PairCon-SLAM: Distributed, Online, and Real-Time RGBD-SLAM in Large Scenarios

Donglin Zhu, Guanghui Xu, Xiaoting Wang, Xiaogang Liu, and Dewei Tian

***Abstract*— This article proposed a PairCon-simultaneous localization and mapping algorithm to construct dense maps of large-scale scenarios in real time, which is operated in an integral platform containing two personal computers (PCs). This platform can operate the mapping thread independently in a PC to guarantee sufficient memory resources. In this context, the synchronous visualization of 3-D maps can be achieved in such a platform. In contrast, traditional algorithms are limited by the short operation distance, under the requirement of simultaneous data collection and 3-D maps construction in a PC. In addition, the memory resource provided by a single PC is limited, which restricts the constructing scale of maps. Thus, we use a separate PC to construct maps independently to relieve the distance constraint and exploit the socket method to conduct the transmission of data with the point cloud. Meanwhile, we introduce the ReBlur algorithm into the semi-direct method to reduce the error accumulation of the odometer in the tracking thread, which improves the robustness performance. In addition, the method combining the memory management and DBow2 algorithm is adopted to improve the accuracy of loop detection. In the considered system, the quality of maps and the performance of the odometer are evaluated by ICL-NUIM and the datasets, such as TUM, DIODE, and so on, respectively. Finally, under the simulation environment of AirSim and Gazebo, we construct maps based on the image data of other scenarios, which is used to show the quality of the construction.**

***Index Terms*— Big-scale-scene, dataset, mapping, RGBD-simultaneous localization and mapping (SLAM), socket.**

I. Introduction

THE 3-D reconstruction is becoming an active area of research, due to the ready availability of depth cameras, such as the Microsoft Kinect and Intel RealSense. At present, the vast majority of dense maps are studies of small indoor scenes, and large scenes are ignored. Processing of large scenes requires longer working hours for the hardware, larger memory resources, longer time to detect loops, and increased cumulative error of the odometer, compared with smaller scenes. In most simultaneous localization and mapping (SLAM) algorithms for small scenarios, the simulation device used is a conventional computer or embedded device. For large scenarios, one of the challenges of relying on a single device for data acquisition and map construction is the size of the memory space required. In large scenarios, devices are used to collect data for long periods of time, but portable devices have small memories. A large amount of memory space is consumed by the collected data, and the space available for map construction is therefore considerably reduced. Previous studies have used a combination of portable devices and UAVs in large scenarios. Trujillo *et al.* [1] and Artieda *et al.* [2] used monocular SLAM to localize UAVs and construct sparse maps, but dense maps could not be built. Pan *et al.* [3] focused on the construction of maps of roads while overlooking the considerable information available about the outdoor circumstances. Caballero *et al.* [4] used a manned aircraft for outdoor positioning, which could construct dense maps, but at a high cost. Whelan *et al.* [5] used vehicle-mounted SLAM to construct dense maps of the neighborhood, but it was inflexible, and objects in high places and off-street could not be mapped. Analysis of the current state of research shows that most SLAM in large scenarios today is used for localization, and very little is used for the construction of dense maps, because of the memory issues, and SLAM is used mostly in the form of car-mounted maps.

A wide variety of objects are involved in large scenarios, and the corresponding image data are large, resulting in long processing times and large amounts of point cloud data (PCD) produced at each stage of processing. If an experimental platform with two personal computers (PCs) is used, the most important issue is the best way in which to ensure the transfer of the point cloud between the PCs. The odometry, loop detection, and map constructing stages need to be designed to suit this scenario while ensuring adequate real-time performance. In RGB-D and stereo-based SLAM systems, direct methods based on photometric error minimization are becoming increasingly popular. The odometer is improved by the use of the semi-direct method [6]. The advantage of this method is that it does not require feature extraction in every frame, so the processing speed is improved, and the accuracy is increased through subpixel feature correspondence.

Manuscript received May 17, 2021; revised September 11, 2021; accepted September 14, 2021. Date of publication November 8, 2021; date of current version November 11, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61971474 and Grant 51507188 and in part by China Postdoctoral Science Foundation Funded Project under Grant 2019T120071. The Associate Editor coordinating the review process was Bardia Yousefi. *(Corresponding authors: Guanghui Xu; Xiaoting Wang.)*

Donglin Zhu and Guanghui Xu are with the School of Communications Engineering, Army Engineering University of PLA, Nanjing 210007, China (e-mail: zdl_edu@sina.com; xugh_xs@163.com).

Xiaoting Wang is with the State Key Laboratory of Estuarine and Coastal Research, East China Normal University, Shanghai 200241, China (e-mail: xiaotingwang_1@163.com).

Xiaogang Liu is with the School of Electronic Information Engineering, Inner Mongolia University, Hohhot 010021, China (e-mail: huliqujing@hotmail.com).

Dewei Tian is with the School of Electronic and Optical Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210023, China (e-mail: 1220024515@njupt.edu.cn).

Digital Object Identifier 10.1109/TIM.2021.3116288

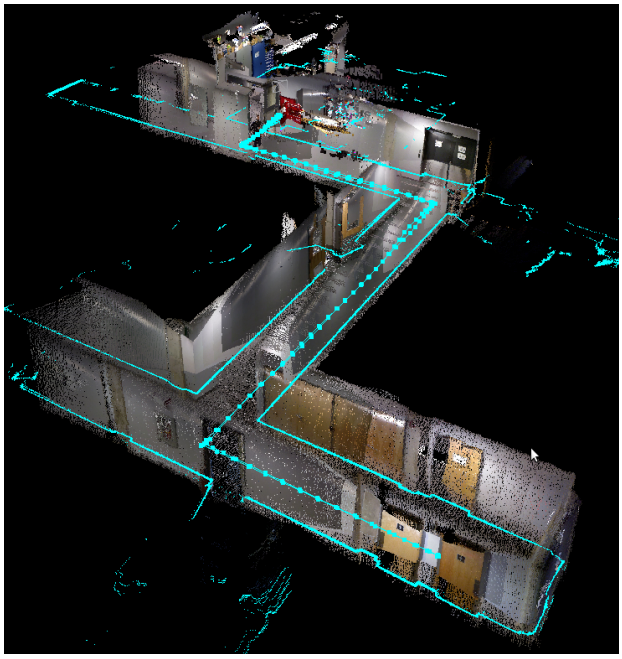

Fig. 1. Left corner shows a mapping result of corridor scene in the teaching building.

Loop detection is improved by using the memory management method [7], so the system can deal with the large-scale and long-term operation, reducing the time required to search through previously visited locations.

To solve this problem, the data acquisition and processing is run on one terminal, while the map construction thread runs independently on another terminal. We propose a method for transferring point clouds between PCs using a small local area network (LAN), which allows the transfer of the current frame of point clouds to be completed and stitched into the map, and then the next frame of point clouds can be implemented. The results of the mapping of the corridor scene in the teaching building are shown in Fig. 1. Two simulation environments were provided, in which SLAM was performed. The robot was successfully able to localize itself and map the 3-D world. The first environment is an outdoor scene from AirSim [8], and the second environment is a Gazebo scene called Software Museum (Fig. 9). An example of the mapping results of soccer_field in the first scene is shown in Fig. 10. The main contributions of this article are as follows.

1) A socket method is used to transmit PCD.
2) A new keyframe selection method is described.
3) A semi-direct method is used, combining detection and restoration of blurred images.
4) The memory management method is combined with DBoW2 and Dnorm.

II. Multimedia Material

These links point to the locations of the various datasets used by the system during testing, and all the constructing results are made into videos for display. All datasets used and reconstruction results are publicly available.

