Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 943510, 11 pages
http://dx.doi.org/10.1155/2015/943510

Research Article

3D Visual SLAM Based on Multiple Iterative Closest Point

Chunguang Li,[1,2] Chongben Tao,[3] and Guodong Liu[1]

[1]*School of Internet of Things Engineering, Jiangnan University, Wuxi 214122, China*
[2]*School of Computer and Information Engineering, Changzhou Institute of Technology, Changzhou 213002, China*
[3]*School of Electronic and Information Engineering, Suzhou University of Science and Technology, Suzhou 215009, China*

Correspondence should be addressed to Chunguang Li; leechunguang76@163.com

Received 5 February 2015; Accepted 10 May 2015

Academic Editor: Erik Cuevas

With the development of novel RGB-D visual sensors, data association has been a basic problem in 3D Visual Simultaneous Localization and Mapping (VSLAM). To solve the problem, a VSLAM algorithm based on Multiple Iterative Closest Point (MICP) is presented. By using both RGB and depth information obtained from RGB-D camera, 3D models of indoor environment can be reconstructed, which provide extensive knowledge for mobile robots to accomplish tasks such as VSLAM and Human-Robot Interaction. Due to the limited views of RGB-D camera, additional information about the camera pose is needed. In this paper, the motion of the RGB-D camera is estimated by a motion capture system after a calibration process. Based on the estimated pose, the MICP algorithm is used to improve the alignment. A Kinect mobile robot which is running Robot Operating System and the motion capture system has been used for experiments. Experiment results show that not only the proposed VSLAM algorithm achieved good accuracy and reliability, but also the 3D map can be generated in real time.

1. Introduction

Motivation. During the past ten years, most robot systems rely on two dimensional (2D) sensors for map creation, self-localization, and motion planning. Robot maps of indoor environments have often been 2D occupancy maps in the past, allowing robots to be localized in terms of position and orientation. However, in real world scenarios, robots have to face a three dimensional (3D) environment, not just 2D environment. The robot can have better performance in higher level missions if 3D environmental information can be obtained. Hence, many researchers now focus on 3D mapping and the RGB depth sensors (RGB-D) cameras are new alternatives to provide the useful information to create the map. Meanwhile, 2D Simultaneous Localization and Mapping based on a laser scanner has been implemented in so many cases that it may now be considered solved [1]. SLAM applications are numerous in hazardous or poisonous environments which require a robot with ability of SLAM-based navigation. Laser scanner is often performed accurately in SLAM, but it is expensive and has limited ability in complex environment.

In recent years, with development of different type of sensors, such as 3D laser scanners and stereo cameras, which have been used in Visual Simultaneous Localization and Mapping (VSLAM) [2], VSLAM has become increasingly important. In particular, RGB-D as one kind of 3D laser scanners are widely used to obtain 3D environment data in many practical applications [3–5]. A technique called distance imaging technology is used in RGB-D sensor which means that the generated 2D image shows the distance between one point and another point set in a scene. The obtained output is called a depth image, which contains a value corresponding to a distance value [2]. Besides, RGB-D cameras offer an alternative to create metric maps of an environment with known scale at a unit price much cheaper than other sensors such as laser range finders. This paper investigates how to use RGB-D sensors to create a 3D map. The problem can be stated as generating a digitized representation of an indoor environment, which can be used for automatic localization in that environment and other applications such as 3D VSLAM and 3D rendering. Most of existing research works on 3D mapping used features extracted from the color images [6, 7]. However, visual features are not robust enough in some situations

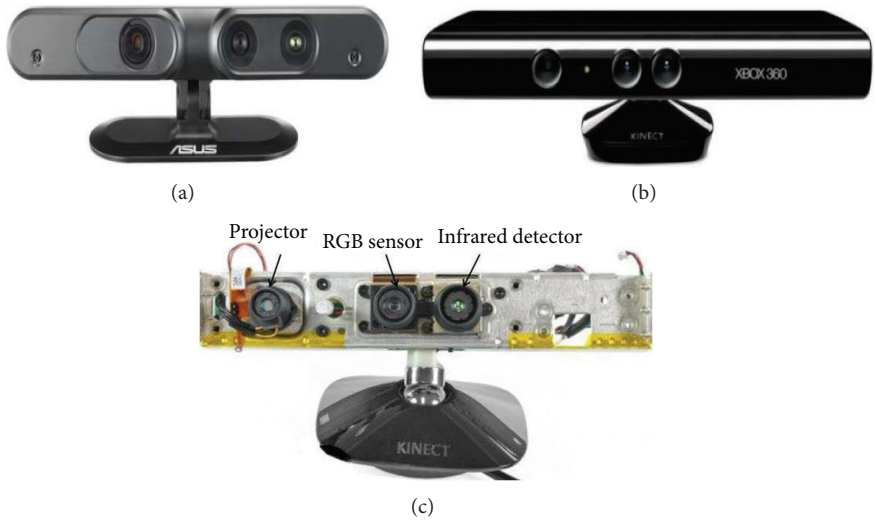


Figure 1: RGB-D sensors. (a) ASUS Xtion Pro live sensor, (b) Kinect sensor, and (c) internal structure of Kinect sensor.

such as when the environment lacks features or when the camera has strong movements. Our objective is to develop a system which can realize 3D VSLAM accurately, quickly, and reliably.

Related Work. Traditional 2D or 3D VSLAM methods usually use a monocular camera, a binocular camera, a trinocular stereo camera, a fish camera, or a panorama camera for visual SLAM [8]. Tong et al. [9] used a fish camera to get images and mapped wide-angle images to spherical images and then used Extended Kalman Filter (EKF) for VSLAM. Farrokhsiar and Najjaran [10] proposed a theoretical framework of Rao-Blackwellized Particle Filter in which higher order state variables and a modified undelayed initialization scheme are incorporated to solve the 3D monocular VSLAM problem. Currently, RGB-D sensors are widely used for various forms of researches. Novel RGB-D sensors like Microsoft Kinect and ASUS Xtion Pro live are two kinds of popular 3D sensors, as shown in Figure 1, which provide RGB images and depth images. In general, VSLAM approaches operating on RGB-D images are different from stereo systems as the input is RGB-depth images instead of color images. Izadi et al. [11] presented a Kinect fusion method to realize real-time 3D reconstruction and interaction by using a moving depth camera. Fioraio and Konolige [12] presented a system which used bundle adjustment to align dense point clouds of Kinect without using RGB images. Henry et al. [6] used sparse key point matches among consecutive color images as an initialization of Iterative Closest Point (ICP). They found that computationally expensive ICP step was only used if few or none key point matches could be established. Engelharda et al. [7] proposed a fundamental method for 3D visual SLAM with a hand-held RGB-D camera, which used SURF instead of SIFT features for feature extraction and matching. Random Sample Consensus (RANSAC) and ICP are, respectively, used for pose estimation and pose refinement. Lee et al. [13] used a 3D-RANSAC algorithm with image features and depth data for visual odometry estimation. And then they proposed an Incremental Smoothing and Mapping graph-based SLAM algorithm to optimize full trajectory. Jan et al. [14] proposed a feature extraction algorithm based on the Oriented FAST and Rotated BRIEF algorithm and tried to build upon relative distances between 3D data points based on a Relative Distance Measure. But they could not realize real-time 3D visual SLAM in a real sense. In this paper, a method based on Multiple Iterative Closest Point (MICP) and improved FastSLAM is presented to complete 3D VSLAM.

