive image matching and g2o optimization [24]. The octree-based volumetric representation method is used to display the map. Another research [13] saves the surfel map for position estimation and also stores a multi-resolution surfel map in octree for later 3D surface reconstruction. After g2o optimization, the final global surface map is obtained. Yuan et al. present an ICP-based method for reconstructing a three-dimensional map [28]. The method first searches for SURF feature points in two color images and then estimates the relative displacement between them in two steps. In the first step, mismatched feature point pairs are deleted based on the Euclidean distance of the features. Next, an initial displacement estimate is determined between successive measurements using the Random sample consensus (RANSAC) algorithm, and feature pairs that have been incorrectly detected are removed. Next, a modified ICP algorithm determines the final estimated displacement. The random selection of features was solved by

removing those too close by determining the Euclidean distance between them. Key images are selected to detect the previously investigated neighborhood, i.e., the loop-closure algorithm works. The key image selection is based on the condition that they are not too close to each other. Their results were compared with those of the SLAM system of Endres et al. [27], using SIFT, SURF, and ORB feature point detectors. Feature point search methods are investigated in many studies, but often the more computationally demanding ones are chosen due to their robustness. ORB is much faster and hardly increases the total error value, but still the SURF detector is chosen more often [11].

One of the critical problems with machine vision-based mapping and tracking is the large amount of data that needs to be processed and stored, mainly due to the size of each point cloud. Thus, some methods are only capable of small-scale reconstruction [29].

3 Methodology

3.1 The System Structure

The improved Simultaneous Localization and Mapping (iSLAM) system presented in this paper is an improved version of our previous system [4, 5]. A summary of its operation is shown in Figure 1. The system estimates sensor motion by frame-to-frame matching of color and depth images from an RGB-D camera. Most SLAM techniques can be divided into two main parts: a front-end and a back-end module. The creation of individual point clouds, storage, running of SURF feature detector, 3D feature point matcher, keypoint finder, frame-to-frame displacement estimation, loop-closure algorithm, and update of a closed loop in case of a successful hit are parts of the front-end module. The back-end module contains only the 3D reconstruction procedure; all new positions are estimated and corrected in the front-end part. The generation of the final 3D model is not performed online, but the continuous update of the path is done in real-time.

3.2 3D Point Cloud with Indexing

The system processes the RGB-D camera's color (I_{RGBn}) and depth (I_{Dn}) images in sequence. From the last depth data, it generates a 3D point cloud to which it assigns the colors in the color camera image, which becomes the 3D color point cloud (X_n). More previous data is needed for displacement estimation and for continuously running the loop-closure algorithm during each matched process. From the current RGB-D camera image, a 3D point cloud has to be generated by detecting feature

points in the color image (SURF detector), and finally, this data has to be stored for later use. Therefore, a three-dimensional point cloud is generated from the RGB-D camera's color and depth data. Each point in the point cloud is indexed by the pixel position of the color camera image for fast matching of the 3D feature points. Each point in the point cloud contains its three-dimensional spatial coordinate in the camera's local coordinate system and its corresponding color value.

