ATLANTIS PRESS

Advances in Engineering Research, volume 140

2nd International Conference on Electrical, Control and Automation Engineering (ECAE 2017)

Sparse Direct Robot Localization Method Based on RGB-D Camera

Rongbo Hou, Wu Wei*, Yeboah Yao and Ting Huang

College of Automation Science & Engineering, South China University of Technology, Guangzhou Guangdong, China

*Corresponding author

***Abstract*—In order to address the current challenges associated with feature-based RGB-D SLAM, this paper puts forward a novel sparse direct localization algorithm. Contributions of the paper are manifold. Firstly, the proposed algorithm achieves rapid feature-point detection as well as camera pose estimation through a minimization strategy of the photometric error associated with image coupling. Secondly, a computational optimization scheme is put forward for the proposed algorithm such that key-frames are selected adaptively using a spatial domain framework which monitors the robot's motion in real-time, and applies a Nearest Neighbor algorithm towards loop closure detection. Finally, the proposed algorithm achieves robot pose estimation and optimization in real-time using a General Framework for Graph Optimization (g2o) strategy. The performance of the proposed algorithm is verified through live robotic experimental evaluation. The achieved results suggest that the scheme attains significantly high localization accuracies with low RMSE within a range of 25 meters. For scenarios where the camera remains fixed, RMSE reaches up to 1% within a range of 29.6 meters. Furthermore, the proposed scheme achieves localization speeds of up to 45 fps, demonstrating superior real-time capabilities, and addressing computational drawbacks associated with state-of-the-art.**

Keywords—simultaneous localization and mapping; sparse direct method; keyframe selection; kinect

I INTRODUCTION

Over the past decade, outdoor positioning has been rapidly developed and widely utilized based on satellite technologies, such as the Global Positioning System (GPS). However, we are more than 70% of the time indoors, therefore, indoor positioning technology offers great research and application value. Indoor environments are complex in nature, and it is therefore necessary to estimate the real-time position of mobile nodes from a series of measurement data. To the best of our knowledge, there are no working solutions with sufficient application potential thus far [1].

In the field of robot localization, the Simultaneous Localization and Mapping (SLAM) algorithm, based on laser and vision has been widely relied upon. SLAM makes use of hardware sensors' (e.g., camera, laser) measurement data to establish the environmental mapping schemes for estimating the position of the mobile robot [2]. SLAM has an important research significance in robotic control, navigation and mission planning [3]. On the other hand, sensors are an important part of SLAM. Currently, sensors commonly used include laser radar, monocular/stereo/panoramic cameras, RGB-D cameras, Inertial Measurement Units and multiple sensor fusion schemes. Amongst them, RGB-D cameras (e.g., Kinect), which can simultaneously obtain the pixel colors and depth information, have been widely researched in recent years. Although the RGB-D SLAM system has evolved rapidly, it is still faced with some challenging issues such as a small Field of View (FOV) associated with the RGB-D camera as well as high computational loads.

The traditional RGB-D SLAM system is mainly composed of front-end and back-end. The front-end mainly constructs the pose graph, including image preprocessing, feature extraction and matching, motion estimation between consecutive frames and loop closures detection. The back-end is also referred to as the graph optimization end, which globally optimizes the pose using the pose graph constructed by the front-end. The loop closures detection can be regarded as an image recognition problem and can be available as a separate subsystem.

At the front end of the RGB-D SLAM system, sparse image features are usually extracted as visual landmarks such as Harris [4], FAST [5], SIFT [6], SURF [7], ORB [8]. Such algorithms have been hugely relied upon because they attain high accuracies in the areas of localization accuracy, repeatability, computational efficiency, stability, uniqueness and invariance. Firstly, the feature points are extracted on each frame. Then the feature points between the two frames are matched by the high-dimensional feature descriptors. Finally, the camera motion is estimated by random sampling and consensus (RANSAC) and the iterative closest point (ICP) [9, 10] method. It can been seen that: (1) This feature-based approach only uses the sparse feature point information, ignoring most of the information in the image such as a straight line or an edge; (2) The performance seriously depends on the feature extraction and matching accuracies. Furthermore, it requires the removal of false matching points; (3) It requires a certain amount of calculation in order to match high-dimensional feature descriptors, limiting the overall speed of the system.

In order to solve the above problems in the traditional RGB-D SLAM system, this paper proposes a sparse direct RGB-D SLAM method. We firstly detect the pixel gray gradient points (FAST image feature points) and estimate the camera pose by minimizing the photometric error between image pairs. Secondly, the keyframes are selected based on a spatial domain method. That is, the frame will be selected as a key frame when the motion of the robot exceeds a certain threshold. We then apply a metrical nearest neighbor search strategy to detect loop closures. A loop closure candidate will then be searched within a sphere with a predefined radius r

around the current keyframe position. Finally, the robot's pose is optimized through implementation with the g2o framework to optimize the pose graph.

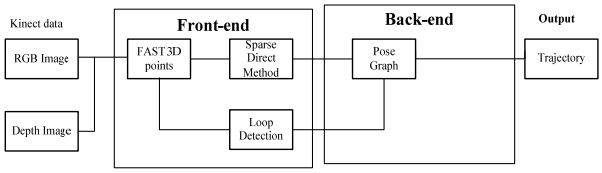

FIGURE I. SCHEMATIC OVERVIEW OF SPARSE DIRECT RGB-D SLAM SYSTEM

II RELATED WORK

Visual Simultaneous Localization and Mapping (VSLAM) uses monocular cameras, stereo cameras and RGB-D cameras as the main sensors in order to localize the robot and map the unknown environment [11]. The VSLAM system is mainly composed of visual odometry, loop closures detection and mapping. This section provides a brief overview of the visual odometry state-of-the-art, which can be divided into feature-based method and direct method.