The rest of the paper is organized as follows: Section 2 describes the MICP-based 3D visual SLAM method. Section 3 introduces hardware and software of the mobile robot platform. The experimental results and comparison with other methods are shown in Section 4. Finally, Section 5 and last part end with a summary and acknowledgement.

2. 3D VSLAM Using a Kinect Sensor

Most traditional VSLAM methods are based on EKF algorithm [2, 6]. In this paper, a novel VSLAM algorithm based on MICP is proposed. Color (RGB) and depth (D) data are both used in the algorithm and it is generic due to the fact that it contains no motion model. Schematic overview of the proposed algorithm is shown in Figure 2. The proposed VSLAM mainly includes five steps.

Step 1. RGB-D sensor is used to collect depth and RGB images with synchronized timestamps. And then feature points from RGB images are extracted by SURF (Speeded Up Robust Feature) algorithm [15]. In order to match two kinds of feature points, extracted feature points from RGB images are projected to depth image. In this step, some uncertainty will be introduced into a chain of operations, not only due to synchronization mismatch between depth and RGB images, but also because of interpolation between points with large differences in depth. The fact is that a minor misprojection of a feature is lying on an object border onto the depth image which can result in a big depth error and makes features picked at object borders unreliable.

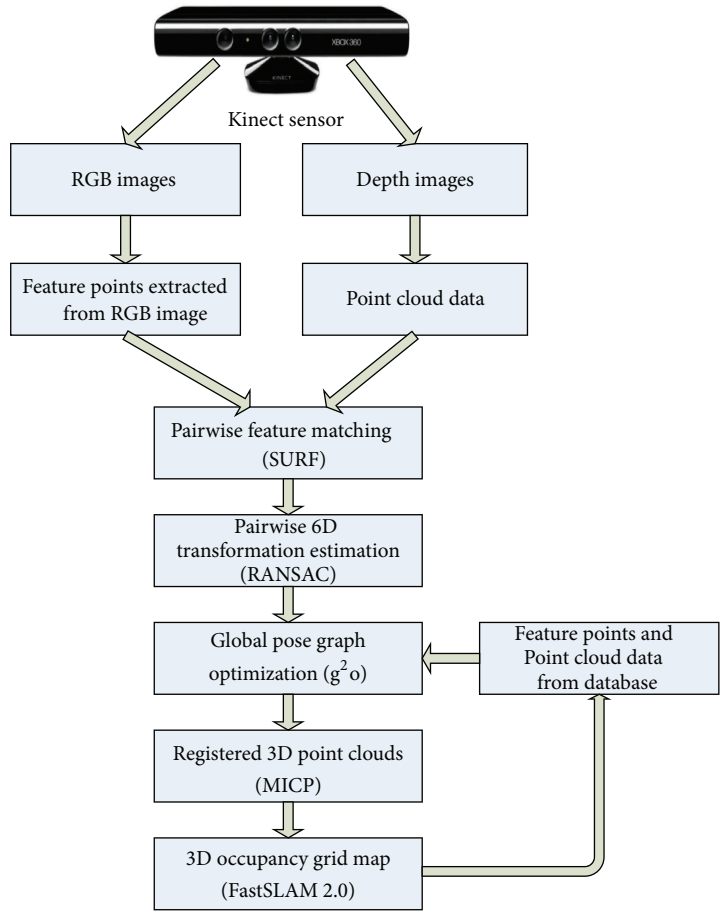


Figure 2: Schematic overview of the proposed 3D VSLAM algorithm.

Step 2. RANSAC algorithm is used to find a 6D transform for the camera position in noise [16]. Feature points are matched with earlier extracted feature points from a set of 30 images in database. The set consists of a subset including some of the most recent captured images and another subset including images randomly selected from the set of all formerly captured images. Three matched feature pairs are randomly selected and are used to calculate a 6D transform.

Step 3. In order to create a globally consistent trajectory, General Graph Optimization (g^2o) algorithm is used to realize global optimization [17], which is an easily extensible graph optimizer that can be applied to a wide range of problems including several variants of SLAM and bundle adjustment. And it performs a minimization of a nonlinear error function that can be represented as a graph.

Step 4. All feature pairs are then evaluated by their Euclidean distances to each other. Pairs whose Euclidian distance is below a certain threshold are counted as inliers. From these inliers a refined 6D transform is calculated using MICP.

Step 5. Registered point clouds are used by an improved FastSLAM 2.0 to complete 3D Occupancy Grid map [18].

2.1. RGB-D Sensor Calibration and Coordinate System Conversion. As position and orientation of RGB-D sensors in the world coordinate system cannot be obtained directly, we can infer the true pose of a sensor through calibrating markers which are on the top of the sensor. The purpose of calibration is to obtain the relationship between the sensor pose and the rigid body which is represented by markers. Three markers are mounted at the top of the sensor, which represent a rigid body which has an ability of rotation and translation. The calibration process involves some relationships among five coordinate systems, which are world coordinate system V (represented by VICON motion capture system) [19], marker coordinate system K, sensor coordinate system C, image coordinate system I, and target coordinate system O, as shown in Figure 3. After we obtain relationship between rigid body and RGB-D sensor, the position of rigid body is able to get through VICON motion capture system. And then position and orientation offset of the sensor is represented by the product of the rigid body position and a homogeneous matrix.