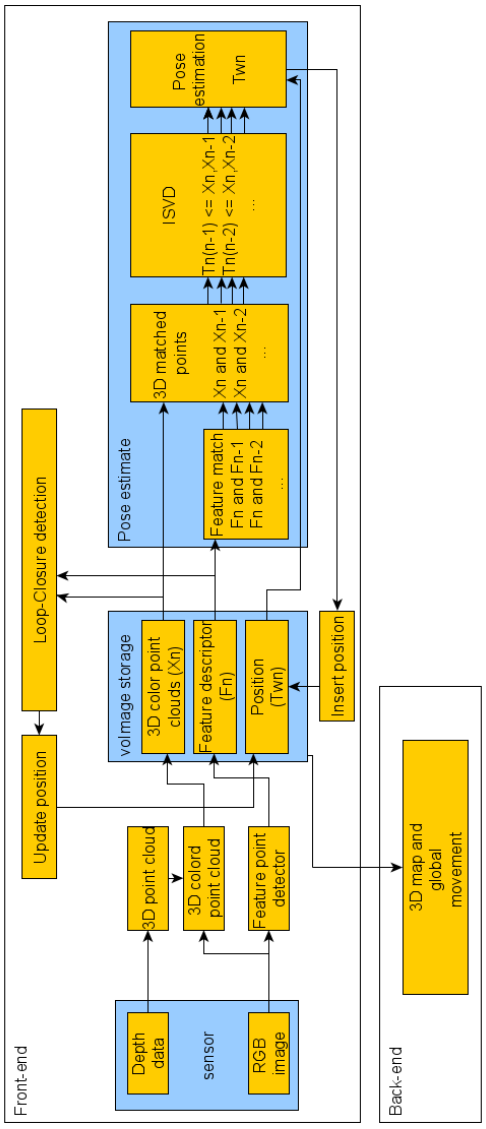

Figure 1
Essential connections in the iSLAM system

3.5. Distance Image Filter (DIF)

A conditional averaging filter was applied to the depth image to reduce the measurement noise in the depth image. Algorithm 1 is based on a 3×3 averaging filter. Still, it skips points where the distance measurement gave an erroneous result, i.e., it has a value of zero.

Input: w, h width and height of the depth image
Input: δ the input depth image
Output: δ' the filtered depth image
  for $i = 1 \rightarrow (w-1)$ **do**
    for $j = 1 \rightarrow (h-1)$ **do**
      $k = 0$
      for $m = i-1 \rightarrow i+1$ **do**
        for $n = j-1 \rightarrow j+1$ **do**
          if $\delta(m,n) \neq 0$ **then**
            $T_k = \delta(m,n)$
            $k = k+1$
      if $k \neq 0$ **then**
        $\delta'(i,j) = (\sum_{l=1}^{k} T_l)/k$

Algorithm 1
Depth image filter (DIF) [3]

3.3 Feature Descriptor

Estimating the displacement between successive images requires the partial overlap of the corresponding images and point clouds. From the common part of the point clouds, the system selects a few points present in both. The selection is made using the descriptors in the color camera image. For each new frame, a search for SURF feature points in the color camera image is performed once (F_n), and then the feature descriptor vector is stored in the *voImage* repository together with the previously created 3D point cloud. The threshold of the SURF detector is varied based on the sharpness of the images; if an image contains too few feature points, the threshold is set to a lower value.

The system continuously selects key frames from the new color camera images. The loop-closure algorithm continuously monitors the match between the new image and the keyframes using feature descriptors. If the sensor is in a similar position as before, the current estimated position can be refined. The selection of keyframes is based on two criteria. One keyframe is selected every 20 frames, such that the sharpest frame with the most feature points is selected from the last 20 frames. Feature descriptors are used to determine the sharpness.

Only a part of the *voImage* repository is permanently present in memory, such as feature descriptors and estimated position (*Tw*) data. Due to its large size, only the last *N* 3D colored point cloud frames are stored in the memory, where *N* is the parameter of the given configuration. Older data are temporarily archived in the background memory. The loop-closure algorithm tries to determine whether the current camera image matches an earlier key image in this section by examining the match between feature descriptors. If a match is found, the 3D point cloud is reloaded to determine the spatial position of the sensor in the local coordinate system for the feature descriptors that just match. Thus, data with high memory requirements only occupy memory for a short time, and the system can work with large amounts of data.

3.4 Improved Position Estimation Method

The position estimation of the new camera image is performed using the acquired data (*voImage* repository) by running the improved multi-step fitting procedure (Algorithm 1), which iteratively applies SVD. Our previous SLAM system used the ICP algorithm [4] to estimate the new position. However, this system had a more significant error than the one using the new ISVD procedure, and the running time of the ICP algorithm was much longer. The system always tries to fit the latest *MP* image to the previously estimated camera positions using the previously stored data ($X_{n\text{-}1}$, $F_{n\text{-}1}$, $X_{n\text{-}2}$, $F_{n\text{-}2}$,..., $X_{n\text{-}MP}$, $F_{n\text{-}MP}$), and finally, the new estimated position (T_{Wn}) is obtained by applying different weights. (The method calculated the weights at the final part of Algorithm 1. is described later.)