Supplemental Video:
https://www.bilibili.com/video/BV1w84y1c7bh
Datasets:
http://www.doc.ic.ac.uk/ ahanda/VaFRIC/iclnuim.html
https://vision.in.tum.de/data/datasets
http://graphics.stanford.edu/projects/objectsensing/#data
https://www.microsoft.com/en-us/research/project/rgb-d-dataset-7-scenes

III. Related Work

There have been hundreds of studies into indoor-based RGB-D systems. We restrict our attention to research involving larger scenarios. Li *et al.* [9] used a lightweight quadrotor equipped with a monocular camera to obtain the RGB images of the scene but found it necessary to estimate the corresponding inverse depth image when constructing the map, resulting in a large value of errors of depth. Visual SLAM (VSLAM) has been proposed as a promising technology in the fields of machine sensing and navigation and has been used in the National Aeronautics and Space Administration (NASA) Mars exploration program and in China's Chang'e-3 and Chang'e-4 lunar exploration programs [10]–[12].

RGBDTAM SLAM (RGBDTAM) [13] improves positioning accuracy by using multiple view geometry and minimizing photometric errors. However, the accuracy of the depth value produced by triangulation is not high, and the assumption of invariance of luminosity must be satisfied. ElasticFusion [14] uses the iterative closest point (ICP) [15] algorithm to estimate pose, and continuously optimize and reconstructs its map to improve the accuracy of map reconstruction and pose estimation. The ICP algorithm is suitable for feature points with known depth but has a large pose estimation error for feature points with large errors. Cheng *et al.* [16] proposed a new improved dynamic SLAM based on ORB-SLAM2, using the YOLO model and Bayesian filter posterior estimation to identify dynamic features. Wang *et al.* [17] and Bahraini *et al.* [18] use the improved random sampling consensus algorithm to track moving objects and realize motion estimation in a dynamic environment. The above-mentioned method loses tracking of static objects in the scene with many moving objects or occluded static objects. Based on the SqueezeSeg method, RangeNet++ method [19] adds the category of dataset [20] annotation and uses k-nearest neighbor (k-NN) clustering to replace the image smoothing method in the SqueezeSeg method. Kim *et al.* [21] proposed a method for interpolation between continuous positions and poses which combines shutter camera and IMU and established a measurement model to cope with the problem of time delay. He *et al.* [22] propose a new framework of pose estimation, which can build maps using a laser radar and multiple sensors in partially GNSS-denied environments.

Loop detection first needs image description. Traditional methods describe the image using artificial features, which are generally divided into local features and global features. Local features describe the whole image, by extracting a large number of feature points. In order to speed up image matching, the descriptor information of a large number of feature points is compared using BOW, FV [23], VLAD [24],

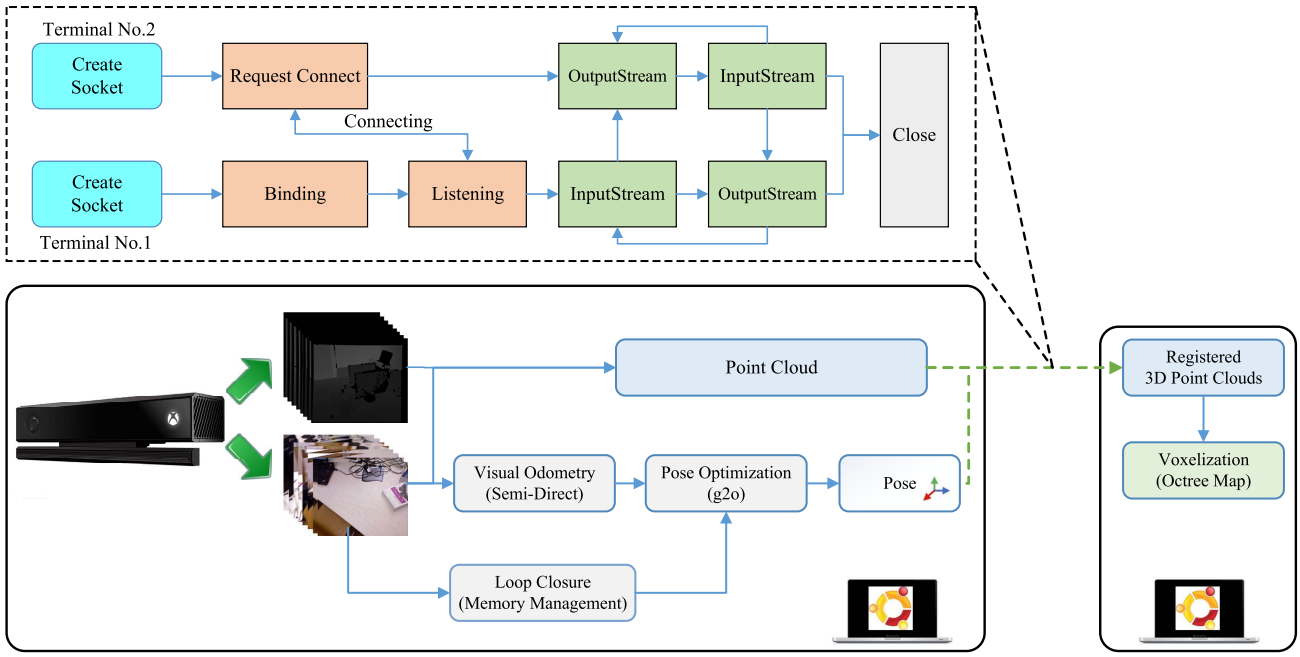

Fig. 2. Structure of our PairCon-SLAM.

or other methods. The bag of words (BoWs) [25] algorithm regards image features as words and uses a dictionary to find corresponding words, an approach that is widely used in loop detection. The dictionary used in the model is trained offline using the K-means algorithm, and the weights of the feature vector are expressed as binary values, so the loading time is short. Moreno *et al.* [26] developed a point-line feature-based slam method for stereo, which fully considered the point and line feature in the loop detection module. LDSO, an algorithm proposed by Gao *et al.* [27] integrates loop and global map optimization into the DSO algorithm and calculates ORB descriptor to establish a BoW model for loop detection. Global descriptors, such as generalized search trees (GISTs) and histogram of oriented gradient (HOG) features, are used to process the whole image. This approach is robust when the illumination changes and the local features are not obvious, but when the viewing angle changes, the global features generated by different images in the same environment may be quite different. Hou *et al.* [28] used principal components analysis (PCA) to reduce the dimensionality of CNN features, speeding up image retrieval.

Network communication based on sockets in a Linux environment has also become a research focus. Sockets are used not only for communication between single processes but also for network communications. At present, a large number of literature use socket algorithm to explore how to transmit or detect non-PCD data. Yubing and Zhanping [29] used asynchronous communication and multithreading technology to compress large files and implemented fast and safe transmission of large files using sockets. Qinghua *et al.* [30] designed a monitoring system using sockets, which can monitor suspicious data in real time, reduce the amount of data, and reduce the network load. Wang *et al.* [31] proposed a client/server wireless network model, which was used for handheld mobile devices in the surveying and mapping industry and implemented the transmission of GIS and other data between different programs. In the research of the PCD transmission, Schwarz *et al.* [32] exploited the MPEG technology to compress PCD, while ignored the combination of the compressed point cloud and the real-time transmission. Besides this, Li *et al.* [33] also proposed Narwhal, a new point cloud video streaming system, which is implemented using DASH. The server divides the video content into slices and each slice can have a different form of encoding. The player can freely select the media segments that need to be played by implementing adaptive bitrate streaming. Jang *et al.* [34] proposed an algorithm for transmission of PCD based on H.264/AVC, but the data needs to be heavily compressed, and some environmental information is lost.

IV. System Model

A. PairCon

Our approach consists of five processing steps, as shown in Fig. 2. A Kinect camera can simultaneously produce RGB and depth images and is used as an environment sensing device. The visual odometer is responsible for reading sensor data, extracting visual features from the incoming color image, and matching these features with those of the previous image to estimate the pose. In this article, we describe a combination of the feature point method and the direct method, a combination which is not easily affected by texture, light, or other environmental factors, and which can run stably on a mobile robot platform. The ORB algorithm is wide because its speed leads to real-time performance, so it was selected for processing the feature points. In order to evenly distribute the extracted ORB feature points, we adopted a fast library for the approximate nearest neighbors (FLANN) algorithm to match features. The front end also includes analysis of the

image quality, and the fuzzy processing of the blurred image to reduce the impact on the construction.