2.1.1. Calibration for RGB-D Sensor. The whole set of the RGB-D sensor has two key parts which are the color camera and the depth camera. The calibration for depth camera is already accomplished by the manufacturer. Then, using the

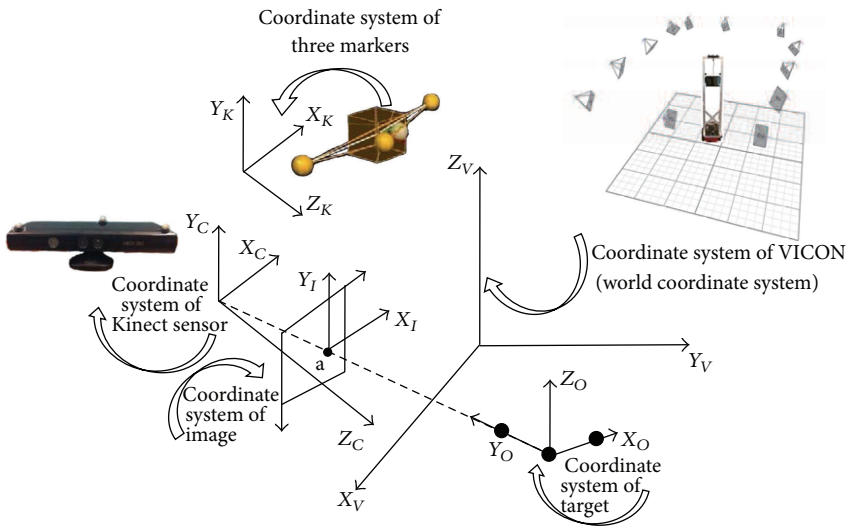


Figure 3: Schematic view of relationship among five different coordinate systems in calibration.

depth camera intrinsics, each pixel (x_d, y_d) in the depth camera can be projected to metric 3D space $P(x, y, z)$ using these following formulas:

$$x = \frac{(x_d - c_{xd}) \times \text{depth}(x_d, y_d)}{f_{xd}},$$
$$y = \frac{(y_d - c_{yd}) \times \text{depth}(x_d, y_d)}{f_{yd}}, \quad (1)$$
$$z = \text{depth}(x_d, y_d).$$

RGB-D sensor calibration is used to establish the relationship between the sensor and the world coordinate system. Before calibrating, we firstly need to discover the relationship between image coordinate system and world coordinate system, which involves the following two steps: the first step is to map the image space to a normalized 3D space using intrinsic parameters K and the second step is to transform the normalized 3D space to world coordinates using the rotation R and translation T. The formula is shown as follows:

$$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} x_v \\ y_v \\ z_v \\ 1 \end{bmatrix},$$
$$K = \begin{bmatrix} m_x & & \\ & m_y & \\ & & 1 \end{bmatrix} \begin{bmatrix} f & \frac{s}{m_x} & c_x \\ & f & c_y \\ & & 1 \end{bmatrix}, \quad (2)$$

where f is the focal length, m_x, m_y are the scale factors, s is the skew coefficient, and c_x, c_y are the principle points.

MATLAB camera calibration toolbox is used to obtain both extrinsic and intrinsic parameters of the camera. A chessboard is used to conduct the calibration. The calibration is based on Herrera's method [20]. Then extracted corner points and the extrinsic results are shown in Figure 4.

2.1.2. Homogeneous Transformation. A 3D coordinate transformation matrix is represented by a 4×4 homogeneous G which contains two-part information including the rotation R and translation T. The relationship among the five coordinates is as follows:

$$G_{CK} = G_{CO} G_{OV} G_{VK}, \quad (3)$$

where $G_{CO} = \begin{pmatrix} R_C & T_C \\ 0 & 1 \end{pmatrix}$, $G_{OV} = \begin{pmatrix} {}^{O}_{V}R & \\ 0 & 1 \end{pmatrix}$, and $G_{VK} = \begin{pmatrix} R_V & T_V \\ 0 & 1 \end{pmatrix}$.

R_C, T_C are the extrinsic result of camera calibration. R_V, T_V are the data obtained from the VICON capture system. G_{OV} can be obtained from the frame represented by the plane of the chessboard. So the offset G_{CK} is represented by the product of these matrices. Then the camera pose is obtained by multiplying this matrix with the coordinate system of the three markers.

2.2. Feature Extraction and Matching. In order to estimate spatial relationship of sensor pose from images, it is necessary to use image feature. And relationship between any two images is achieved by matching image feature points. In the paper, Speeded Up Robust Feature (SURF) algorithm [15] is used for feature extraction and matching. SURF algorithm includes detectors and descriptors, which are used to extract feature and discover consistency between two feature points, respectively. Specific implementation of SURF algorithm is divided into three steps: feature point detection, construction of feature point descriptor, and fast matching.

Step 1 (feature point detection). This step is divided into integral images, approximately Hessian matrix discriminate extreme point, scale space description, and features location (x, y, σ). Integral image calculates pixels sum in a rectangular

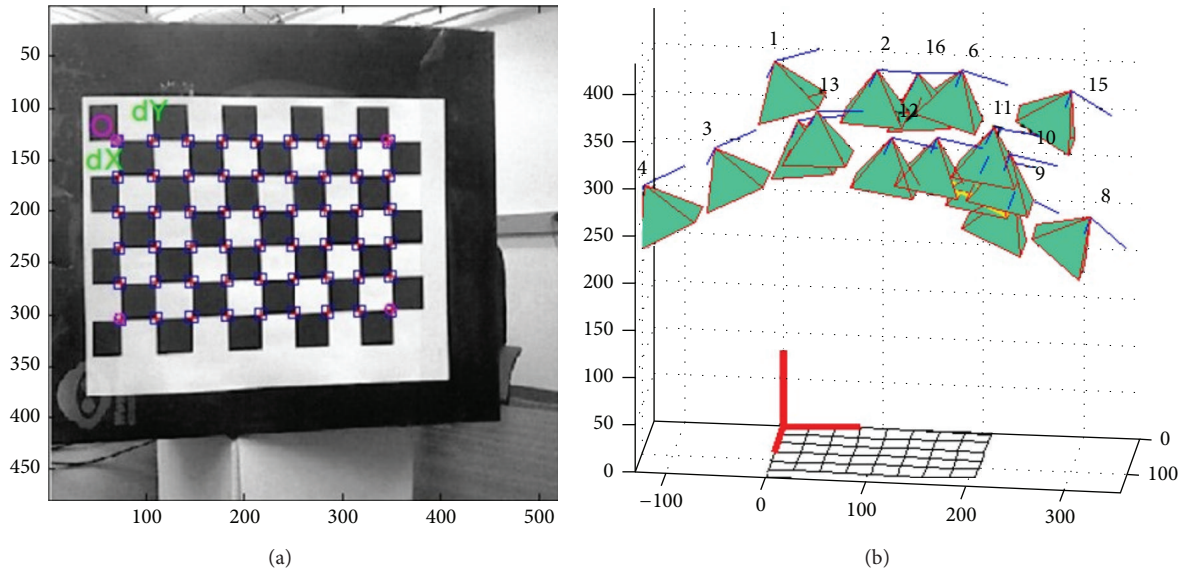


Figure 4: Corner points extraction (a) and results for extrinsic parameters (b).