A. Feature-based Method

This method divides the visual odometry into two parts: Firstly, it extracts the image feature points as the visual landmarks, and then estimates the camera motion from these sparse feature points according to the geometry relations. Recently, the traditional RGB-D SLAM algorithms [9, 10], as well as some state-of-the-art monocular SLAM algorithms, such as PTAM (Parallel Tracking and Mapping) [12], ORB-SLAM (Oriented fast and Rotated Brief-Simultaneous Localization and Mapping) [13], have applied the feature-based method to address visual odometry. Endres et al. The work in [9] proposed a method for constructing an accurate 3D map based on an RGB-D camera. This method firstly obtains color images and depth streams from a Kinect camera. Then, it extracts and matches the ShiTomasi feature points based on the SURF descriptors. Finally, RANSAC and ICP are applied in estimating camera motion. In the experimental results, the paper compares the performance of SURF +ShiTomasi, ORB, SIFT (GPU) and SURF, and offers an outlook on how to efficiently select features in different scenarios. Mur-Artal et al. [13] proposed an ORB-SLAM algorithm for monocular, stereo and RGBD cameras. The algorithm possesses three parallel threads, including tracking, local mapping and loop closing, which collectively, effectively solve the problem of loop closures detection, relocalization and map initialization. It is capable of localizing the robot accurately in real-time and in a wide range of uncontrolled environments. The visual odometry component of this method is aimed at extracting the efficient ORB features and utilizing the ORB image feature bag of words (Bag of Word, BoW) to match the feature points. Then the RANSAC iterations are performed between consecutive frames, and using a PnP (Perspective-n-Point) algorithm to estimate the pose of camera. The feature-based visual odometry method simplifies the problem, but it has some serious limitations, such as the large amount of feature matching computation. Also, it applies less feature points which fails to represent the entire image. The scheme further suffers from false positive matches.

B. Directed Method

This method is different from the feature-based approach, skipping the image feature extraction and matching, and using the original image data - the intensity of each pixel. The premise of the direct method is based on the assumption that light intensity is unchanged at the same point of the scene in different camera perspectives. The direct method overcomes the limitations of feature-based methods, which estimate the camera motion by minimizing the photometric error, and harnesses most of the pixel information within the image. Therefore, it boasts a high accuracy and robustness in performance even in the low feature environments. The direct method can be divided into dense, semi-dense and sparse schemes according to the volume of image pixels relied upon. Newtonbe et al. [14] proposed a DTAM algorithm based on dense direct method, which utilizes every pixel information of the image to estimate the camera motion by minimizing the photometric error of all pixels and build dense 3D map. As this method has very large computation, but this scheme can only be achieved on GPU architecture. LSD-SLAM [15] is a typical semi-dense direct method, which uses pixel gray gradient points to estimate the robot motion and builds a semi-dense map. SVO [16] is a classic representative of the sparse direct method, which matches image feature points by directly registering the points with obvious gray scale gradient, and then optimizes the matching points. Finally, it uses Bundle Adjustment (BA) to estimate the camera pose. Because SVO lacks loop closures detection and global pose optimization, there is a constant accumulation of error, leading to drift.

III SPARSE DIRECT VISUAL ODOMETRY

In this section, we propose a sparse direct method for estimating the camera motion between consecutive frames based on RGB-D data acquired by the Kinect sensor. Specifically, the Kinect acquires an RGB image I_t and a depth image Z_t at time t. Using I_t, Z_t and I_{t+1}, Z_{t+1} acquired at consecutive times, the camera motion ${}^{t+1}_{t}\mathbf{T}$ is estimated by minimizing the photometric error over the image pixels.

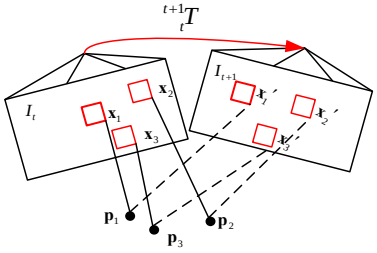

FIGURE II. THE GENERAL IDEA OF OUR APPROACH

The most important assumption of our method is that, given a 3D point $\mathbf{p}$ in the scene and the correct motion ξ, we can compute its corresponding pixel coordinates in the first image $\mathbf{x}$ and in the second image $\mathbf{x}'$. The measured intensity at these two pixel points should be identical, i.e.

$$I_t(\mathbf{x}) = I_{t+1}(\tau(\xi, \mathbf{x})) \tag{1}$$

where $\tau(\xi, \mathbf{x})$ is a warping function for calculating the pixel coordinates in the second image $\mathbf{x}'$ from the pixel coordinate in the first image $\mathbf{x}=(u,v)^T$. The following will give the specific form of this warping function.

A. Direct Motion Estimation

1) Camera model

The Kinect sensor consists of a color camera and a depth sensor. The depth sensor consists of an infrared transmitter and a receiver, as illustrated in Fig.3. The color camera obtains the RGB image and the depth sensor measures the distance information of each pixel, essentially building a depth representation of the scene. According to the imaging principle of the Kinect, as depicted in Fig.4, the color and the depth images can be merged to generate a 3D point cloud.