On system start-up, the first camera position will be the origin of the global coordinate system. A new camera image is only aligned with a specific number of immediately preceding camera images. In our system, a given image is matched to 4 previous images (*f* iteration); with a smaller number of matches, a less accurate map can be produced. In our experience, fitting to more previous images only slightly improves position estimation but significantly increases the processing time. For efficiency, the position estimation of a given image is done by parallel processing on multiple threads for the previous few images. The first step in matching the current frame to a previous frame is to select a reduced point cloud from the current and the previous point cloud. The function *match* (X_n, F_n, X_{n-1}, F_{n-1}) determines the color camera pixels that match in the two frames based on feature points and then assigns their spatial coordinates to the matching points from the previously generated indexed point cloud. Thus, two reduced point clouds are obtained, in which each point has its corresponding spatial pair (D_{n-f}, S_{n-f}). In the next step, the relative displacement of the camera between the two adjacent images is determined. The initial relative displacement $T_{n-f} = 0$ between the sets D_{n-f}, and S_{n-f} is recalculated and refined in several steps. At each iteration, the transformation (T_{n-f}) between the two point sets is determined with Singular Value Decomposition (SVD) and used to perform a trial test run. During the test fit, the system converts

the set S into the coordinate system of the set D using the resulting transformation ($S' = T_{n-f} * S$). Finally, it checks the accuracy of the fit and the Euclidean distance of adjacent point pairs and finally deletes any point pairs that generate a larger error than the prescribed error.

To determine the quality of the match (1), the system takes the average distance of the points after the test fit, and the reciprocal of this is used to give the weight (W_{n-f}) representing the quality of the fit, where $|D|$ denotes the number of pairs of points currently being tested, $\left(d_{i_x}, d_{i_y}, d_{i_z}\right) \in D$, $\left(s'_{i_x}, s'_{i_y}, s'_{i_z}\right) \in S'$ are the coordinates of the matched points:

$$W_{n-f} = \frac{1}{\frac{1}{|D|}\sum_{i=0}^{|D|}(|d_{i_x}-s'_{i_x}|+|d_{i_y}-s'_{i_y}|+|d_{i_z}-s'_{i_z}|)} \tag{1}$$

The higher this value, the closer the points are after the fit, so there are fewer pairs of faulty points in the reduced set, and the lower the noise of the point sets. The distance of the incorrectly detected point pairs will be larger than the others, so the point pairs whose distance is larger than a threshold value *th* are deleted in each step. The threshold value decreases over the iterations according to a power function (2), where j is the number of iterations and the α, β constants are set empirically (see later).

$$th = \alpha * j^{\beta} \tag{2}$$

Each matching estimation ends when the cycle reaches a maximum number of iterations (*maxIt*), or too few points remain after deleting non-matching point pairs (*minPoint*), or the threshold falls below a certain level. For the current image and a previous image, the relative fit is added to the spatial position of the previous image to obtain the global spatial position of the current image from the perspective of the previous few images. Using these values together with the previous weights, the spatial position (Tw_n) of the current image is obtained.

3.6 Loop-Closure Detection

As a last step, the front-end module uses the loop-closure algorithm to continuously monitor the matches between the current image and the keyframes. It determines the match by comparing the feature descriptors of the current image with the feature descriptors of the previous key images. If it finds a match with a previous image, it tries to determine a relative displacement between the key image and the current image using multi-step matching (Algorithm 2).

The system does not examine all current images; it only runs the loop-closure algorithm if the image has sufficient sharpness. The sharpness is determined based on the feature descriptors of the previous 20 images. If the number of feature descriptors in the current image is at least 1.2 times greater than the median of the

number of feature descriptors found in the previous images, then that image is less blurred and more detailed than the others. The system then checks if the position of the current image and the global position of the key image are close to each other, if the sensor is facing in the same direction, and if it is in the same position as before. If there is a large discrepancy, it will not examine the key image because either it is a false-detection case or there may be very little overlap between the key image and the current image. Thus, the running time of the algorithm is significantly reduced because a small number of key images have to be examined by multi-step matching, which has a non-negligible running time due to the search for feature pairs.

Input: MP the number of previous point clouds involved in the matching
Input: $X_n, X_{n-1}, ..., X_{n-MP}$ the $n.$ and the preceding colored point clouds
Input: $F_n, F_{n-1}, ..., F_{n-MP}$ set of feature points on $n.$ and the preceding images
Input: $maxIt$ maximum number of iterations
Input: $minError$ the minimum acceptable value of the error
Input: $minPoint$ minimum number of feature point pairs
Input: α, β constants defining the acceptable error
Output: T_n the weighted estimated displacement from each match
Output: W_{n-f} the weight of each match

for $f = 1 \rightarrow MP$ **do**
  $D_{n-f}, S_{n-f} = match(X_n, F_n, X_{n-f}, F_{n-f})$
  $T_{n-f} = 0$
  for $j = 1 \rightarrow maxIt$ **do**
    $T_{n-f} = SVD(D_{n-f}, S_{n-f}, T_{n-f})$
    $th = \alpha * (j)^{\beta}$