area. Integral image $I_{\sum}(x)$ at a point $x = (x, y)$ of image I is defined as follows:

$$I_{\sum}(x) = \sum_{i=0}^{i\leqslant x}\sum_{j=0}^{i\leqslant y} I(i,j). \tag{4}$$

In Hessian matrix in a given image I at a point $x = (x, y)$, scale σ is defined as

$$H(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix}, \tag{5}$$

where $L_{xx}(x,\sigma)$ is a divalent partial derivative of Gaussian $\partial^2/\partial x^2$, which convolute image I at a point x. Box filters approximate Gaussian function in order to accelerate integral images; the approximate Hessian matrix is as follows:

$$\det\left(H_{\text{approx}}\right) = D_{xx}D_{yy} - \left(\omega D_{xy}\right)^2, \tag{6}$$

where weight ω is a parameter, which is used to balance Hessian determinant expression.

Scale-space description is usually used to describe pyramid image. After calculating extreme value at the point of (x, y, σ) in scale image according to Hessian matrix, a quadratic fitting function is used for interpolation:

$$D(X) = D + \frac{\partial D^T}{\partial X}X + \frac{1}{2}X^T\frac{\partial D}{\partial X^2}X. \tag{7}$$

Derive (7) and obtain the extreme value at extreme points

$$D(X) = D + \frac{\partial D^T}{\partial X}\widehat{X} \geqslant 0.03. \tag{8}$$

Step 2 (construction of feature point descriptor). Descriptor is used to describe pixel values distribution of feature points in scale space field. Haar wavelet response of partial derivative at x and y directions is used to create distribution of 64D, which is accelerated by integral image at the same time:

$$V = \left(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\right). \tag{9}$$

Step 3 (fast matching). Fast matching is divided into preliminary matching and exact matching. Euclidean distance of feature vector is used in preliminary matching as a similarity judgment measure of key points in two images. Kd-tree search [21] is used to discover Euclidean distance of original feature points in prepared register image:

$$\text{Dis}_{ij} = \left[\sum_{k=0}^{k=n}\left(X_{ik} - X_{ik}\right)^2\right]^{1/2}, \tag{10}$$

where X_{ik} means the kth element of the ith feature descriptor in prepared register image; X_{jk} is the kth element of the jth feature descriptor in reference image; n represents dimension number of feature vector.

2.3. Global Pose Graph Optimization. RANSAC is an iterative algorithm which is used to adapt parameters of a mathematical model to experimental data and is a suitable method when a data set contains a high percentage of outliers, that is, measurements that suffer from measurement errors so large that validity of the measurements is low. RANSAC is the robust way that handles outliers in the data set, and it is suggested to be a suitable method for feature matching in VSLAM. More details of the algorithm can be found in [16].

g^2o implements a general and efficient graph optimizer. A graph optimizer achieves a minimization of a nonlinear error function represented as a graph. Nodes represent vectors of parameters and edges how well two parameter vectors match to external constraint, relating to the two parameter vectors. In the case where g^2o is used by VSLAM for global pose

graph optimization, each node in the graph represents a state variable of the robot and each edge represents a pairwise observation between nodes that are connected by edge. The algorithm is fast due to sparse connectivity property of graph and advanced solvers of sparse linear systems are used. More thorough presentation is presented in [17].

2.4. 3D Point Clouds Register. ICP algorithm is one of the most widely used intuitive registration methods [22]. The ICP algorithm can be summarized in two key steps.

Step 1. Find matched feature points between two point clouds.

Step 2. Compute a transform T, which includes a rotation and translation such that it minimizes the Euclidian distance between matched points.

In the paper, we propose a MICP algorithm, which can be thought of as a plane-to-plane matching algorithm. The most likelihood estimate of the transform can be achieved by assigning probabilistic properties to extracted features. Furthermore, probabilistic framework can make use of all general research in probabilistic techniques to increase robustness, such as outlier rejection. The probabilistic model assumes that points in the measured point clouds $\widehat{D} = \{\widehat{\alpha}_i\}$ and $\widehat{S} = \{\widehat{s}_i\}$ are independently Gaussian distributed [12]. This will generate probabilistic point clouds D and S with points $\alpha_i \sim N(\widehat{\alpha}_i, C_i^D)$ and $s_i \sim N(\widehat{s}_i, C_i^S)$. Here C_i^D and C_i^s are three-dimensional covariance associated with the measured points. Define T^* as the correct transform when perfect compliance between points occurs as

$$\widehat{s}_i = T^* \widehat{d}_i. \tag{11}$$

Defining the function

$$e_i(T^*) = s_i - T^* d_i \tag{12}$$

and considering this function evaluated with T^*, as in (13), give the probabilistic function

$$e_i(T) \sim N\left(\widehat{s}_i - (T^*)\,\widehat{d}_i, C_i^S + (T^*)\,C_i^D\,(T^*)^T\right). \tag{13}$$

This is an objective function for optimization and the most likelihood estimate can be found solving

$$T = \underset{T}{\operatorname{argmax}} \left\{ \sum_i e_i(T)^T \left(C_i^S + T C_i^D T^T\right)^{-1} e_i(T) \right\}. \tag{14}$$

The solution of (14) gives a transform which can be used to optimize the point cloud match in a probabilistic manner. By setting $C_i^S = I$ and $C_i^D = 0$, the standard point-to-point MICP algorithm is obtained. In this case (14) becomes

$$\begin{aligned} T &= \underset{T}{\operatorname{argmax}} \left\{ \sum_i e_i(T)^T e_i^T \right\} \\ &= \underset{T}{\operatorname{argmax}} \left\{ \sum_i \left\| e_i(T)^T \right\|^2 \right\}, \end{aligned} \tag{15}$$

which coincides a point-to-point MICP algorithm for matched features.