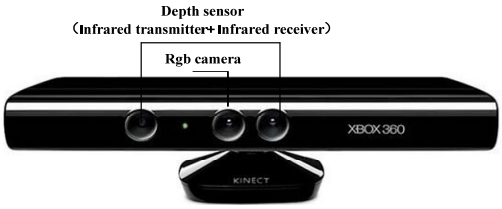

FIGURE III. THE STRUCTURE OF KINECT

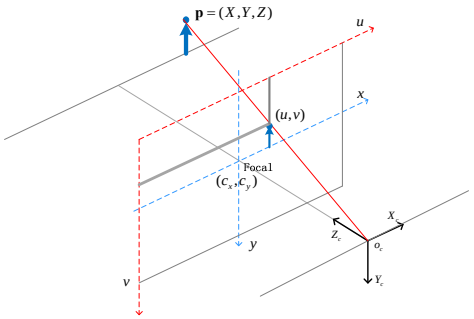

FIGURE IV. PINHOLE MODEL IMAGING PRINCIPLE

Given a 3D point $\mathbf{p}=(X,Y,Z)^T$ in the camera frame and the pixel coordinates $\mathbf{x}=(u,v)^T$ in the image plane coordinate, the following relationship holds:

$$Z\begin{bmatrix}\mathbf{x}\\1\end{bmatrix}=\begin{bmatrix}f_x & 0 & c_x\\0 & f_y & c_y\\0 & 0 & 1\end{bmatrix}\mathbf{p} \tag{2}$$

We can therefore define the projection function from the camera frame $\mathbf{p}$ to the image plane coordinate $\pi(\mathbf{p})$:

$$\mathbf{x}=\pi(\mathbf{p})=\left(\frac{Xf_x}{Z}+c_x,\frac{Yf_y}{Z}+c_y\right)^T \tag{3}$$

Given an RGB image I and a depth image Z acquired by the Kinect, we can define the inverse projection function $\pi^{-1}(\mathbf{x},Z)$ to compute the 3D point $\mathbf{p}$ in the camera frame.

$$\mathbf{p}=\pi^{-1}(\mathbf{x},Z)=\left(\frac{u-c_x}{f_x}Z/s,\frac{u-c_y}{f_y}Z/s,Z/s\right)^T \tag{4}$$

where f_x, f_y, c_x, c_y are intrinsic camera parameters, (u,v) is a pixel coordinate, (X,Y,Z) is a 3D point in the camera frame, s is the scale factor of the actual distance and the measured distance Z, here it takes 1000:1, transforming the unit of depth from millimeters to meters.

2) 3D rigid body motion

The 3D rigid body motion can be represented by the transformation matrix between the coordinate. For example, the poses of two adjacent coordinates $\{t\},\{t+1\}$ are ${}^{w}_{t}\mathbf{T}$, ${}^{w}_{t+1}\mathbf{T}$, then the transformation matrix from coordinate $\{t\}$ to coordinate $\{t+1\}$ is ${}^{t+1}_{t}T$. The relation between the transformation matrix and the two coordinates is ${}^{t+1}_{t}\mathbf{T}={}^{w}_{t+1}\mathbf{T}^{-1}\,{}^{w}_{t}\mathbf{T}$.

$${}^{t+1}_{t}\mathbf{T}=\begin{bmatrix}{}^{t+1}_{t}\mathbf{R} & {}^{t+1}_{t}\mathbf{t}\\0 & 1\end{bmatrix} \tag{5}$$

${}^{t+1}_{t}\mathbf{T}\in SE(3)$ consist of a rotation matrix ${}^{t+1}_{t}\mathbf{R}\in SO(3)$ and a displacement vector ${}^{t+1}_{t}\mathbf{t}\in\mathbb{R}^3$. Given a 3D point $\mathbf{p}^t=(X,Y,Z)^T$ in the camera coordinate $\{t\}$, we can apply a transfer function to compute its value $\mathbf{p}^{t+1}$ in the coordinate $\{t+1\}$:

$$g({}^{t+1}_{t}T,\mathbf{p}^t)=\mathbf{p}^{t+1}={}^{t+1}_{t}R\mathbf{p}^t+{}^{t+1}_{t}t \tag{6}$$

There are 12 unknown parameters in ${}^{t+1}_{t}T$, but actually there only 6 degrees of freedom. Therefore we use the exponential mapping of Lie algebra to represent the transformation matrix ${}^{t+1}_{t}\mathbf{T}=\exp(\hat{\xi})$, where $\hat{\xi}\in se(3)$, $\xi=(v_1,v_2,v_3,w_1,w_2,w_3)\in\mathbb{R}^6$, the physical meaning of v_1,v_2,v_3 are the linear speed around the axis respectively and w_1,w_2,w_3 are the angular speed. We can rewrite (6):

$$g(\xi,\mathbf{p}^t)=\mathbf{p}^{t+1}=\mathbf{D}\exp(\hat{\xi})\breve{\mathbf{p}}^t \tag{7}$$

where $\mathbf{D}=(\mathbf{I}_{3x3}\ \ \mathbf{0}_{3x1})$, $\breve{\mathbf{p}}^t=(\mathbf{p}^t\ 1)^T$.

3) Warping function

After defining the projection function and transfer function, we apply a warping function $\tau(\xi,\mathbf{x})$ to compute the corresponding pixel $\mathbf{x}'$ in the second image from the pixel $\mathbf{x}=(u,v)^T$ given a 3D rigid body motion:

$$\mathbf{x}'=\tau(\xi,\mathbf{x})=\pi(g(\xi,\pi^{-1}(\mathbf{x},Z(\mathbf{x})))) \tag{8}$$

4) Error function

With the pixel relation between two images based on warping function $\tau(\xi,\mathbf{x})$, we define the photometric error

$e_i(\xi, \mathbf{x}_i)$ for a pixel $\mathbf{x}_i$ as

$$e_i(\xi, \mathbf{x}_i) = I_{t+1}(\tau(\xi, \mathbf{x}_i)) - I_t(\mathbf{x}_i) \tag{9}$$

5) Optimization function

The basic assumption of the sparse direct method is $I_t(\mathbf{x}) = I_{t+1}(\tau(\xi, \mathbf{x}))$. In the ideal case, the photometric error $e_i(\xi, \mathbf{x}_i)$ should be zero, but the error will be distributed according to the probabilistic $p(e_i \mid \xi)$ due to the sensor noise. By assuming that the noise of all pixels is independent and identically distributed, the likelihood of the photometric error $\mathbf{e} = (e_1, e_2 \cdots e_n)$ is:

$$p(\mathbf{e} \mid \xi) = \prod_i p(e_i \mid \xi) \tag{10}$$

According to the Bayesian rules, we obtain the posteriori likelihood of camera motion ξ given the error $e = (e_1, e_2 \cdots e_n)$,

$$p(\xi \mid \mathbf{e}) = \frac{p(\mathbf{e} \mid \xi)\, p(\xi)}{p(\mathbf{e})} \tag{11}$$

where $p(\xi)$ is the prior distribution over camera motions.