The data acquired by the camera are inevitably affected by the outside world, so pose estimation faces cumulative drift. Therefore, it is necessary to reduce cumulative error and further optimize pose, through loop closure and back-end optimization. The principle of loop detection is to calculate the similarity of keyframes. The specific method used was to create a dictionary using the BoW model, measure the similarity between keyframes, and then judge the existence of loops in a short distance. The assessment of long-distance loops involves the memory management method. The combination of the two methods can improve the accuracy and efficiency of detection. In the back end, it is necessary to construct an optimized graph model using optimization theory. This model constructs the pose graph from the estimated pose and loop-closure detection data, optimizes the structure of the pose graph using the BA algorithm, and obtains the optimal trail of the robot over time, to construct a globally consistent robot motion trail and environment map. This part uses the graph optimizer G2O, which is based on nonlinear error functions, to solve the problem. The output of our algorithm at this stage is a globally consistent 3-D model of the perceived environment, represented as a colored point cloud. The 3-D point cloud map can be used with large-scale data. To reduce hardware memory requirements, a compact Octomap is used for data storage. Octomap can easily compress and update information, and simultaneously adjust resolution, according to the requirements of an application, to produce navigation maps with specified precision.

In this article, we describe the wireless transmission of point cloud and pose using socket technology, meeting the requirements of unlimited access in LANs. The socket is used to describe the IP address and port and is regarded as the handle of the communication chain and the basic operating unit of the communication network supporting the TCP/IP protocol. Computers can receive and transmit data using this method. The contents in the dotted box in Fig. 2 represent the socket communication steps. Stream sockets are reliable two-way communication data streams and are error-free. Terminal No. 1 first establishes a socket according to the address type (IPv4 or IPv6), socket type, and protocol, then binds the IP address and port number for the socket, monitors the port number request to prepare for receiving the connection from Terminal No. 2 at any time, and finally receives the socket request from Terminal No. 2 and sends the data after successful connection. Terminal No. 2 creates and opens the socket, tries to connect Terminal No. 1 according to the IP address and port number, and receives and prints the data from Terminal No. 1 after successful connection.

B. Tracking

1) Odometry: The input image needs to obtain the corresponding five-layer pyramid. First, the fast feature points and edge features of the first image are detected. If the number of feature points exceeds a set threshold T_1, the image is regarded as the first keyframe. Otherwise, the first keyframe is identified from the subsequent images. From the first image, the optical flow method is used to continuously track the feature points. In this process, the number of matching points between the current and the reference frame is greater than the threshold T_2, and the median parallax is greater than the threshold T_3. If the variance of parallax is large, the motion between frames is slightly large, and the E matrix is selected to calculate the pose. If the variance of parallax is small, the H matrix is selected to get the pose.

The previous frame is the reference frame, and the pose is the initial pose of the current frame. The block of the feature point on the reference frame I_{k-1} is projected to the current frame I_k to get the block of the corresponding position. The projection method is to determine the 3-D coordinates of the feature point according to the depth image, and then project to the current frame to get the 2-D coordinates, which ensures the accuracy of the block to be estimated. According to the assumption of local binary pattern (LBP) (gray scale invariant), the residual value of the block is adjusted by optimizing the position of the pixels projected from the reference frame, based on the pixel value of the current frame. The residual can be formulated as

$$\delta I = I_k(\pi({}_k p)) - f\left(u - \left(T(\psi)T_{k,k-1}\, p_{k-1} - l\right)\right) \tag{1}$$

where l is the size of half a block, ψ represents the random disturbance term, $T_{k,k-1}$ is the photometric error of all patches, p are 3-D points, and the camera projection model is π. Jacobi is calculated as follows:

$$J = \frac{\partial \delta I(\psi, u)}{\partial \psi} = -\frac{\partial I_{k-1}(u)}{\partial u}|_{u=u_i} \cdot \frac{\partial \pi(p)}{\partial p}|_{p=p_i} \cdot \frac{\partial T(\psi)}{\partial \psi}|_{\psi=0} \cdot p_{k-1} \tag{2}$$

where ψ is obtained by the Gauss–Newton method

$$\psi = -H^{-1} J^T \delta I. \tag{3}$$

In the above-mentioned process, image alignment is used to initialize the pose, and the photometric error of the pixel difference at the projection position of the same landmark is minimized, to obtain the camera pose of the previous frame. The object of optimization here is the relative pose between frames, that is, the use of the direct method.

The next step does not consider the camera pose, but directly optimizes the 2-D position of each block in the current frame, to further minimize the photometric error of the block, which establishes the reprojection error. Here, the feature blocks are aligned and the 2-D coordinates corresponding to the reprojection points are refined. The optimized object is the 2-D position of the block in the current frame, which is equivalent to the reprojection position.

In the process of pose optimization, u' is the new feature point successfully tracked in the current frame. By optimizing the pose $T_{k,w}$ of the current frame, the residual value δ between the predicted projection position and the best matching position is minimized in the corresponding layers. The residuals on the corresponding layers are as follows:

$$\delta = u' - \pi\left(T(\psi)T_{k,w}\, p\right), T(\psi) = \exp\left(\hat{\psi}\right) \tag{4}$$

where w is the world coordinate.

Algorithm 1 Visual Odometry

Require: $s1$ - The dataset path
M - Number of frames in the dataset
Ensure: $T_{k,w}$ - The camera pose
Set T_1, T_2, T_3 and other parameters.
for k = 0: M-1 **do**
  Frame $F \leftarrow$ *ReadFrames*(s1,k)
  Calculate the square difference of pixels in the image: $Temp$
  Calculate the sum of the absolute values of the differences: Num
  DR = temp / num
  if $DR < \gamma$ **then**
    dark_channel $\leftarrow$ *DarkChannelPrior*(F)
    a $\leftarrow$ *Airlight*(F, dark_channel)
    tran $\leftarrow$ *guidedFilter*()
    $F' \leftarrow$ *Deblur*(F, tran, a)
  else
    $F' \leftarrow F$
  end if

  if KeyFrames.size()< 1 **then**
    detector = FastFeatureDetector()
    detector $\rightarrow$ detect(F', kps);
    if kps > T_1 **then**
      KeyFrames $\leftarrow$ *addFrame*(F')
      FirstPose $\leftarrow$ *Initialize*()
    else
      Continue
    end if
  end if

  FPyr $\leftarrow$ *CreatePyramid*()
  size_t *align_n_tracked* = *img_align*(ref_keyframe, new_frame)
  if align_n_tracked > T_2 **then**
    KeyFrames $\leftarrow$ *addFrame*(F')
    last_frame_ = ref_keyframe
    frame_jacob $\leftarrow$ *precomputePatches*()
    ResValue $\leftarrow$ *computeResiduals*()
    if ResValue < T_3 **then**
      $H \leftarrow$ 4 pairs of matched feature points
      $(R, t) \leftarrow H = \left(R + \frac{tn^T}{d}\right)$
    else
      $E \leftarrow$ Eight-Point Algorithm
      $(R, t) \leftarrow E = R[t]_\times$
    end if
  else
    reset to last well localized pose.
  end if

  reproject2Map(F', ref_keyframe)
  new_refkeyframe $\leftarrow$ projector.n_matches;
  if new_refkeyframe < QualityMin() **then**
    $T_{k,w} = T_{k-1,w}$
    Do not update reference frame.
  end if

  for size_t iter = 0: n_iter **do**
    set parameter scale
    for m = KeyFrames.begin():KeyFrames.end() **do**
      $f(x_k) = \delta I,\ J(x_k) = f'(x_k)$
      $J^T(x_k)J(x_k)\Delta x_k = -J^T(x_k)f(x_k)$
    end for
    if $\Delta x_k \leq$ EPValue **then**
      break
    else
      $x_{k+1} \leftarrow x_k + \Delta x_k$
    end if
  end for