In the first layer, firstly ICP algorithm is used in the MICP algorithm to align the current frame with the previous frame. If estimation error is extremely small, then rotation matrix and translation matrix will be updated. In the second layer, if matching fails due to situations such as shaking and sudden change of the sensor direction, then ICP algorithm will be started again until the nearest frame in map M is found out to match the current frame. If both of the two levels of registration failed, the current frame will be discarded; if a frame is at a distance of d with all frames in map M, then it will be considered as an update and added into map M.

The detail of the proposed MICP algorithm is shown in Algorithm 1.

2.5. 3D Occupancy Grid Mapping. FastSLAM algorithm decomposes VSLAM problem into mobile robot localization problem and landmark position estimation problem. And particle filters are used to realize mobile robot pose estimation of the entire path; meanwhile, EKF is used for estimating landmark position. The FastSLAM algorithm has two versions 1.0 and 2.0 [23]. In the paper, the improved FastSLAM 2.0 version algorithm was used [18].

We analyze and solve VSLAM problem based on the perspective of probability. Firstly, we need to estimate the robot pose x_k at time k and joint posterior probability density $p(x_k, M \mid Z_{0:k}, U_{0:k})$, of landmark position M. For convenience, $b_k(x_k, M)$ is used to replace $p(x_k, M \mid Z_{0:k}, U_{0:k})$, which is like a Markov chain, that is, the current state $b_k(x_k, M)$ only related with the former state $b_{k-1}(x_{k-1}, M)$, which can be expressed as a probability model as follows:

$$\begin{aligned} &b_k(x_k, M) \\ &\quad = \eta \cdot p(z_k \mid x_k, M) \\ &\qquad \cdot \int p(x_k \mid x_{k-1}, u_k)\, b_{k-1}(x_{k-1}, M)\, dx_{k-1}, \end{aligned} \tag{16}$$

where η is a normalization constant, $p(z_k \mid x_k, M)$ is a probability of observation model, and $p(x_k \mid x_{k-1}, u_k)$ is a probability of motion model.

From entire trajectory of the mobile robot, we can obtain the following formula:

$$b_k(x_{0:k}, M) = b_k(X_{0:k}) \prod_{i=1}^{n} p(m_i \mid x_{0:k}, Z_{0:k}). \tag{17}$$

Equation (17) decomposes the joint probability density of robot pose and landmark position into estimate $n+1$: one for the robot trajectory $b_k(X_{0:k})$, the remaining n is the position estimate of each landmark on the basis of known path.

Particle filters are used in FastSLAM algorithm use to estimate trajectory of the mobile robot $b_k(X_{0:k})$ [24], each particle in the particle filter represents a path estimate and associates with n independent EKF, and each EKF corresponds to a landmark. Particle jth at time k can be expressed as follows:

$$S_k^i = \left\{ x_k^j, \mu_{l,k}^j, \Sigma_{l,k}^j, \ldots, \mu_{n,k}^j, \Sigma_{n,k}^j \right\}, \tag{18}$$

(1) **for** current frame i **do**
(2) $(R^i_{\text{ini}}, t^i_{\text{ini}}) \leftarrow$ motion data
(3) previous frame $i-1$ with $(R^{i-1}_{\text{ini}}, t^{i-1}_{\text{ini}})$
(4) *ICP*(current frame i, previous frame $i-1$)
(5) **if** *ICP* succeeds **then**
(6) *Update* $(R^i_{\text{est}}, t^i_{\text{est}}) \leftarrow (R^i_{\text{ini}}, t^i_{\text{ini}})$
(7) **else**
(8) find nearest frame p $(R^p_{\text{est}}, t^p_{\text{est}})$ with in map M
(9) *ICP*(current frame i, frame p)
(10) **if** *ICP* succeeds **then**
(11) *Updated* $(R^i_{\text{est}}, t^i_{\text{est}}) \leftarrow (R^i_{\text{ini}}, t^i_{\text{ini}})$
(12) **else**
(13) *Updated* failed, discard the frame
(14) **end if**
(15) **end if**
(16) **if** *Update* succeeds **then**
(17) **if** $\arg\max_{p\in M}\|(R^i_{\text{est}}, t^i_{\text{est}} - (R^p_{\text{est}}, t^p_{\text{est}}))\| >$ threshold **then**
(18) add frame i to M
(19) **end if**
(20) **end if**
(21) **end**

Algorithm 1

where $\mu^j_{l,k}$ and covariance $\Sigma^j_{l,k}\ldots$ are Gaussian parameters of posterior probability density of each landmark, so each EKF estimated a landmark; therefore, it is a low-dimensional EKF.

The improved FastSLAM algorithm is used in the paper, and the process is as follows [24].

Step 1. For each particle in S_{k-1}, a new position and orientation x^j_k is used

$$x^j_k \sim p\left(x_k \mid X^j_{0:k-1}, Z_{0:k-1}, U_{0:k-1}\right). \tag{19}$$

Step 2. Update estimation of landmark observation. The observation function $h(\cdot)$ is linear, and its adopted update mode is similar to the EKF-SLAM.

Step 3. Resampling: each particle is given a weight w^j_k = Target Distribution/Proposal Distrution. m particles are resampled based on probability and then start the next round of iteration.

3. Hardware and Software Design

3.1. Hardware Setup. In this paper, a smart mobile robot platform is proposed for VSLAM experiments. Hardware of the platform is mostly made up of Pioneer 3DX robot [25], microcomputer FitPC2 [26], laser range finder, and Kinect sensor [27], as shown in Figure 5. Mobile chassis of the platform is based on Pioneer 3DX robot. FitPC2 is a kind of fanless microcomputer; both of Windows Operating System and Linux Operation System can be used in it. Kinect sensor incorporates several technologies, including RGB imaging, 3D imaging, audio processing, and motor control. Our mobile platform only uses its camera function, which is developed by Israel PrimeSense Company based on

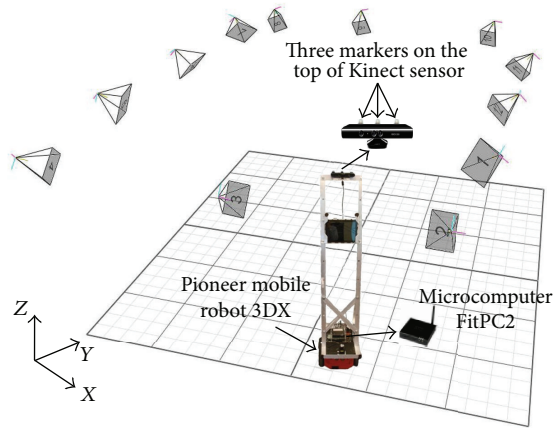


Figure 5: A Kinect-based mobile robot platform and the VICON motion capture system.