    $S' = T_{n-f} * S_{n-f}$
    $D = D_{n-f}$
    $e = 0;$
    ▷ $s'_i \in S'$ and $d_i \in D$ pairs of feature points to be tested
    for $i = 1 \rightarrow |D|$ **do** ▷ $|D|$ the number of points pairs in the match
      $e = e + |d_{i_x} - s'_{i_x}| + |d_{i_y} - s'_{i_y}| + |d_{i_z} - s'_{i_z}|$
    $W_{n-f} = 1/(e/|D|)$

    for $i = 1 \rightarrow |D|$ **do**
      if $\sqrt[2]{(d_{i_x} - s'_{i_x})^2 + (d_{i_y} - s'_{i_y})^2 + (d_{i_z} - s'_{i_z})^2} > th$ **then**
        $remove(d_i, s_i)$
    if $th < minError$ OR $|D| < minPoint$ **then**
      end iteration
$T = 0$
$w = 0$
for $f = 1 \rightarrow MP$ **do**
  $T = T + (Tw_{n-f} * T_{n-f}) * W_{n-f}$
  $w = w + W_{n-f}$
$Tw_n = T/w$

Algorithm 2
Multi-stage three-dimensional point cloud matcher (Improved ISVD)

3.7 3D Reconstruction

The back-end allows the reconstruction of a three-dimensional point cloud of the covered area. The system uses previous position estimations to merge the individual point clouds into a global cloud. Points that are too close are deleted. Finally, the point cloud can be exported to a .ply file for post-processing. The point cloud is post-processed using MeshLab. The size of the original point cloud must be reduced as a first step. After random sampling [30] from the surface points, the remaining surface point mesh points are triangulated according to the BPA [31] algorithm (triangle mesh).

4 Results and Evaluation

4.1 Datasets

The iSLAM system presented in this paper is compared with current similar SLAM procedures. In all cases, the tests were run on a computer with the following configuration: CPU: AMD Ryzen 5 5600X, RAM: 32GB. For the comparison, the algorithm parameter settings were identical for all datasets. The accuracy of the system was tested on two public benchmark datasets from different sources. One of the large datasets available online is from the Technical University of Munich recorded with several different Kinect sensors [7]. The tests were run on datasets belonging to the *fr1* and *fr2* groups. The University of Poznan dataset [8] was used as a second source. This was created and made available for mobile robot development. Several new datasets were added to the original four measurement lines, where the recording was repeated with Kinect 1 and 2 sensors. Both datasets contain real absolute spatial position values for each image taken at each position. (Although there are outliers for absolute-space location for some datasets).

4.2 Results

In similar studies, the Absolute Trajectory Error (ATE) measure is often used to determine the accuracy of the estimated displacement [32]. The ATE value shows the difference between the estimated trajectory and the actual position. In displacement estimation, the minor errors at each fit are accumulated, so if an error occurs at one fit, it will be reflected at others. The ATE values for the estimated and actual displacements were measured in meters. In the following tables (Table I-III), the sum of the ATE values Root Mean Square Error is compared with the results of similar works.

In the next section, the testing of the algorithm on TUM datasets is presented. Many researchers use these to test the accuracy of their systems, so we have purposely chosen these datasets for comparability.

In Table I, the ATE values are used to test the accuracy of the improved ISVD algorithm. We also tested the accuracy of the ISVD matching algorithm alone, with the loop-closure (LC) algorithm and with the DIF filter. When the algorithm was run, the value of the MP parameter (number of previous images to which the matching is performed) was 4 in all cases.

The results show that when only the improved ISVD algorithm was used to estimate the motion, the worst results were obtained for the examined data sets. When the LC algorithm was added, better results were obtained in all cases, either the data set contained larger or smaller closed loops. When the LC algorithm was added with the DIF filter, the accuracy was further improved in some cases. For the 21 data sets tested, the DIF filter gave more accurate results than when no filter was used in 12 cases. Using the DIF filter 59% of the time, the results were better for randomly selected data sets.

Table I
The Absolute Trajectory Error (ATE RMSE) values of the proprietary algorithm were tested on fr1, fr2 and fr3 datasets. All values are in meters. Bold numbers highlight the best results.