  SetTrackQuality(n_edges)
  if track_quality == normal **then**
    $T_{k,w} = T_{k-1,w}$
  else
    KeyFrames $\leftarrow$ *addFrame*(F')
  end if
end for

Jacobi is calculated as follows:

$$J = \frac{\partial \delta}{\partial \psi} = -\frac{\partial\left(\pi\left(T(\psi)T_{k,w}p\right)\right)}{\partial \psi} = -\begin{bmatrix} 1/z, 0, -x/z^2, -xy/z^2, 1+x^2/z^2, -y/z \\ 0, 1/z, -y/z^2, -1-y^2/z^2, xy/z^2, x/z \end{bmatrix} f. \quad (5)$$

Similarly, the ψ value of disturbance is

$$\psi = -\left(J^T J\right)^{-1} J^T \delta. \quad (6)$$

In this final step, we optimize the camera pose $T_{k,w}$ to minimize the reprojection residuals

$$T_{k,w} = \arg\min_{T_{k,w}} \frac{1}{2} \sum_i \left\| u_i - \pi\left(T_{k,w} p_i\right) \right\|^2. \quad (7)$$

Keyframe selection is very important in the whole system. In large scenarios, the amount of original data is large, but keyframe technology can reduce the number of frames to be optimized, greatly improving the efficiency of the algorithm, and avoiding redundancy of data between adjacent images. In this scheme, the selection of keyframes adopts two restrictive conditions. On the one hand, when selecting keyframes from all of the frames, the terrain of the large scene is very complex, which can easily cause the collected image to be blurred or damaged. Therefore, the ReBlur algorithm [35] can be used to determine the degree of image damage and then clear the blurred image. Use the following atmospheric scattering model:

$$\begin{aligned} I(x, y) &= J(x, y)t(x, y) + L_\infty(1 - t(x, y)), \\ t(x, y) &= e^{-\beta(y)d(x)} \end{aligned} \quad (8)$$

where $I(x, y)$ is the blurred image, $J(x, y)$ is the repaired image, L_∞ and $t(x, y)$ represent the atmospheric optical

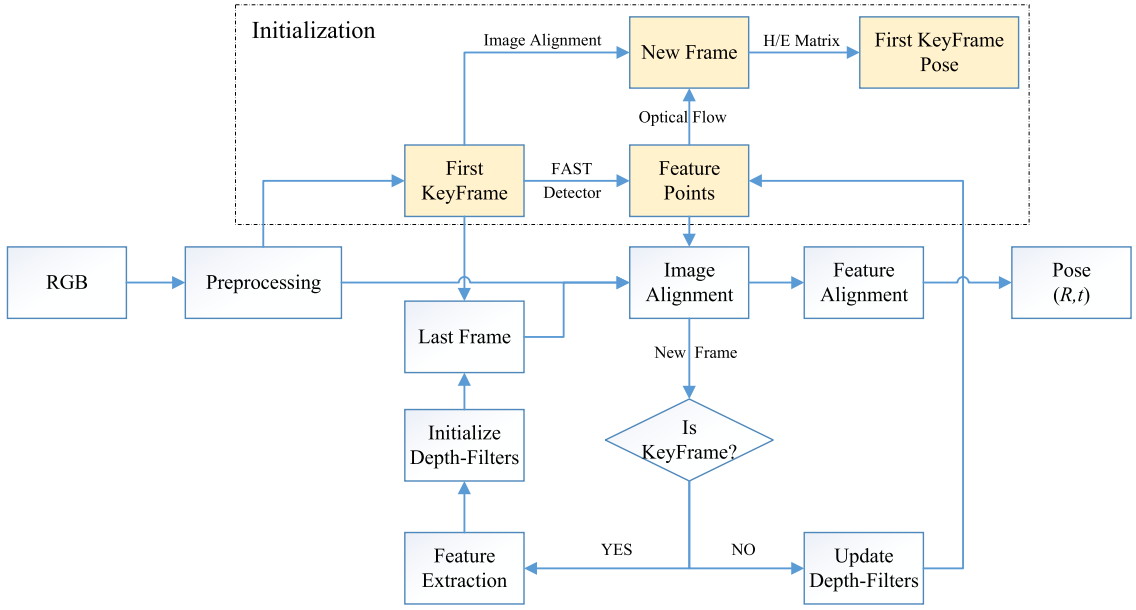

Fig. 3. Generalized overview of VO algorithm.

parameter at infinity and transmittance, respectively, the parameter x represents the position of the pixel in the image, and y represents the wavelength of the light. Using the above-mentioned formula, the final restored image can be obtained by solving the atmospheric optical parameters at infinity and transmittance. On the other hand, the number of feature points should be sufficient, and the distribution of feature points should be as uniform as possible. There are a lot of feature points that differ between keyframes.

Visual odometry can approximately estimate the pose of the camera in each frame. The pose of each frame is apparent in Fig. 3. Odometry significantly decreases drift and paves the way for a globally optimal map. The integration of all components into a general g2o solver further reduces drift. Algorithm 1 illustrates the overall visual odometry process.

2) Detection and Restoration of Blurred Image: We attempted to improve the performance of construction and reduce the effect of blurred images. First, the ReBlur algorithm is used to judge the extent of damage to the image, and then the dark channel prior algorithm based on guided filtering is used to repair the image.

The general idea of the ReBlur algorithm is to judge whether an image is blurred is that if the blurring original image is blurred again, the high-frequency component of the processed image should not change greatly. If a clear image is blurred, however, the high-frequency component of the image will change greatly. Using this information, the ReBlur algorithm first performs a Gaussian blur operation on the original image to obtain a degraded image, then calculates the gray changes of adjacent pixels in the image before and after processing, and finally compares and analyzes the gray changes of the two images to determine the definition of the original image. The ReBlur algorithm flow is shown in Fig. 4.

Combined with the above-mentioned atmospheric scattering model, the general idea of the dark channel prior algorithm is

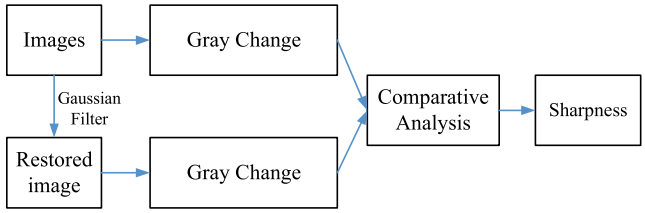

Fig. 4. Flowchart of ReBlur algorithm.

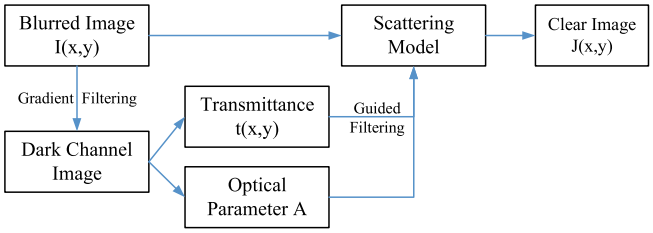

Fig. 5. Flowchart of dark channel prior algorithm.

that of a clear image taken in clear weather. A significant feature of the image is that most of the pixels have a low value in at least one of the three RGB color channels. Using this information, first, the dark channel image can be obtained, and then the atmospheric light value and transmittance can be estimated by using the dark channel image. The transmittance value can be optimized using guided filtering or soft matting methods. Finally, the restored image can be obtained by taking the results into the atmospheric scattering model. The steps of the algorithm are shown in Fig. 5.