RGB camera and depth sensor [28]. This device is connected to a computer via USB port. Its effective sensing range is from 0.4 meters to 4 meters, vertical viewing angle range is ±43°, horizontal range is of ±57°, and frame rate is of 30 fps. Beyond major components mentioned above, an external battery is used to supply power for FitPC2 computer and Kinect sensor. And a USB-powered minifan is used to cool FitPC2.

3.2. Software Setup. Robot Operating System (ROS) is an open source metaoperating system, including low-level device control, wireless communications, and software package management [29]. ROS has two basic parts, one is the core part of ROS which functions as an "Operation System" and the other part is the packages contributed by the whole ROS community. In ROS, a program can be divided

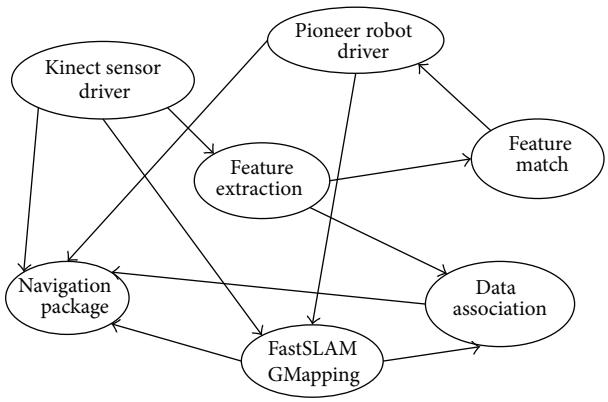


Figure 6: Structure chart of nodes in our ROS.

into different nodes which can be distributed to different computing devices in the same network. According to the form of software packages, ROS can effectively create and manage programs. Executable files are executed by "Nodes." A program can be divided into different "Nodes" in ROS. These "Nodes" can be running on different computers in the same network. A node is an independent process, which can receive and publish "Topics" from other "Nodes." A part of the hardware can be regarded as a node, and a data processing algorithm also can be used as a node, as long as all these "Nodes" are working in the same "ROS Master." The whole algorithm can be successfully achieved in such a distributed mode.

There are two kinds of "Nodes" in our ROS. One kind is hardware drivers including Pioneer robot driver node and Kinect driver node. Another is functional procedures including feature extraction node, feature point match node, data association node, and FastSLAM GMapping node, as shown in Figure 6. Pioneer robot 3DX driver program is called "ROSARIA package," developed by Aria library; Kinect sensor driver program is called "openni_kinect" which comes from ROS community. Programs of feature extraction node, feature point match node, and data association node are written by us except the FastSLAM GMapping package which is used for robot localization in this part; FastSLAM GMapping package and navigation package are supplied by ROS community which are used for VSLAM and autonomous navigation.

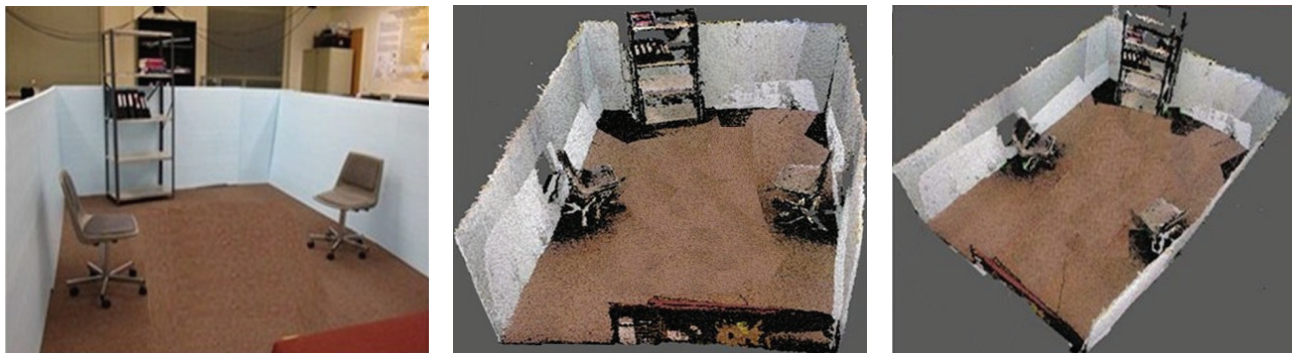
4. Experiment Results and Comparison

A VICON motion capture system [19] is mounted in a laboratory which was used as a test environment, as shown in Figure 5. A Kinect sensor was used as a RGB-D sensor, and three markers were fixed on the top of Kinect as a rigid body, which is used for camera calibration. ROS [29] was used as a software platform to implement the proposed algorithm. All experiments were carried out on a powerful computer who has a hexa-core CPU with 8 GB of memory. All processes (also called "Nodes") are developed in accordance with the framework of ROS, including Kinect drivers, image compression and processing procedures, movement data receiver program, point cloud data processing, and display program.

4.1. Evaluation of the Proposed MICP-Based 3D VSLAM. In this section, we firstly present results on the accuracy from the proposed MICP-based visual SLAM approach and then evaluate the accuracy, reliability, and real-time performance of proposed method. Finally, we investigate the influence of various parameters on the running time of our VSLAM method. In order to evaluate our proposed method, an evaluation tool that computes the root mean square error (RSME) [30, 31] is used which provides an evaluation for our visual SLAM method. Furthermore, a VICON motion capture system is used to provide precise RGB-D sensor positioning data. In order to assess validity and reliability of the proposed method, VSLAM tests were conducted in an artificial environment and an office environment, respectively. Meanwhile, a VICON motion capture system was used to verify accuracy of the proposed algorithm. According to the proposed MICP algorithm, a handle was used to control the mobile robot to surround the room at a steady speed, and then a 3D map of the entire environment was created by Kinect sensor mounted on the robot. After 50 frames were captured, a 3D map could be generated quickly. The VSLAM results are respectively shown in Figures 7 and 8. Figure 7(a) is the 2D image of an artificial environment; Figures 7(b) and 7(c) are images from the front view and top view of the 3D artificial environment, respectively. Figures 8(a) and 8(b) are images from left corner and right corner of an office, respectively; Figures 8(c) and 8(d) are images from the front view and top view of the 3D artificial environment, respectively.