Data set	no LC, no DIF, MP = 4	LC + MP = 4	LC + DIF + MP = 4
fr1/rpy	0.0466	**0.0260**	0.0320
fr1/xyz	0.0232	**0.0140**	0.0148
fr1/360	0.1028	**0.0683**	0.0757
fr1/desk	0.0455	0.0300	**0.0298**
fr1/room	0.2727	0.1755	**0.1403**
fr1/desk2	0.0847	**0.0534**	0.0535
fr1/plant	0.0517	**0.0320**	0.0338
fr2/desk	0.3431	0.2170	**0.1039**
fr2 pioner 360	0.5301	0.4069	**0.3877**
fr2/pioner slam	0.5569	0.4421	**0.3571**
fr2/pioner slam 2	1.2000	**1.1754**	1.3110
fr2/pioner slam 3	0.4710	**0.4558**	0.9211
fr1 teddy	0.1414	0.1432	**0.1383**
fr3 long	0.4502	0.2267	**0.1100**
fr2/flowerbouquet	0.1688	0.1682	**0.1202**
fr2/metallic_sphere	0.7879	**0.7785**	0.8273
fr3 walking static	0.0391	0.0268	**0.0252**
fr3 walking xyz	0.2696	0.1184	**0.0900**
fr3 walking halfsphere	0.4286	**0.1044**	0.1063
fr3 sitting xyz	0.0716	0.0468	**0.0324**
fr3 sitting halfsphere	0.0716	0.0524	**0.0499**

For one comparison, we selected datasets from the TUM datasets where recordings were made in a static environment. These datasets were compared with the work of Guclu et al. [9], Whelan et al. [10], Liu et al. [11], Endres et al. (RGB-D SLAM) [12], and Stückler et al. MRSMap system [13], the results of which are summarized in Table II. In five cases, fr1/rpy, fr1/xyz, fr1/360, fr2/flowerbouquet and fr2/metallic sphere, we obtained better ATE values than the similar previous works. In these cases, we obtained slightly better results, and in the other cases, the ATE values we obtained were of the same order of magnitude as the tested ones.

Table II

The Absolute Trajectory Error (ATE RMSE) values of the proprietary algorithm were tested on fr1, fr2 and fr3 datasets, in static environment. All values are in meters. Bold numbers highlight the best results.

Data set	iSLAM (own)	Ext. RGBD SLAM [9]	Whelan et al. [10]	Qiang et al. [11]	RGBD SLAM [12]	MRSMap [13]
fr1/rpy	**0.025**		0.028			0.027
fr1/xyz	**0.013**	0.014	0.017	0.013	0.014	0.013
fr1/360	**0.056**	0.075				
fr1/desk	0.026	**0.022**	0.037	0.064	0.026	0.043
fr1/room	0.140		0.075		0.087	**0.069**
fr1/desk2	0.049	**0.034**	0.071			0.049
fr1/plant	0.034	0.068	0.047			**0.026**
fr2/xyz	0.015			0.015	**0.008**	
fr2/desk	0.120	0.090	**0.034**		0.057	0.052
fr2/ flowerbouquet	**0.120**	0.137			0.131	
fr2/metallic sphere	**0.779**	0.914			1.099	
fr2/pioner slam	0.360	**0.349**			0.367	
fr2/pioner slam 2	1.175	0.400			**0.381**	
fr2/pioner slam 3	0.4558	**0.3410**			0.511	
fr3 long	0.1100			**0.028**	0.032	

Next, we compared datasets from the TUM datasets where the recordings were made in a dynamic environment. These datasets were compared with Co-fusion [16], StaticFusion [17], Hachiuma [18], Liu [19], and Guo's work [20], the results of which are summarized in Table III. The results show that our method can handle this environment. For the selected datasets, in addition to the works used for comparison, our system did not give better results in this case, but in order of magnitude, our results are comparable to the systems under study.

Table III

The Absolute Trajectory Error (ATE RMSE) values of the proprietary algorithm were tested on fr1, fr2 and fr3 datasets, in dynamic environment. All values are in meters. Bold numbers highlight the best results.

Data set	iSLAM (own)	Co-fusion [16]	StaticFusion [17]	Hachiuma (DF) [18]	Liu (BS+DR) [19]	Guo (DUCK) [20]
fr3 walking static	0.0252	0.551	**0.014**	0.036	0.029	0.0235
fr3 walking xyz	0.0900	0.696	0.127	0.085	0.126	**0.0426**
fr3 walking halfsphere	0.1044	0.803	0.391	**0.072**	0.336	
fr3 sitting xyz	0.0324	0.027	0.040	0.052	0.045	**0.0203**
fr3 sitting halfsphere	0.0499	**0.036**	0.040	0.041	0.037	

Table IV shows the evolution of our algorithm. Our previous results [4], [5] on the TUM fr1/desk and the trajectory1 dataset from the Poznan one-frame were compared to the current, more accurate procedure. It can be seen that the 2018 version of fr1/desk [6] did not yet include the LC algorithm and is much less accurate than the current values, which already included the LC procedure. The most accurate result was obtained with the current version, where the improved SLAM method uses the sensor data with a conditional averaging filter, and closed loop (LC) detection is also continuously applied.