C. Keyframe Selection

The basis of the keyframe selection algorithm is the reliable measurement of the similarity between two images. Excluding the impact of extreme conditions, such as high reflection and repeated texture in the actual environment on the optical

Fig. 6. Movement route of camera.

camera, adjacent images are assumed to follow the assumption of gray-scale invariance, and the camera moves at an approximately uniform speed for a short period of time. In this study, the method of similarity measurement S based on feature matching is studied. S is related to the distance function D of the two images

$$D\big[(CF)_t, (KF)_i\big] = \frac{\min\{\sum_t, \sum_i\} - M}{\min\{\sum_t, \sum_i\}} \tag{9}$$

$$S\big[(CF)_t, (KF)_i\big] = \frac{D_{\max} - D\big[(CF)_t, (KF)_i\big]}{D_{\max} - D_{\min}} \tag{10}$$

where $\sum_t$ and $\sum_i$, respectively, represent the number of visual features in the current frame $(CF)_t$ and the previous keyframe $(KF)_i$, and M represents the number of matching features between the two frames. $D_{\max}$ and $D_{\min}$ are the maximum and minimum expectations of the distance function D between images, and the range of S is [0, 1].

Regarding the first frame input by the system as the initial keyframe, the selection of the next keyframe is based on the following two considerations.

1) Since the previous keyframe was added, more than 15 frames have been input by the system.
2) The number of detected inliers is greater than the set minimum threshold, and the similarity measurement S is less than the set maximum threshold.

D. Loop Closure

In a real situation, a camera sensor moves along an O-scan route to construct a map of a whole area. This route includes many local loops, but a few global large loops. The point of intersection in Fig. 6 represents the splicing of adjacent blocks, in which local loops are involved. Global loops are more important than local loops with localization as the only focus. However, in the current large scene, local loops are more important. The introduction of loops can better optimize the overall pose. The influence of all frames is taken into consideration, which is helpful for the production of joint point clouds and the improvement of map accuracy.

The memory management method pays more attention to global loops in localization. Because of the large amount of data stored, this method requires a relatively long detection time in scenes with many local loops. The core idea of the memory management method is to first open two areas of memory space, named WM and LTM, which manage the possible and impossible location points, respectively. Here, the possible degree of location points is measured by the number of times that the location points are accessed, or judged by the length of time for which the location points are stored in the WM memory area. The location points stored in WM are applied to loop detection. Loop possibility is estimated using a discrete Bayesian method, which compares new location points with the location points stored in WM and detects a loop when it is found that there is a high probability of forming a closed loop between the new and old location points. Some location points in WM participate in loop detection for a long time. When no loop is detected, and thus, the time threshold is exceeded, these points need to be removed from WM and stored in LTM. The points in the LTM do not participate in loop detection. The next step is to open the third memory area, STM, with a capacity determined in advance. It observes the time similarity of continuous images, updates the weights of the location point, and transfers the top location points in the rankings of the occurrence times to WM, which indicates that these location points are accessed frequently. The storage capacity of STM depends on the moving speed of the robot and the acquisition frequency of location points. When the number of location points reaches K, the location point with the longest storage time in STM is transferred to WM.

Therefore, to simultaneously ensure efficient detection of global and local loops, we further improved the memory management method using Shi–Tomasi corner. Based on the Shi–Tomasi [36] algorithm, an improvement of the Harris [37] algorithm, we determine the feature points by comparing the minimum eigenvalues of the gradient matrix. This approach can achieve better results in many cases, improving the robustness and real-time performance of the nontexture region and avoiding redundancy in the loop closure. Shi–Tomasi's score is as follows:

$$R = \min(\lambda_1, \lambda_2) \tag{11}$$

where λ_1 and λ_2 are the eigenvalues of matrix M and M is the matrix reflecting the change of pixels in the image. It can be seen that the stability of the corner is related to the smaller eigenvalue of the matrix M, so the smaller eigenvalue is directly used as a score to measure which regions in the image have corners.

First, we construct loops by means of Shi–Tomasi feature points. We detect corner features by calculating Shi–Tomasi scores and calculate an ORB descriptor [38] for each corner after detecting a certain number of corners in the first keyframe and corresponding to BoW. We use DBow2 [39] to create a BoW dataset and determine if the current keyframe is a loop closure by querying the dataset. If the similarity score of the two frames calculated according to the BoW dataset is greater than the threshold T_s, there may be loop detection in the current keyframe. If the number of feature points in two frame matchings is small, the frame is of low quality, and loop detection is not used. Next, we should consider whether the poses of the two frames are reasonable and measure the degree of looping. We calculate a value $Dnorm$ to measure

Algorithm 2 Loop Closure

Require: $CurrFrame$ - Current keyframe
$NearL\_n$ - N frames adjacent to the CurrFrame
$RandL\_m$ - Random extraction of M keyframes
$KeyFrames$ - A collection of keyframes
Ensure: L - the set for loop detecting corresponding to

Initialize L is empty
for i = 0: $NearL\_n$ **do**
  score(CurrFrameN,i)=*query*(CurrFrame, KeyFrames[i])
  reslt ← *estimateMotion*(CurrFrame, KeyFrames[i])
  Dnorm ← $\|\Delta t\| + \min(2\pi - \|r\|, \|r\|)$
  if $Dnorm > T1 \&\& Dnorm < T2$ **then**
    if score(CurrFrameN, i)> T_s **then**
      CurrFrameN, i are added to L
    end if
  end if
end for

for k = 0: $RandL\_m$ **do**
  index = rand()%RandL_m + NearL_n
  score(CurrFrameN,index)=*query*(CurrFrame, KeyFrames[index])
  reslt ← *estimateMotion*(CurrFrame, KeyFrames[index])
  Dnorm ← $\|\Delta t\| + \min(2\pi - \|r\|, \|r\|)$
  if $Dnorm > T1 \&\& Dnorm < T2$ **then**
    if score(CurrFrameN, index)> T_s **then**
      CurrFrameN, index are added to L
    end if
  end if
end for

for m = 0: L.size()-1 **do**
  addEdge(L[m])
end for

if L.size()==0 **then**
  t_1 ← *TimeNow*()
  M_t ← *LocationPointCreation*(CurrFrame)
  if z_t(of M_t) is a bad signature (using T_{bad}) **then**
    Delete M_t
  else
    Insert M_t into STM, adding a neighbor link with M_{t-1}
    Weight Update of M_t in STM (using $T_{similarity}$)
    if STMs size reached its limit (M_{STM}) **then**
      Move oldest locationPoints of STM to WM
    end if
    $p(S_t|M_t)$ ← Bayesian Filter Update in WM with M_t
    Loop Closure Hypothesis Selection (S_t = k)
    if S_t = k is accepted (using T_{loop}) **then**
      Add loop closure link between M_t and M_k
      t, k are added to L
      addEdge(L[0])
    end if
    Retrieval(M_k)
    t_2 ← TimeNow() - t_1
    if $t_2 > T_{time}$ **then**
      Transfer()
    end if
  end if
end if

the motion

$$\text{Dnorm} = \|\Delta t\| + \min(2\pi - \|r\|, \|r\|). \tag{12}$$

This problem can be regarded as the norm sum of translation and rotation. Dnorm $> E_{\text{error}}$ refers to too large motion, wrong calculation, and no loop detection for this frame. Dnorm $< E_{\text{key}}$ indicates that it is very close to the previous frame, and this operation is also abandoned.

Finally, if the local loop is not detected, the memory management method is used to detect the global loop. Algorithm 2 shows the overall loop closure detection process. The loop detection framework is shown in Fig. 7.

V. Experiments

We evaluate the performance of our system both quantitatively and qualitatively in terms of trajectory estimation, computational performance, and surface reconstruction accuracy.

A. Datasets and Experimental Platform

All experiments have been conducted on the platform configured x64 Ubuntu16.04 operation system, equipped with one Intel Core i7-10875H CPU (2.30 GHz), one NVIDIA GeForce GTX 2060 GPU with 16-GB memory in the first terminal. Another experimental platform is performed on an x64 Ubuntu16.04 desktop with an Intel Core i5-5200 2.20-GHz processor with 8-GB memory, and one AMD R5 GPU.