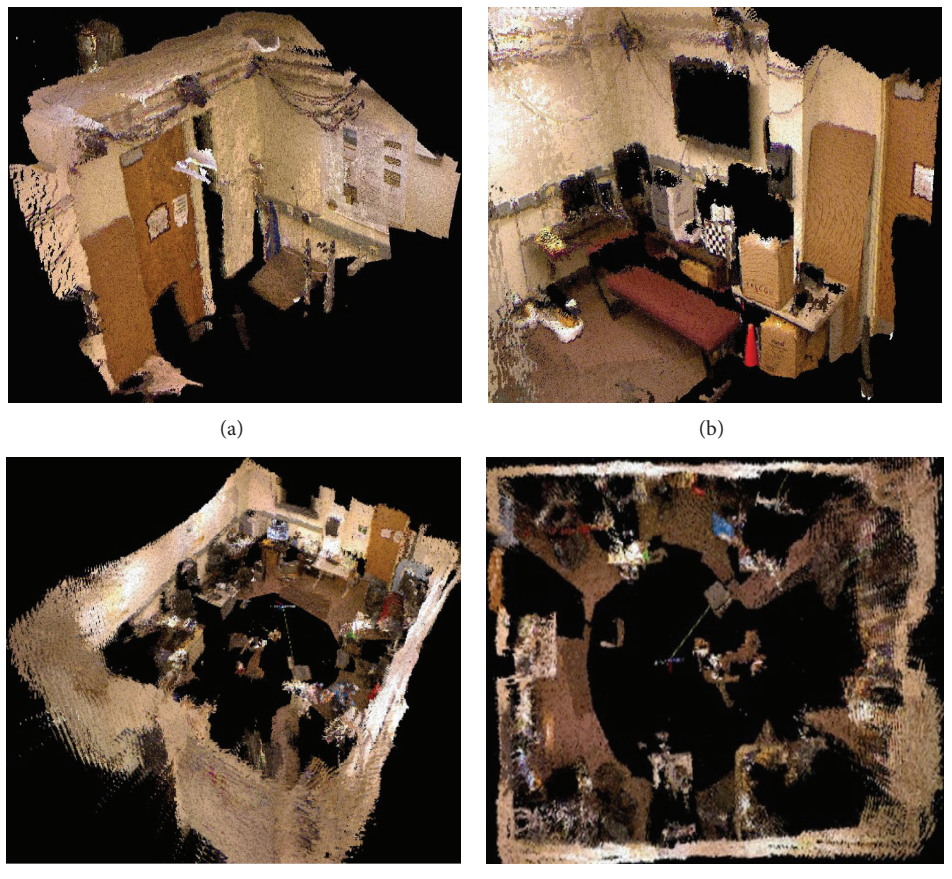
In the first round of experiments, we evaluated the accuracy of the proposed MICP on all objects in the artificial room and in the office. On the objects "Floor" and "Desk," we, respectively, obtained the best values of 1.9 cm and 4.1 cm RMSE error. Meanwhile, we achieved the worst result of 21.8 cm RMSE error on the object "Bookshelf," as indicated by numbers in bold in Table 1. From Table 1, we can find that the range of camera degrees is varied from 15.48 deg/s to 66.23 deg/s, and the range of velocities is varied from 0.09 m/s to 0.47 m/s. And the proposed method achieves on average an accuracy of 11.9 cm and 4.16°. Furthermore, high angular and translational velocity pose have no obvious difficulty even though some frames had a certain level of overlap. In general, these evaluation results show that our approach performs well in most of these objects. And with an average frame processing time of 0.618 s our approach is suitable for online operation.

Results on these objects from Table 1 show that the capabilities of the proposed approach are in the best case. However, good results can usually be achieved when the robot is carefully moved in an indoor environment. Generally, a map accomplished by someone who hold a RGB-D sensor is better than other map which is built by RGB-D sensor fixed on a mobile robot, as human being can keep the RGB-D sensor moving in a uniform velocity. However, when someone uses a joystick or a computer to control a robot, it is usually hard to keep the robot moving at a constant speed.



(a) (b) (c)

Figure 7: 3D VSLAM in an artificial room. (a) Original image, (b) front view, and (c) top view.



(a) (b)

(c) (d)

Figure 8: 3D VSLAM in an office. (a) Left corner, (b) right corner, (c) front view, and (d) top view.

On the other side, when the robot tries to cover areas of a larger space, data from RGB-D sensor are more challenging as the unrestricted camera motions.

4.2. Comparison of Our MICP-Based 3D VSLAM with Other Methods. Furthermore, we compared the proposed method with method based on Kinect fusion [11], method based on RGB-D SLAM [7], and method only based on ICP, respectively. Comparison results are shown in Table 2. The Kinect fusion-based method can achieve the highest accuracy of 5 mm, but its reliability only can last for 21 hours. Another method based on RGB-D SLAM had the fastest speed of 0.5 fps. However, all calculations are based on hardware. Additionally, its process requires a lot of memory to store data, so the created map size is limited.

Besides, when we only used the ICP method in the proposed algorithm, the running speed of the algorithm was nearly 6 fps. But when MICP was used in the algorithm,

Table 1: Data from an artificial room and an office.

Object name	Length (m)	Average angle velocity (deg/s)	Average translation velocity (m/s)	Frames	Translation RMSE (m)	Rotation RMSE (°)	Running time (s)
Desk	3.12	51.37	0.24	127	0.041	1.27	75
Door	1.25	60.89	0.22	98	0.137	4.62	33
TV	1.32	63.7	0.13	103	0.092	3.86	47
Bed	2.81	57.23	0.16	142	0.104	4.03	70
Wall	13.28	20.31	0.35	552	0.125	4.49	528
Window	2.24	52.74	0.27	134	0.149	4.86	64
Chair	0.53	66.23	0.09	75	0.087	3.51	19
Bookshelf	1.78	59.61	0.15	169	0.218	8.34	42
Floor	18.27	18.94	0.39	832	0.019	0.43	519

Table 2: Comparison among different methods.