Table IV

The Absolut Trajectory Error (ATE RMSE) value of the own algorithm examined on the fr1/desk and Poznan/traj1 datasets, shows the development of our procedure. All values are given in meters.

Data set	2018 surf4 [5]	ISVD+LC surf-4 [6]	ISVD+LC surf-4	iSLAM (ISVD+LC+DIF surf-4)
fr1/desk	0.0907	0.0628	0.0300	0.0298
PUTK2/traj1, on 500 images	0.1952		0.2012	0.1961

The back-end part of the system provides the possibility to reconstruct the traversed area in 3D. Figure 2 shows the 3D reconstruction of the fr1/desk dataset.

The time to fit each new frame and run the loop-closure algorithm averaged 100 ms for smaller data sets and 150 ms for larger data sets over the processing of all images. The actual and estimated paths for the data series in Table II. are summarized in Figure 3. Estimated paths are in blue, real paths in black, and ATE values per point in red.

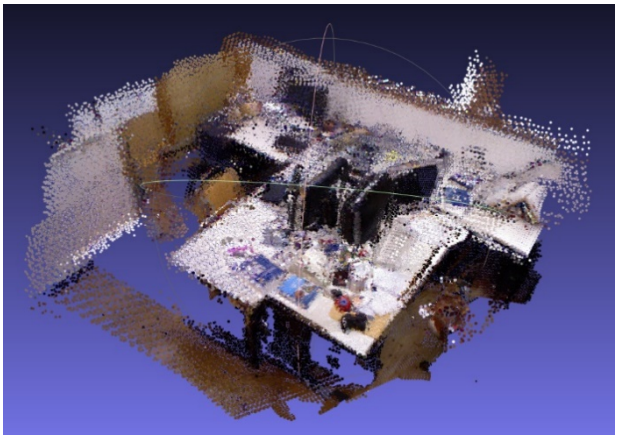

Figure 2
The 3D reconstruction of fr1/desk data

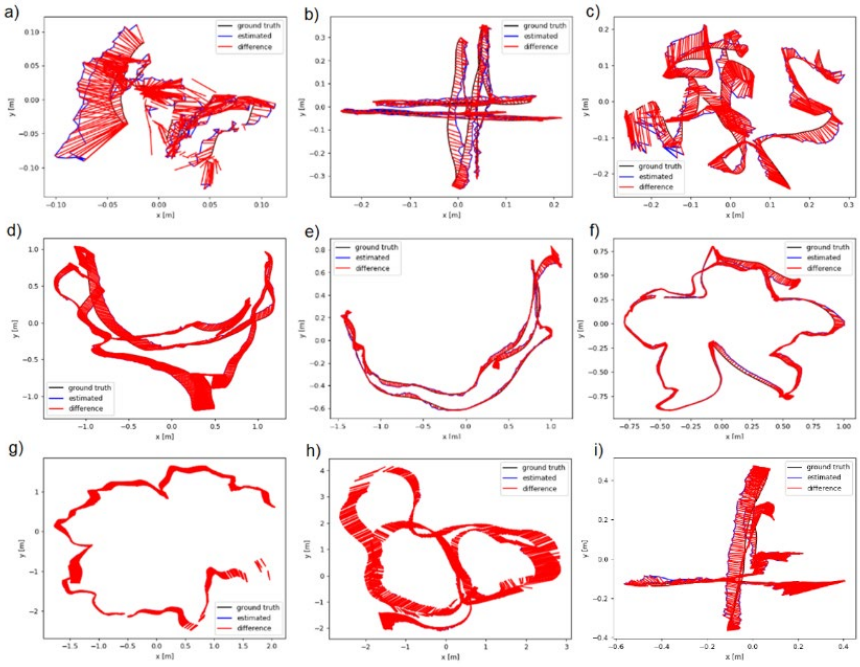

Figure 3
Running results of iSLAM (ISVD+LC+DIF) algorithms on a) fr1/rpy, b) fr1/xyz, c) fr1/360, d) fr1/room, e) fr1/desk2, f) fr1/plant, g) fr2/desk, h) fr2/pioner slam 2 and i) fr3/walking xyz datasets. The actual path is shown in black, the estimated path in blue, and the error in red.