The camera motion ${}^{t+1}_{t}\mathbf{T} = \exp(\hat{\xi}^*)$ can be estimated by maximizing the posteriori likelihood,

$$\xi^* = \arg\max_{\xi} p(\xi \mid \mathbf{e}) \tag{12}$$

Combined with (10) and (11) to rewrite (12):

$$\xi^* = \arg\max_{\xi} \prod_i p(e_i \mid \xi) p(\xi) \tag{13}$$

By minimizing the negative log-likelihood we can equivalently write

$$\xi^* = \arg\min_{\xi} -\sum_i \log p(e_i \mid \xi) - \log p(\xi) \tag{14}$$

Due to the fact that prior distributions over camera motions require other sensors such as IMU, we omit this part in this paper. Assume that the photometric error probability distribution is a Gaussian distribution with mean zero and variance σ_i^2, i.e. $e_i \sim \mathrm{N}(0, \sigma_i^2)$, then

$$p(e_i \mid \xi) = \left(2\pi\sigma_i^2\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\frac{e_i^2}{\sigma_i^2}\right) \tag{15}$$

According to (15), we rewrite (14)

$$\begin{aligned} \xi^* &= \arg\min_{\xi} -\left(a + \sum_i -w(e_i) e_i^2\right) \\ &\Rightarrow \arg\min_{\xi} \sum_i w(e_i) e_i^2 \end{aligned} \tag{16}$$

where $w(e_i)$ is the weight factor of e_i, in order to simplify the problem, let $w(e_i) = 1/2$. Then we derive the final expression of the optimization function:

$$\xi^* = \arg\min_{\xi} \frac{1}{2} \sum_i e_i(\xi, \mathbf{x}_i)^2 \tag{17}$$

6) Problem solution

The camera motion can be estimated by solving the nonlinear minimization problem (17) iteratively.

$${}^{t+1}_{t}\mathbf{T} = \exp(\xi^*) \tag{18}$$

(1) Initialization: Suppose the initial value of ξ is ξ^0, then the value of the goal function is

$$h(\xi^0) = \frac{1}{2} \sum_i e_i(\xi^0, \mathbf{x}_i)^2 \tag{19}$$

(2) Perturbation and Linearization: Adding the disturbance $\Delta\xi$ and computing the first order Taylor approximation of $e_i(\xi^0 \oplus \Delta\xi, \mathbf{x}_i)$. The appendix provides a brief proof of the first order Taylor approximation of $e_i(\xi^0 \oplus \Delta\xi, \mathbf{x}_i)$ in the manifold.

$$\begin{aligned} e_i(\xi^0 \oplus \Delta\xi, \mathbf{x}_i) &= I_{t+1}(\tau(\xi_0, \mathbf{x}_i)) - I_t(\pi(g(\Delta\xi, \mathbf{p}_i))) \\ &= e_i(\xi^0, \mathbf{x}_i) + J_i \Delta\xi \end{aligned} \tag{20}$$

After linearizing the error $e_i(\xi^0 \oplus \Delta\xi, \mathbf{x}_i)$, we can rewrite the goal function $h(\xi^0 \oplus \Delta\xi)$. $e_i(\xi^0) \triangleq e_i(\xi^0, \mathbf{x}_i)$

$$\begin{aligned} h(\xi^0 \oplus \Delta\xi) &= \sum_i \frac{1}{2}(e_i(\xi^0) + J_i \Delta\xi)^2 \\ &= \sum_i \frac{1}{2}(e_i(\xi^0) + J_i \Delta\xi)^T (e_i(\xi^0) + J_i \Delta\xi) \\ &= \sum_i \frac{1}{2}(e_i(\xi^0)^2 + 2e_i(\xi^0)^T J_i \Delta\xi + \Delta\xi^T J_i^T J_i \Delta\xi) \end{aligned} \tag{21}$$

When the derivative of $h(\xi^0 \oplus \Delta\xi)$ with respect to $\Delta\xi$ is zero, $h(\xi^0 \oplus \Delta\xi)$ is minimum.

$$\frac{\partial h(\xi^0 \oplus \Delta\xi)}{\partial \Delta\xi} = \sum_i J_i^T e_i(\xi^0) + \sum_i J_i^T J_i \Delta\xi = 0 \tag{22}$$

Finally we obtain the normal equations

$$\sum_i J_i^T J_i \Delta\xi = -\sum_i J_i^T e_i(\xi^0) \tag{23}$$

Get the increment $\Delta\xi$ by solving the normal equations.

(3) Update:

$$\xi^1 \leftarrow \log(\exp(\hat{\xi}^0)\exp(\Delta\hat{\xi})^{-1}) \tag{24}$$

According to (24), we update the camera motion and iterate over steps (1)~(3) until reaching the maximum number of iterations or the overall photometric error increases.