Method	Average speed (fps)	Memory (GB)	Accuracy (mm)	Reliability (h)
Kinect [9]	3	20.5	5	21
RGB-D SLAM [12]	0.5	17.8	8	43
Only ICP	6	11.4	17	87
Proposed MICP	0.6	9.1	9	102

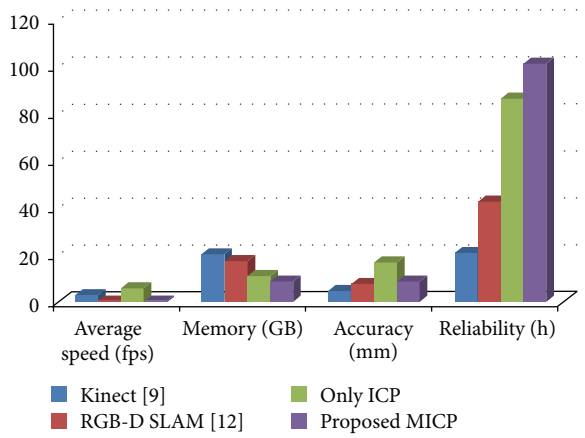


Figure 9: Comparison among different methods.

the processing speed was 0.6 fps, which was tenfold of ICP. As shown in Figure 9, we can see clearly the average speed,the memory, the accuracy, and reliability relationship among these four methods. Therefore, the proposed method is more effective than the other three methods.

5. Conclusions

In this paper, a MICP-based data association approach is proposed to realize 3D VSLAM for a Kinect mobile robot. Features extracted from color images and depth images by SURF algorithm are used to localize them in 3D images. RANSAC algorithm is used to estimate transformations between RGB-D frames and g^2o algorithm is used to optimize the pose graph. Then the proposed MICP algorithm is used to register point clouds. Finally, these registered point cloud data are used in an improved FastSLAM 2.0 algorithm to generate a 3D Occupancy Grid map of an indoor environment, which can be used for robot localization, path planning, and navigation. We have evaluated our approach based on a VICON motion capture system and compared the proposed method with other three methods. Experimental results showed that the proposed approach has good accuracy, reliability, and real-time performance.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This project is supported by the National Natural Science Foundation of China (60277605).

References

[1] G. Dudek and M. Jenkin, *Computational Principles of Mobile Robotics*, Cambridge University Press, Cambridge, UK, 2010.

[2] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '12)*, pp. 1691–1696, 2012.

[3] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE Multimedia*, vol. 19, no. 2, pp. 4–10, 2012.

[4] L. M. Paz, P. Piniés, J. D. Tardós, and J. Neira, "Large-scale 6-DOF SLAM with stereo-in-hand," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 946–957, 2008.

[5] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6d slam 3d mapping outdoor environments: research articles," *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 699–722, 2007.

[6] 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 *Experimental Robotics: The 12th International Symposium on Experimental Robotics*, vol. 79, pp. 477–491, Springer, Berlin, Germany, 2014.

[7] N. Engelharda, F. Endresa, J. Hess, J. Sturm, and W. Burgard, "Real-time 3D visual SLAM with a hand-held RGB-D camera," in *Proceedings of the RGB-D Workshop on 3D Perception in Robotics*, Vasteras, Sweden, 2011.

[8] J. Aulinas, Y. Petillot, J. Salvi, and X. Lladó, "The SLAM problem: a survey," in *Proceedings of the Conference on Artificial Intelligence Research and Development*, pp. 363–371, 2008.

[9] G. Tong, Z. Wu, and J. Tan, "Omni-vision mobile robot vSLAM based on spherical camera model," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO '11)*, pp. 829–834, IEEE, December 2011.

[10] M. Farrokhsiar and H. Najjaran, "Monocular vSLAM using a novel Rao-Blackwellized particle filter," in *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM '10)*, pp. 73–78, July 2010.

[11] S. Izadi, D. Kim, O. Hilliges et al., "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*, pp. 559–568, October 2011.

[12] N. Fioraio and K. Konolige, "Realtime visual and point cloud slam," in *Proceedings of the RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conference (RSS '11)*, 2011.

[13] D. Lee, H. Kim, and H. Myung, "2D image feature-based real-time RGB-D 3D SLAM," in *Robot Intelligence Technology and Applications 2012: 1st International Conference on Robot Intelligence Technology and Applications*, vol. 208 of *Advances in Intelligent Systems and Computing*, pp. 485–492, Springer, Berlin, Germany, 2013.

[14] H. Jan, F. Dariush, L. Marek, K. Jan Helge, and E. Maehle, "Real-time visual SLAM using FastSLAM and the microsoft Kinect camera," in *Proceedings of the 7th German Conference on Robotics (ROBOTIK '12)*, pp. 1–6, Munich, Germany, May 2012.

[15] H. Bay, A. Ess, T. Tuytelaars, and L. van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[16] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[17] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: a general framework for graph optimization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '11)*, pp. 3607–3613, Shanghai, China, May 2011.

[18] R. Havangi, M. A. Nekoui, and M. Teshnehlab, "An improved fastslam framework using soft computing," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 20, no. 1, pp. 25–46, 2012.

[19] VICON Motion Capture Systems, http://www.VICON.com/.

[20] C. Daniel Herrera, J. Kannala, and J. Heikkilä, "Joint depth and color camera calibration with distortion correction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 10, pp. 2058–2064, 2012.

[21] K. Zhou, Q. Hou, and R. Wang, "Real-time kd-tree construction on graphics hardware," *ACM Transactions on Graphics*, vol. 27, no. 5, pp. 153–160, 2008.

[22] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[23] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–108, 2006.

[24] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[25] Pioneer robot 3DX, http://www.mobilerobots.com/researchrobots/pioneerp3dx.aspx/.

[26] Fitbit, http://www.fitbit.com/.

[27] Microsoft Kinect, http://www.microsoft.com/en-us/kinectforwindows/.

[28] PrimeSense, http://www2.technologyreview.com/tr50/primesense/.

[29] M. Quigley, K. Conley, B. P. Gerkey et al., "Ros: an open-source robot operating system," in *Proceedings of the ICRA Workshop on Open Source Software*, 2009.

[30] J. Sturm, S. Magnenat, N. Engelhard et al., "Towards a benchmark for rgb-d slam evaluation," in *Proceedings of the RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conference (RSS '11)*, Los Angeles, Calif, USA, June 2011.

[31] R. Kümmerle, B. Steder, C. Dornhege et al., "On measuring the accuracy of SLAM algorithms," *Autonomous Robots*, vol. 27, no. 4, pp. 387–407, 2009.