Conclusions

In this paper, an improved SLAM (iSLAM) system is presented and compared with other similar systems. The system estimates the sensor motion by frame-to-frame matching color and depth images from an RGB-D camera and generates a 3D point cloud of the covered area. It uses a SURF feature detector to match new images to previous ones. By searching for feature point pairs, it determines matching points between the new frame and some previous frames and then converts these points into a 3D point cloud. During the matching process, a multi-step matching algorithm estimates the relative transformation between the new frame and some of the previous ones. The resulting multiple estimated displacements are taken into account with the weights obtained during the fitting process to determine the final displacement of the new frame. The system uses the loop-closure algorithm to mark key image frames continuously and then fits the new frames to them, reducing the cumulative error. To determine the quality of our method, we used two open databases to run the algorithm on 22 different datasets and compared the resulting ATE RMSE values with the results of seven other systems. In five cases, our system gave better results; in the other cases, the results differed only slightly from the best values of the other systems.

References

[1] D. Scaramuzza, F. Fraundorfer, Part I the first 30 years and fundamentals IEEE Robotics & Automation Magazine, Vol. 18, No. 4, pp. 80-92, 2011

[2] G. Csaba, L. Somlyai and Z. Vámossy, "Differences between Kinect and structured lighting sensor in robot navigation," 2012 IEEE 10th International Symposium on Applied Machine Intelligence and Informatics (SAMI), Herl'any, 2012, pp. 85-90, DOI: 10.1109/SAMI.2012.6208934

[3] L. Somlyai, Z. Vámossy, "Map Building with RGB-D Camera for Mobil Robot" IEEE 16th International Conference on Intelligent Engineering Systems 2012 (INES 2012), pp. 489-493, ISBN: 978-1-4673-2695-7, Lisbon, Portugal, 2012

[4] L. Somlyai, Z. Vámossy, SLAM algorithm for mobile robot localization with RGB-D camera, Fluids, Heat and Mass Transfer, Mechanical and Civil Engineering, ISBN: 978-1-61804-358-0, pp. 89-94

[5] L. Somlyai and Z. Vámossy (2022) "ISVD-Based Advanced Simultaneous Localization and Mapping (SLAM) Algorithm for Mobile Robots" Machines Vol. 10, No. 7: 519, https://doi.org/10.3390/machines10070519

[6] L. Somlyai, G. Csaba and Z. Vámossy, "Benchmark system for novel 3D SLAM algorithms," 2018 IEEE 16th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Kosice, 2018, pp. 131-136, DOI: 10.1109/SAMI.2018.8324000

[7] J. Sturm, N. Engelhard, F. Endres, W. Burgard, D. Cremers A benchmark for the evaluation of RGB-D SLAM systems IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 573-580, Vilamoura, 2012

[8] A. Schmidt, M. Fularz, M. Kraft, A. Kasinski, M. Nowicki An Indoor RGB-D Dataset for the Evaluation of Robot Navigation Algorithms Prof. of Int. Conf. on Advances Concepts for Intelligent Vision Systems, Lecture Notes in Computer Science Vol. 8192, pp. 321-329

[9] O. Guclu, A.B. Can. Fast and effective loop closure detection to improve SLAM performance J. Intell. Robot. Syst., pp. 1-23, 2017

[10] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. Leonard, and J. McDonald. Real-time large-scale dense RGB-D slam with volumetric fusion. The International Journal of Robotics Research, 34: pp. 598-626, 2015

[11] Q. Liu, R. Li, H. Hu and D. Gu, "Building semantic maps for blind people to navigate at home," 2016 8th Computer Science and Electronic Engineering (CEEC), Colchester, 2016, pp. 12-17, DOI: 10.1109/CEEC.2016.7835881

[12] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3-D mapping with an RGB-D camera. IEEE Transactions on Robotics, 30(1), 2014

[13] J. Stückler and S. Behnke. Multi-resolution surfel maps ¨ for efficient dense 3D modeling and tracking. Journal of Visual Communication and Image Representation, 25(1): pp. 137-147, 2014

[14] W. Wei, B. Shirinzadeh, M. Ghafarian, S. Esakkiappan and T. Shen, "Hector SLAM with ICP Trajectory Matching," 2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), pp. 1971-1976, doi: 10.1109/AIM43001.2020.9158946, Boston, MA, USA, 2020