In this article, we first select some data from the TUM dataset that met the experimental requirements. Considering the impact of global and local features, for the scene selected for the experiment, the situation in which objects in different positions had the same shape due to different perspectives were excluded, as were situations where there was only descriptive information around feature points in the scene, and information without the relative distribution of features. The TUM dataset [40] is an experimental scene dataset from the Technical University of Munich, Munich, Germany, in which the *fr1* and *fr2* series are static scene sequence sets, while the *fr3* series consists of typical dynamic scene sequence sets. The dataset provides RGB and depth images taken using Kinectv1 cameras and ground truth provided by external high-precision sensors. These sequences include different camera motion states when collected, such as fast movement, slow movement, and strong rotation, and so on. Other datasets used include ICL-NUIM, Microsoft, Stanford, and so on, and the Gazebo and Airsim simulators. Instructions for the use of relevant data will be explained later.

Systematic evaluation indicators included: absolute trajectory error (ATE), the absolute error between the camera trajectory and the true trajectory at each time point used, that is, the root-mean-square error for calculation of the Euclidean

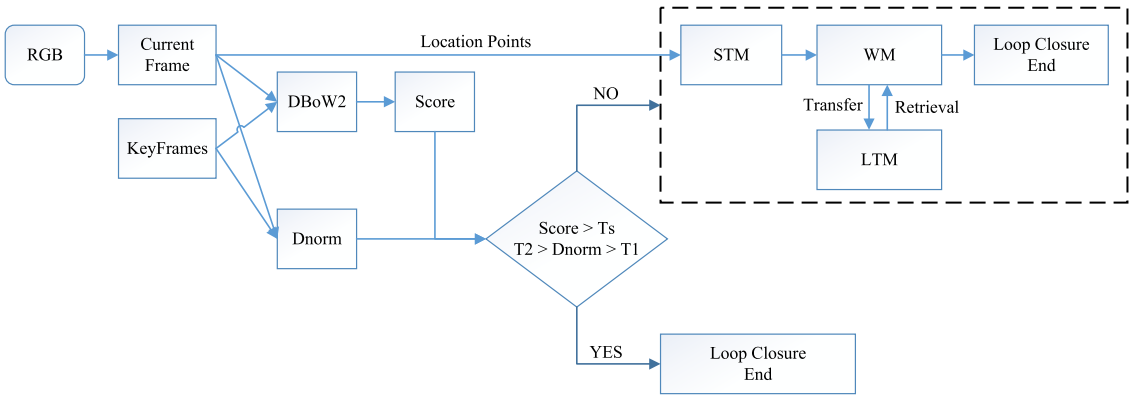

Fig. 7. Generalized overview of loop closure.

TABLE I
RESULTS OF RMSE ATE [m] USING TUM SEQUENCES FOR SEVERAL DIFFERENT SLAM

SLAM	Sequences									
	fr1/desk	**fr1/room**	**fr1/floor**	**fr1/xyz**	**fr2/desk**	**fr2/xyz**	**fr3/office**	**fr3/w_half**	**fr3/w_xyz**	**fr3/s_half**
DVO	0.021	0.053	0.232	0.011	0.017	0.018	0.035	0.263	0.436	0.102
RGBD	0.025	0.084	0.035	0.014	0.057	0.008	0.032	-	-	-
MRSMap	0.043	0.069	0.137	0.013	0.052	0.021	0.042	-	-	-
Kintinuous	0.037	0.075	0.363	0.017	0.034	0.029	0.038	-	-	-
SVO	0.033	0.045	0.250	0.067	0.407	0.034	0.166	0.041	0.037	**0.072**
ORBSLAM 2	0.017	0.410	**0.026**	0.018	0.014	**0.008**	0.035	0.495	0.696	0.082
LSD-SLAM	0.107	0.560	0.380	0.092	0.046	0.029	0.396	-	-	-
PairCon	**0.012**	**0.037**	0.120	**0.008**	**0.013**	0.016	**0.023**	**0.038**	**0.035**	0.066

distance between the two, which is applicable to the evaluation of the performance of the VSLAM system. The ATE is obtained as follows:

$$\mathrm{ATE}(\hat{X}, X) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[\mathrm{trans}(\hat{X}) - \mathrm{trans}(X)\right]^2}. \quad (13)$$

Relative pose error (RPE) is a measure of the difference in pose variation in a time interval. The true pose and the estimated pose are calculated at the same time. Then, the variation is subtracted to obtain the RPE. The standard is applicable to the evaluation of the drift of visual odometers.

B. Performance Comparison of Keyframe and Odometry—Datasets

Each sequence of the TUM benchmark dataset contains RGB and depth images captured from Microsoft Kinect RGB-D cameras and can obtain accurate camera motion tracks through the motion capture system. We compared the results of our algorithm with those of seven SLAM algorithms on TUM datasets: DVO SLAM (DVO), RGBD-SLAM [41], MRSMap, Kintinuous, semi-direct visual odometry (SVO), ORB-SLAM2, and LSD-SLAM [42]. The ATE results of each algorithm are shown in Table I. "-" indicates the loss of frame and poses that cannot be recovered.

As shown in Table I, the PairCon-SLAM algorithm in this article performed better than the other algorithms. On the *fr2/xyz* sequence with rich texture, the performance of the algorithm in this article was slightly worse than that of ORB-SLAM2, mainly because this method cannot completely overcome the cumulative drift in the translation direction among these sequences that move along the Kinect spindle. However, for low-texture sequences like *fr3*, the algorithm in this article was much more effective than the ORB-SLAM2 algorithm. Even comparing DVO systems, its performance was still outstanding. For the DVO algorithm, there is no effective keyframe selection and loop detection to optimize the overall drift error. In low-texture scenarios, this algorithm achieves more robust and accurate positioning effects.

TABLE II
KEYFRAMES OF PAIRCON-SLAM IS DEGRADED 5%–23% COMPARING WITH ORIGINAL ORBSLAM2

SLAM	Sequences							
	fr1/desk	fr1/room	fr1/floor	fr1/xyz	fr1/360	fr1/desk2	fr2/desk	fr2/xyz
	595	1360	1242	798	755	639	2964	3669
ORBSLAM2	71	172	82	38	105	95	168	**40**
PairCon	**55**	**156**	**68**	**31**	**84**	**90**	**132**	65

TABLE III
AVERAGE PROCESSING TIME (S) OF PER FRAME COMPARING WITH ORIGINAL ORBSLAM2

SLAM	Sequences							
	fr1/desk	fr1/room	fr1/floor	fr1/xyz	fr1/360	fr1/desk2	fr2/desk	fr2/xyz
	595	1360	1242	798	755	639	2964	3669
ORBSLAM2	0.0533	0.0476	0.0352	0.0467	0.0392	0.0527	0.0563	0.0418
PairCon	0.0571	0.0493	0.0384	0.0511	0.0421	0.0532	0.0611	0.0435

Under the *sitting* series of low-dynamic scenarios, the ORB-SLAM2 algorithm uses the BA optimization algorithm with the culling of outliers for pose optimization. A small number of moving points can be judged to be error matches. Because of the greater number of static points produced, the ORB-SLAM2 algorithm includes more information and has a better performance in camera pose optimization. In the *walking* series of high dynamic scenarios, there are more dynamic feature points and the ORB-SLAM2 algorithm could not continue to eliminate dynamic points. When optimizing pose, dynamic points are treated as static points so that pose errors are large.

The types of features that cannot be distinguished by the feature-based LSD-SLAM algorithm are moving objects in *fr3* scenes. Feature points on dynamic objects can result in errors, while the depth filtering method cannot converge to the final result, a situation that will directly lead to low or even error in the estimated accuracy. Experiments show that our approach can effectively handle large-scale scenarios, and the operation results were better than those of ORB-SLAM2 and LSD-SLAM. Compared with the system, it was slightly less than SVO on only one sequence. The localization error of RGBD-SLAM on the *fr1/room* scene was the largest, at up to 0.084 m, but the error of this method on this packet is only 0.037 m. An *fr3* picture is captured by a Kinect camera mounted on the top of a Pioneer robot. As the robot's motion is nonuniform and fast, violent shakes and large scenarios challenge the stability of SLAM systems. Most SLAM systems, including MRSMap and Kintinuous, fail to track in these sequences. Through evaluation on an available RGB-D benchmark, we demonstrate that our approach achieved higher accuracy on average than existing feature-based methods.