The calculation of the Jacobin matrix J_i:

$$\begin{aligned} J_i &= \frac{\partial e_i(\xi^0 \oplus \Delta\xi, \mathbf{x}_i)}{\partial \Delta\xi} \\ &= -\frac{\partial I_t(\pi(g(\Delta\xi, \mathbf{p}_i)))}{\partial \Delta\xi} \\ &= -\frac{\partial I_t(\mathbf{x})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}=\mathbf{x}_i} \cdot \frac{\partial \pi(\mathbf{p})}{\partial \mathbf{p}}\bigg|_{\mathbf{p}=\mathbf{p}_i} \cdot \frac{\partial g(\Delta\xi, \mathbf{p}_i)}{\partial \Delta\xi}\bigg|_{\Delta\xi=0} \end{aligned} \tag{25}$$

We use the information of 4X4 image patch near the pixel to compute the pixel gradient:

$$\frac{\partial I_t(\mathbf{x})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}=\mathbf{x}_i} = \begin{bmatrix} \frac{\partial I_t(\mathbf{x})}{\partial u} & \frac{\partial I_t(\mathbf{x})}{\partial v} \end{bmatrix} \tag{26}$$

where

$$\frac{\partial I_t(\mathbf{x})}{\partial u} = \sum_{m=1}^{2}\sum_{n=0}^{1} \frac{I_t(u_i+m, v_i+n) - I_t(u_i+m-2, v_i+n)}{2} \tag{27}$$

$$\frac{\partial I_t(\mathbf{x})}{\partial v} = \sum_{m=0}^{1}\sum_{n=1}^{2} \frac{I_t(u_i+m, v_i+n) - I_t(u_i+m, v_i+n-2)}{2} \tag{28}$$

In this paper, the method of solving the nonlinear minimization problem has the advantages of computational efficiency compared with the traditional direct method [17, 18]. The main differences are in steps (2) and (3), which are mainly reflected in the calculation of Jacobin matrix. The equation (25) shows that J_i is only related to reference frame $I_t(\mathbf{x})$ and $\mathbf{p}_i$. Both of them are not changed during the iteration, so that the Jacobin matrix J_i only needs to be computed once which improves the efficiency of solving the nonlinear minimization problem.

The first order Taylor approximation of photometric error $e_i(\xi^0 \oplus \Delta\xi, \mathbf{x}_i)$ is only effective for the small camera motion change $\Delta\xi$. This paper uses the image pyramid method to deal with the large motion situation. Firstly, the I_t and I_{t+1} image pyramids are established according to the principle that the width and height of the L-th layer are the half of the $L-1$-th layer. Then using direct method to estimate the camera motion ξ, and using this motion ξ to initialize the next image pyramid.

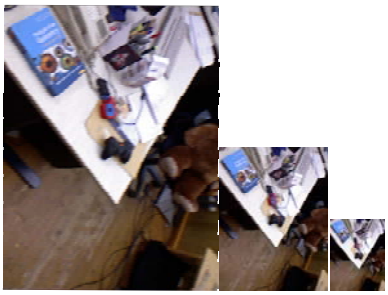

FIGURE V. THE IMAGE PYRAMIDS

B. *Sparse Analysis*

The procedure of using the direct method to estimate the camera motion shows that Jacobin matrix J_i is closely related to the pixel gradient. This signifies that if the pixel gradient is small, the contribution to the problem solving is negligible. Therefore, we do not use the dense method similar to DTAM [14] or LSD-SLAM [15] which calculates the photometric error of most or even all pixels, but rather, we extract the FAST feature points. Because FAST feature points have obvious gray scale gradients, computational efficiency and significant real-time performance improvements are achieved.

In order to avoid the coincidence of FAST feature points, we firstly grid the image and then only extract one feature point in each grid. In addition, in order to improve the robustness, we use the neighborhood information of the sparse feature points by adding the photometric errors of the 4x4 image patch near the feature point to the optimization function. Fig. 6 shows the 4x4 image patch:

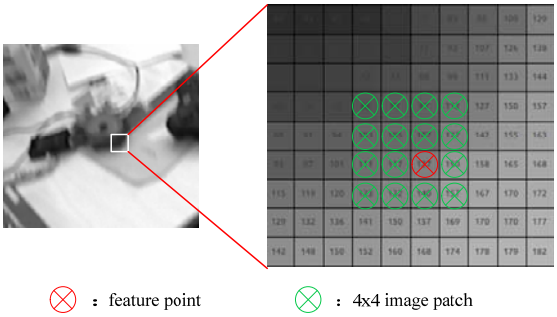

FIGURE VI. THE 4X4 IMAGE PATCH NEAR FEATURE POINT

When calculating the intensity $I_{t+1}(\tau(\xi, \mathbf{x}_i))$, the subtrahend of $e_i(\xi, \mathbf{x}_i) = I_{t+1}(\tau(\xi, \mathbf{x}_i)) - I_t(\mathbf{x}_i)$, we use bilinear interpolation method to compute it for $\mathbf{x}_i' = \tau(\xi, \mathbf{x}_i) = (u, v)^T$ is a float type vector.

$$\begin{aligned} I_{t+1}((u,v)^T) = {} & (1-uu)*(1-vv)*I_{t+1}((su,sv)^T)+ \\ & uu*(1-vv)*I_{t+1}((su+1,sv)^T)+ \\ & (1-uu)*vv*I_{t+1}((su,sv+1)^T)+ \\ & uu*vv*I_{t+1}((su+1,sv+1)^T) \end{aligned} \tag{29}$$

where $u=su+uu, v=sv+vv$, su, sv are integers, uu, vv are float numbers between 0 and 1.

IV LOOP DETECTION AND POSE OPTIMIZATION

In the sparse direct visual odometry, the current camera motion estimation error will be accumulated and carried forward into the next frame. There is a constant accumulation of error, leading to an estimated camera pose with significant drift. In order to reduce the cumulative error, our approach detects loop closures based on keyframes to build up pose graphs and then utilize the g2o framework to optimize the camera pose globally.

A. Keyframe Selection

The basic element of the proposed method is the keyframe, which contains the following important data: (1) RGB and depth image acquired by Kinect; (2) Pose of robot presented by Lie algebra ${}_w^t\mathbf{T}=\exp(\xi)$;(3)The measurements of feature points and its Jacobin Matrix J_i.