[15] D. Wu, Y. Meng, K. Zhan and F. Ma, "A LIDAR SLAM Based on Point-Line Features for Underground Mining Vehicle," 2018 Chinese Automation Congress (CAC), pp. 2879-2883, doi: 10.1109/CAC.2018.8623075, Xi'an, China, 2018

[16] Martin Rünz, Lourdes Agapito. " Co-Fusion: Real-time Segmentation, Tracking and Fusion of Multiple Objects". In IEEE International Conference on Robotics and Automation (2017), pp. 4471-4478

[17] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, D. Cremers. " StaticFusion: Background Reconstruction for Dense RGB-D SLAM in Dynamic Environments". In IEEE International Conference on Robotics and Automation (2018)

[18] R. Hachiuma, C. Pirchheim, D. Schmalstieg. "DetectFusion: Detecting and segmenting both known and unknown dynamic objects in real-time SLAM". 30th British Machine Vision Conference, BMVC 2019, arXiv preprint arXiv:1907.09127 (2019)

[19] Y. Liu, W. Gao, Z. Hu. ” 3D Scanning of High Dynamic Scenes Using an RGB-D Sensor and an IMU on a Mobile Device”. IEEE Access Vol. 7. pp. 24057-24070 (2019 Febr.) doi: 10.1109/ACCESS.2019.2900740

[20] R. Gou, G. Chen, C. Yan, X. Pu, Y. Wu, Y. Tang. ”Three-dimensional dynamic uncertainty semantic SLAM method for a production workshop”. Engineering Applications of Artificial Intelligence 116 (2022 Nov.) doi: 10.1016/j.engappai.2022.105325

[21] Z. Ren, L. Wang and L. Bi, “Robust GICP-Based 3D LiDAR SLAM for Underground Mining Environment” MDPI, Sensors, 19, 2915; doi:10.3390/s19132915, 2019

[22] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In Proc. of the Intl. Symp. on Experimental Robotics (ISER) 2010

[23] P. Henry, M. Krainin, E. Herbst, et al. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. The International Journal of Robotics Research31(5): pp. 647-663, 2012

[24] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA), 2011

[25] M. I. A. Lourakis and A. A. Argyros. SBA: a software package for generic sparse bundle adjustment. ACM Transactions on Mathematical Software, 2009

[26] A. S. Huang. et al. (2017) Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera. In: Christensen H., Khatib O. (eds) Robotics Research. Springer Tracts in Advanced Robotics, Vol. 100, Springer, Cham. DOI: 10.1007/978-3-319-29363-9_14

[27] F. Endres, J. Hess, N. Engelhard, et al., An Evaluation of the RGB-D SLAM System IEEE International Conference on Robotics and Automation, pp. 1691-1696, 2012

[28] B. Yuan and Y. Zhang, “A 3D Map Reconstruction Algorithm in Indoor Environment Based on RGB-D Information,” 2016 15th International Symposium on Parallel and Distributed Computing (ISPDC), Fuzhou, 2016, pp. 358-363, DOI: 10.1109/ISPDC.2016.59

[29] T. Whelan, R.F. Salas-Moreno, B. Glocker, A.J. Davison, and S. Leutenegger. Elasticfusion: Real-time dense slam and light source estimation. The International Journal of Robotics Research: pp. 1697-1716, 2016

[30] M. Corsini, P. Cignoni, R. Scopigno, Efficient and Flexible Sampling with Blue Noise Properties of Triangular Meshes IEEE Transactions on Visualization and Computer Graphics, Vol. 18, pp. 914-924, 2012

[31] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, G. Taubin, The ball-pivoting algorithm for surface reconstruction IEEE Transactions on Visualization and Computer Graphics Vol. 5, Issue. 4, pp. 349-359, ISSN: 1077-2626, 1999

[32] J. Sturm, S. Magnenat, N. Engelhard, F. Pomerleau, F. Colas, et al. Towards a benchmark for RGB-D SLAM evaluation. RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conf. (RSS), Los Angeles, United States, 2011

[33] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald. "RealTime Large-Scale Dense RGB-D SLAM with Volumetric Fusion." The International Journal of Robotics Research 34, Nos. 4-5 (April 1, 2015): pp. 598-626, F. Endres, J. Hess, N. Engelhard, et al., An Evaluation of the RGB-D SLAM System IEEE International Conference on Robotics and Automation, pp. 1691-1696, 2012