The keyframe selection proposed in this article can select more keyframes, to provide more reliable data for map constructing and optimization, without affecting the real-time performance of the system. Table II shows the results of the improved SLAM algorithm and ORB-SLAM2 for the number of keyframes under partial sequences. The algorithm can always reduce the number of keyframes in different sequences, so data redundancy can be removed for local or global optimization, loop detection, and other processes, to improve the accuracy of the pose. After adding the combination optimization strategy and the keyframe mechanism, the accuracy of the pose was improved without affecting its real-time performance using ordinary CPUs. Table III shows the results of a comparison of the results of the optimized SLAM algorithm and ORB-SLAM2 with respect to the average transaction time of each frame. In most sequences, the improved algorithm takes only about 3 ms more per frame than ORB-SLAM2 on average, because the process of selection of keyframes is strict, and it takes some time to implement the composing strategy. However, in the actual process, it will not affect the real-time operation of the system. Instead, it can improve the accuracy of the camera pose and reduce the localization error.

(a) (b) (c)

Fig. 8. Result of image definition judgment and the effect of image restoration. (a) Clear images. (b) Blurred images. (c) Restored images.

C. Processing Results of Blurred Image—Datasets

In this study, we selected some data from the VOC and COCO dataset as the dataset of clear images. Images with rain and fog were taken from the RTTS dataset and images on the Web. The ReBlur algorithm was used to judge the clarity of the image, and then the blurred image was restored. The judgment result and image inpainting result are shown in Fig. 8, where Fig. 8(a) is an image judged to be clear, Fig. 8(b) is an image judged to be fuzzy, and Fig. 8(c) is the blurred image after restoration. The ReBlur algorithm can accurately judge whether an image is clear or not. However, the dark channel prior algorithm based on guided filtering was used to restore the unclear image. After restoration, the details of the object are clearer and richer, the contour is more obvious, and recognition is better.

D. Performance Comparison of Construction—Datasets, Gazebo, and Airsim

In this article, we introduce the Gazebo simulator under the open-source robot operating system (ROS) and used its powerful physics engine to build a simulation platform of the Xbot robot, used a programming interface to implement an RGBD camera with adjustable parameters mounted on the

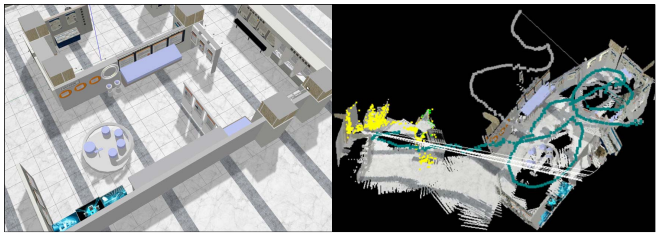

Fig. 9. Software museum world and 3-D map of the scene.

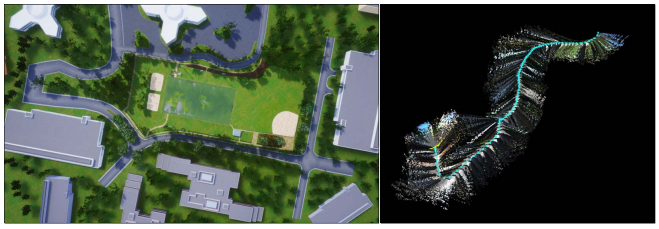

Fig. 10. Airsim world and 3-D map of the scene.

Fig. 11. Dense map of the proposed approach in Microsoft, Stanford, and Tum Datasets.

robot, and then used a high-quality 3-D model to build a test scene with variable surface texture. The model renders an actual scene of the Institute of Software Chinese Academy of Sciences (ISCAS) with specific proportions, forming a complete VSLAM simulation environment. We used the data obtained from this scene to construct a map, and the results are shown in Fig. 9.

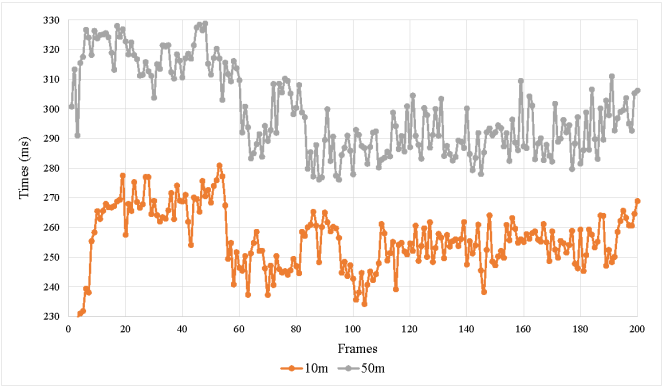

Fig. 12. Time spent of PCD in Condition 1.

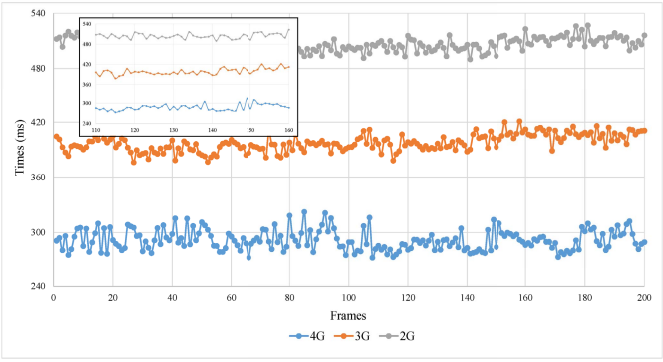

Fig. 13. Time spent of PCD in Condition 2.

Airsim is a simulation software based on the Unreal engine that can be used for the simulation of robots, such as UAVs and cars. It is open-source and can be used under different operating systems and supports online simulation with PX4 or ArduPilot. Airsim has been developed as a plugin for Unreal and can be configured to any Unreal environment. Airsim provides application programming interfaces (APIs) for actions, such as saving data, controlling robots, and so on. We used these APIs to grab the data in a scene to construct a map with which to test the performance of the algorithm. The results are shown in Fig. 10.

We evaluated the surface reconstruction results of our system on the ICL-NUIM dataset. This benchmark provides ground truth poses for an RGBD camera moved in a complex environment and a ground truth 3-D model, which can be used to evaluate the accuracy of surface reconstruction. Although this dataset is synthetic, the structure and texture in the scene are rendered realistically by professional 3-D software. We evaluated our method on four trajectories of the *living room* scene, including synthetic noise, and provided surface reconstruction results compared with the same SLAM systems listed in Table IV. The performance of the proposed algorithm was better than those of the other algorithms. The reconstruction result of the PairCon-SLAM system was not the best in the *kt2* sequence, but good reconstruction accuracy was achieved in the *kt0*, *kt1*, and *kt3* sequences. DVO achieved lower accuracy than the system in this article in most of the tests of scene sequences containing planar features because it does not utilize planar features.