The number of keyframes can seriously affect the performance of the system. It mainly manifests in the following ways: (1) If there are too many redundant frames, then it will increase the loop closures detection and pose global optimization of the calculation when there are more keyframes; (2) If the key frames selected is not sufficient in number, there is not a sufficient coincidence area between the frames, then the sparse direct motion estimation will have a large error leading to the failure of positioning. Therefore, the keyframe selection method is an important problem of RGB-D SLAM systems. The keyframe selection method in spatial domain is introduced to eliminate redundant frames. We use the proposed method to estimate motion between the last keyframe and the current frame, if the estimated motion is between the minimum motion threshold $\phi_{\min}$ and the maximum motion threshold $\phi_{\max}$, then this frame will be selected as a new keyframe. Since the motion is represented as a transform matrix, we use the weighted sum of norm of the relative Euler angle and the translation vector as a measurement:

$$\phi=\left\|\mathbf{w}_1\cdot(\Delta x,\Delta y,\Delta z)\right\|_2+w_2\left\|(\alpha,\beta,\gamma)\right\|_2 \tag{30}$$

where $(\Delta x,\Delta y,\Delta z)$ is the translation vector and (α,β,γ) is the relative Euler angle spin around the x,y,z axis respectively. It is important to choose the weight factors $\mathbf{w}_1$ and w_2. Since the FOV of the Kinect is limited (the horizontal and vertical direction are 57º, 43º), so the scene change caused by the camera rotation movement is much larger than the translational motion, which will seriously affect the accuracy of motion estimation. In other words, the motion estimation is more sensitive to rotation, so the frame should be selected as a new keyframe when it has a small rotation or a larger translation. In this paper, we choose $\mathbf{w}_1$=(0.6,0.7,0.7) , w_2=1.3 , $\phi_{\min}$=0.25 and $\phi_{\max}$=0.5 as our keyframe selection parameters by experience.

B. Loop Closures Detection

Loop closures detection defines the task of detecting whether the position of the current frame exists in the past keyframes. Once a loop is found, an edge will be added to the existed pose graph. Finally, the g2o frame is applied to optimize the pose graph globally to reduce the accumulation of drift. However, the error of the RGB-D SLAM system will be larger if the loop is a false positive. The loop must therefore be accurately handled.

There are several approaches for detection of loop closures. The simplest strategy is a linear search over all the past keyframes. With such a scheme the computational load increases significantly as the number of keyframes grows. It is therefore important to prune the search space efficiently and only match against the most likely candidates. In our approach, we rely upon the metrical nearest neighbor search method [9], a loop closure candidate will be searched in a sphere with a predefined radius r around the current keyframe position. We use the proposed method to estimate the transform matrix between the current keyframe and the keyframes in the sphere. A loop closure candidate will be chosen when the photometric error between two keyframes is less than the error threshold e_θ.

$$\sum_i I_{t+1}(\tau(\xi,\mathbf{x}_i))-I_t(\mathbf{x}_i)<e_\theta \tag{31}$$

Finally, we choose the candidate with least photometric error as the final loop and add an edge to connect these two keyframes.

C. Pose Optimization

The pose graph can represent the relationship between the keyframes. In our approach, the vertices of the graph are denoted by the pose of each keyframe ${}_i^w\mathbf{X}$ ${}_j^w\mathbf{X}$, while the constraint edge from vertex i to vertex j is the transform matrix ${}_j^i\mathbf{T}$ which is computed by the proposed method. Finally, the poses of the keyframes are optimized by minimizing the following cost function:

$$\underset{{}_i^w\mathbf{X},{}_j^w\mathbf{X}}{\operatorname{argmin}}\sum_{i,j}(\mathbf{e}_{i,j}^T\Delta_{i,j}\mathbf{e}_{i,j}) \tag{32}$$

$$\mathbf{e}_{i,j}=\log_{se(3)}({}_j^i\mathbf{T}^{-1}\,{}_i^w\mathbf{X}^{-1}\,{}_j^w\mathbf{X}) \tag{33}$$

where $\log_{se(3)}$ maps pose error to a six dimensional Euclidean space $\mathbb{R}^6$, $\mathbf{\Delta}_{i,j}$ is the information matrix of the edge from vertex i to vertex j.

V EXPERIMENTS

In order to evaluate the performance of the proposed method, two experiments are presented: (1) Open dataset experiment, where we perform the localization tests on an RGB-D open dataset that is provided by Zurich University; (2) Real-world experiments, where we test performance by using the Turlebot2 in the laboratory corridor. The following two sections are the experimental processes and the results of the two experiments.

A. Open Dataset

This section mainly analyzes the performance of the algorithm from the accuracy and positioning speed. The dataset contains several color and depth images acquired by the depth camera, the ground truth pose in every timestamp of the depth camera obtained by the high-accuracy motion capture system and the intrinsic camera parameter.

This dataset is acquired by a depth camera that points towards the ground in a fixed height. It contains 186 images and the total motion distance of the camera is 29.6 meters. We use the proposed method to estimate the camera trajectory. Fig. 7 presents a comparison between ground truth trajectory and estimated trajectory.

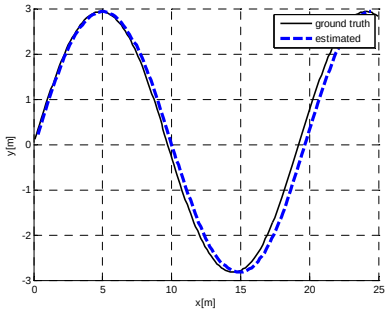

FIGURE VII. THE COMPARISON BETWEEN ESTIMATED TRAJECTORY (BLUE DASH) AND GROUND TRUTH TRAJECTORY (**BLACK** SOLID)

The experimental result show that: (1) the method can locate the robot position accurately in a large range. In the range of 29.6 meters, the RMSE of the robot is 0.31 meters. (2) the proposed method can meet real-time requirements. It takes 20 milliseconds to process every frame on average. The localization speed up to 50 fps.

B. Real-world Environments

In order to analyze the effect in the practical robot localization further, we test the proposed method by localization the Turtlebot2 robot (an open source robot) in the lab corridor. It is made up of a KOBUKI mobile pedestal，a kinect depth camera and a laptop. Fig.8 shows the Turtlebot2 robot. The proposed scheme is implemented in C++ and operates on Ubuntu operating system with I5-3210M CPU and 8GB RAM.

FIGURE VIII. TURTLEBOT2 ROBOT

FIGURE IX. THREE SECTIONS OF THE CORRIDOR

The laboratory corridor consists of three straight roads. Fig.9 shows that there are some similar objects (such as doors, tiles etc.). Meanwhile the tiles have less corner features which will bring huge challenges to feature extracted and detect loop closure. Turtlebot2 robot is equipped with a kinect, and the robot frame is same with the kinect frame. During the experiment, we use the XBOX joystick to control the robot with 0.3 m/s horizontal motion. The proposed method can estimate the trajectory in real-time. Fig. 10 shows the estimated horizontal 2D trajectory of the Turtlebot2 robot.

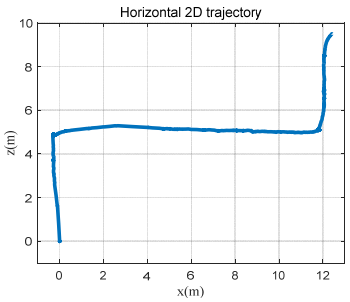

FIGURE X. HORIZONTAL 2D TRAJECTORY OF THE ROBOT

During this experiment, we failed to provide the ground truth pose in every timestamp of the robot for lack of motion capture systems. In order to analyze the performance in the localization accuracy, we select six positions in the three straight roads and apply them as measurement points. The true positions (x, z) of these six measurement points can be measured by a tape. Finally, the localization can be evaluated by the Root Mean Square Error (RMSE) of these six measurement points. $RMSE = \sqrt{(\sum_{i=1}^{n}(t_i - \bar{t}_i)^2)/n}$, where $\bar{t}_i$, t is the estimated position and true position respectively, n=6.Table 1 shows the estimated position, true position and RMSE of the six measurement points. The estimated trajectory of the robot possesses some accumulated errors during it motion in the laboratory corridor. The length of the robot motion is 25 meters and the RMSE of these measurement points is 0.73 meters, which represents 2.9%.

TABLE I. RESULTS ON POSITION ESTIMATION OF THE MEASURING POINTS

Measurement points	***True positions(m)***	***Estimated positions(m)***	***Error's square(m^2)***
1	(-0.3,4.96)	(-0.31,4.93)	0.001
2	(4.12,5.61)	(3.92,5.40)	0.29
3	(8.13,5.86)	(7.4,5.2)	0.97
4	(10.21,5.88)	(9.3,5.3)	1.079
5	(12.29,9.34)	(11.8,8.6)	0.79
6	(12.29,10.6)	(12.4,9.5)	1.22
RMSE(m)	0.73		

In the experiment, the method processes 944 frames in total and the average processing time is 22 milliseconds. This suggests that the localization speed is up to 45 fps which satisfies real-time requirements.

VI CONCLUSION

In order to address the problems associated with feature-based RGBD-SLAM techniques, including the high computational constraints associated with image feature matching, this paper proposes an RGB-D SLAM algorithm that firstly detects the FAST image feature points and estimates the camera pose by minimizing the photometric error between two images. Secondly, the key frames are extracted based on the method on the spatial domain. That is, the frame will be selected as a key frame when the robot's motion exceeds a certain threshold. Also, the nearest neighbor method based on spatial domain processing is used to detect loop closures. Finally, the robot pose is optimized by using the g2o framework to optimize the pose graph.

To evaluate the performance of the proposed method, we present experiments on the open dataset and a physical robot in real-world environments respectively. The results show that the method can locate the robot position accurately large ranges. In the range of 25 meters, the RMSE of the robot is 0.73 meters, representing 2.9%. Additionally, the localization speed reaches up to 45 fps. The RMSE is capable of reaching 1% in the range of 29.6 meters when the camera motion towards the ground is of a fixed height. However, proposed method still exhibits some drawbacks. Its performance is degraded in scenarios where features are significantly low.

APPENDIX

In the following we will prove the linearization of $e_i(\xi^0 \oplus \Delta\xi, \mathbf{x}_i)$ in (20), which function variable is not linear variation. So we need to compute the first order Taylor approximation of $e_i(\xi^0 \oplus \Delta\xi, \mathbf{x}_i)$ in the manifold.

Given

$$e_i(\xi^0, \mathbf{x}_i) = I_{t+1}(\tau(\xi_0, \mathbf{x}_i)) - I_t(\pi(g(0, \mathbf{p}_i))) \tag{34}$$

$$\exp(\Delta\hat{\xi}) \approx 1 + \Delta\hat{\xi} \tag{35}$$

Then

$$\begin{aligned} e_i(\xi^0 \oplus \Delta\xi, \mathbf{x}_i) &= I_{t+1}(\tau(\xi_0, \mathbf{x}_i)) - I_t(\pi(g(\Delta\xi, \mathbf{p}_i))) \\ &= I_{t+1}(\tau(\xi_0, \mathbf{x}_i)) - I_t(\pi(\mathbf{D}\exp(\Delta\hat{\xi})\breve{\mathbf{p}}_i)) \\ &\approx I_{t+1}(\tau(\xi_0, \mathbf{x}_i)) - I_t(\pi(\mathbf{D}(1+\Delta\hat{\xi})\breve{\mathbf{p}}_i)) \\ &= I_{t+1}(\tau(\xi_0, \mathbf{x}_i)) - I_t(\pi(\mathbf{D}\breve{\mathbf{p}}_i + \mathbf{D}\Delta\hat{\xi}\breve{\mathbf{p}}_i)) \\ &\approx I_{t+1}(\tau(\xi_0, \mathbf{x}_i)) - I_t(\pi(\mathbf{D}\breve{\mathbf{p}}_i)) - \left.\frac{\partial I_t(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\pi(\mathbf{D}\breve{\mathbf{p}}_i)} \left.\frac{\partial \pi(\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{D}\breve{\mathbf{p}}_i} \mathbf{D}\Delta\hat{\xi}\breve{\mathbf{p}}_i \\ &= e_i(\xi^0, \mathbf{x}_i) - \left.\frac{\partial I_t(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_i} \left.\frac{\partial \pi(\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_i} \cdot \mathbf{D}\breve{\mathbf{p}}_i^{\odot}\Delta\xi \end{aligned} \tag{36}$$

where $\breve{\mathbf{p}}_i^{\odot}$ is defined by the following:

$$\breve{\mathbf{p}}_i^{\odot} = \begin{bmatrix} \mathbf{p}_i \\ 1 \end{bmatrix}^{\odot} = \begin{bmatrix} I_{3\times 3} & -\hat{\mathbf{p}}_i \\ \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} \end{bmatrix} \tag{37}$$

Because of

$$\frac{\partial e_i(\xi^0 \oplus \Delta\xi, \mathbf{x}_i)}{\partial \Delta\xi} = J_i = -\left.\frac{\partial I_t(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_i} \left.\frac{\partial \pi(\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_i} \cdot \mathbf{D}\breve{\mathbf{p}}_i^{\odot}\Delta\xi \tag{38}$$

We obtain

$$e_i(\xi^0 \oplus \Delta\xi, \mathbf{x}_i) = e_i(\xi^0, \mathbf{x}_i) + J_i\Delta\xi \tag{39}$$

ACKNOWLEDGMENT

The work in this paper was supported in part by the Guangdong Provincial Science and Technology Project under Grant No. x2zdB2152380 and the National Natural Science Foundation of China under Grant No.x2zdB5150710.

REFERENCES

[1] Davide D et al. Indoor Tracking Theory, Methods, and Technologies2015a [J]. IEEE Transaction on Vehicular Technology, 64(4): 1263-1278 (2015).

[2] Choi H, Yang K W, Kim E. Simultaneous global localization and mapping [J]. IEEE/ASME Transactions on Mechatronics, 19(4):1160–1170 (2014).

[3] Chen Z, Samarabandu J, Rodrigo R. Recent advances in simultaneous localization and map building using computer vision [J]. Advanced Robotics, 21(3-4): 233–265 (2007).

[4] J. Shi, C. Tomasi, Good features to track, in: Computer Vision and Pattern Recognition [C], 1994. Proceedings CVPR' 94, IEEE Computer Society Conference on, IEEE, 1994, pp. 593–600 (1994).

[5] E. Rosten, T. Drummond, Machine learning for high-speed corner detection, in: Computer Vision–ECCV 2006, Springer, pp. 430–443 (2006).

[6] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Computer Vision. 60 (2): 91–110 (2004).

[7] H. Bay, T. Tuytelaars, L. Van Gool. SURF: speeded up robust features, in: Computer Vision–ECCV 2006, Springer, pp. 404–417 (2006).

[8] Rublee E, Rabaud V, Konolige K, et al. ORB: an Efficient Alternative to SIFT or SURF[C]//Computer Vision （ICCV）, IEEE International Conference on. IEEE, 2011: 2564-2571 (2011).

[9] Henry P, Krainin M, Herbst E, et al. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments[C]//12th International Symposium on Experimental Robotics. Berlin, Germany: Springer, 2014: 477-491.

[10] Endres F, Hess J, Engelhard N, et al. An evaluation of the RGB-D SLAM system[C]//IEEE International Conference on Robotics and Automation. Piscataway, USA: IEEE, 2012: 1691-1696.

[11] Z. Chen, J. Samarabandu, R. Rodrigo, Recent advances in simultaneous localization and map-building using computer vision, Adv. Robot. 21 (3–4) (2007) 233–265

[12] Klein G, Murray D. Parallel tracking and mapping for small AR workspaces[C]//Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on. IEEE, 2007: 225-234.

[13] Mur-Artal, et al. ORB-SLAM: A Versatile and Accurate Monocular SLAM System [J]. IEEE Transaction on Robotics. 2015, 31(5):1147-1163.

[14] Newcombe R A, Lovegrove S J, Davison A J. DTAM: Dense tracking and mapping in real-time[C]//IEEE International Conference on Computer Vision. Piscataway, USA: IEEE, 2011: 2320-2327.

[15] Engel J, Schops T, Cremers D. LSD-SLAM: Large-scale direct monocular SLAM[C]//13th European Conference on Computer Vision. Berlin, Germany: Springer, 2014: 834-849.

[16] Forster C, Pizzoli M, Scaramuzza D. SVO: Fast semi-direct monocular visual odometry[C]//IEEE International Conference on Robotics and Automation. Piscataway, USA: IEEE, 2014: 15-22.

[17] Kerl C, Sturm J, Cremers D. Robust odometry estimation for RGB-D cameras[C]//Robotics and Automation (ICRA), 2013 IEEE International Conference on. IEEE, 2013: 3748-3754.

[18] Kerl C, Sturm J, Cremers D. Dense visual SLAM for RGB-D cameras[C]//2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2013: 2100-2106.