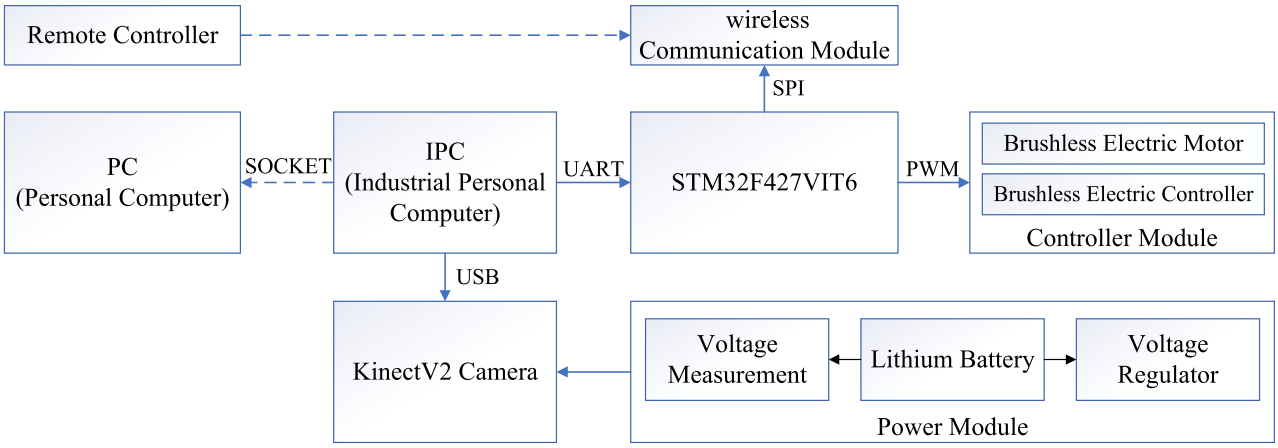

Fig. 14. Diagram of system hardware structure.

TABLE IV
SURFACE RECONSTRUCTION ACCURACY RESULTS FOR DIFFERENT SLAM SYSTEMS

SLAM	Sequences			
	kt0	kt1	kt2	kt3
DVO	0.032m	0.061m	0.119m	0.053m
RGB-D SLAM	0.044m	0.032m	**0.031m**	0.167m
MRSMap	0.061m	0.140m	0.098m	0.248m
PairCon	**0.018m**	**0.024m**	0.034m	**0.031m**

Finally, we use this algorithm to construct maps for some scenes in the Microsoft, Stanford, and TUM datasets and the results are shown in Fig. 11. The experimental results show that the method in this article can effectively reconstruct various scenes.

E. Performance Test of Transmission in Socket—Datasets

The depth camera can simultaneously capture the RGB data and depth data of a target object at a frame rate of about 30 fps, called RGBD data. RGBD can be converted into PCD with approximately 300 000 dots per frame with color information. In the real-time transmission conditions of specific environmental data, the system can improve the point cloud based on the Hextree using "frame discard." The image of the current frame and the previous frame at the sending end are compared. If the changes are small, the current frame is discarded and not transmitted. This improvement can reduce the number of frames transmitted per second, saving bandwidth when the camera is moving slowly. Finally, PCD can be transmitted using limited bandwidth, to ensure the quality and efficiency of real-time mapping. Consider the following two conditions for experiment and analysis.

Condition 1: In a 4G network environment, three sets of the same data at different distances are transmitted for testing, and the corresponding transmission time of each set is recorded.

Condition 2: In an environment with the same dataset and constant transmission distance of 3 m, three sets of the same data in different network environments are transmitted for testing and the transmission time of each set is recorded.

In the case of Condition 1, the selected data are the *ikea-table* in the Stanford dataset, and the distance between two terminals was set as 10 and 50 m, respectively. Specifically, we have tested the data and plotted all frames (transmission time of 200 frames) to present the respective general trend of each set. Then, we zoomed in on the curves between 110 and 160 frames to show the difference in detail of each set of data. In Fig. 12, we observed that the average transmission time of the same data at different distances was different (255.769 ms versus 299.173 ms). Thus, the distance between terminals can affects the transmission speed of the PCD.

In the case of Condition 2, the selected data are the *ikea-table* in the Stanford dataset. And we adopt three different network environments, i.e., 2G, 3G, and 4G. In Fig. 13, we observe that the average transmission time of the same data in different network environments was different (2G: 505.173 ms versus 3G: 397.408 ms versus 4G: 291.518 ms). It was noted that the case of 4G has a larger fluctuation of the curve than the one in the case of 2G and 3G, which indicates that the different networks affect the transmission speed of the PCD.

F. Performance Test of Construction in the Real World

The system can use industrial computers produced by manufacturers, such as Intel, as the operating carrier of the software system. Among them, the industrial PC (IPC) is characterized by small size and low power consumption. The physical connection relationship in the hardware platform is shown in Fig. 14. The UAV/UGV is equipped with a KinectV2 camera, a 16 800-mAh mobile power supply, a wireless module, and an IPC. Among them, the depth camera is used as a sensor to collect RGB and depth images to obtain data from the outside world. The wireless module controls the movement of the unmanned ground vehicle through the single-chip microcomputer after receiving the control command sent by the remote controller. On the IPC, the odometer, back-end optimization, and loop closure parts of the algorithm are completed. On the other PC, the mapping thread is run to

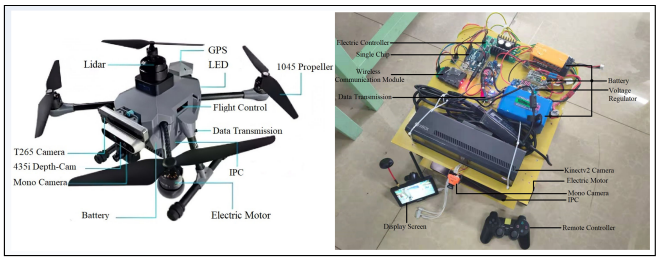

Fig. 15. Mobile robot platform used for VSLAM tests.

Fig. 16. Mapping result of buildings in a residential area.

map the received data, and the final object is shown in Fig. 15. Meanwhile, the mapping result of buildings in a residential area, as shown in Fig. 16.

VI. Conclusion

In this article, we focus on dense construction in large scenarios. We found that the improved VO algorithm improved the localization performance. Compared with the original algorithm, we could reduce the RMSE-ATE value and the number of keyframes in different datasets. The blurring of images caused by environmental factors was rectified, making tracking and location of feature points in images more reliable. The semi-direct method, combining detection and restoration of blurred images, is robust to foggy weather and nontextured environments. Memory management was combined with DBoW2 and Dnorm to build the loop closure part, which improved the accuracy and speed of detection and could detect loops at large and small distances in real time. The socket method realizes the transmission of each frame of point cloud data depending on communication velocity and was tested on several datasets and simulation platforms.

References

[1] J.-C. Trujillo, R. Munguia, S. Urzua, E. Guerra, and A. Grau, "Monocular visual SLAM based on a cooperative UAV–target system," *Sensors*, vol. 20, no. 12, pp. 3531–3562, Mar. 2020.

[2] J. Artieda *et al.*, "Visual 3-D SLAM from UAVs," *J. Intell. Robotic Syst.*, vol. 55, nos. 4–5, pp. 299–321, Jan. 2009.

[3] Y. Pan, X. Xu, X. Ding, S. Huang, Y. Wang, and R. Xiong, "GEM: Online globally consistent dense elevation mapping for unstructured terrain," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–13, 2021.

[4] F. Caballero, L. Merino, J. Ferruz, and A. Ollero, "Vision-based odometry and SLAM for medium and high altitude flying UAVs," *J. Intell. Robotic Syst.*, vol. 54, nos. 1–3, pp. 137–161, Mar. 2009.

[5] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, and J. Mcdonald, "Kintinuous: Spatially extended KinectFusion," *Robot. Auto. Syst.*, vol. 20, pp. 1–10, Jul. 2012.

[6] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Hong Kong, May 2014, pp. 15–22.

[7] M. Labbé and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," *IEEE Trans. Robot.*, vol. 29, no. 3, pp. 734–745, Jun. 2013.

[8] S. ShahEmail, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," *Field Service Robot.*, vol. 5, no. 40, pp. 621–635, Nov. 2017.

[9] T. Li, S. Hailes, S. Julier, and M. Liu, "UAV-based SLAM and 3D reconstruction system," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Macau, China, Dec. 2017, pp. 2496–2501.

[10] Z. Liu, D. Kaichang, L. Jian, and X. Cui, "Landing site topographic mapping and rover localization for Chang'e-4 mission," *Sci. China Press*, vol. 4, no. 63, pp. 166–177, Apr. 2020.

[11] B. F. Wang, J. L. Zhou, G. S. Tang, and D. Kaichang, "Research on visual localization method of lunar rover," *Scientia Sinica Informat.*, vol. 4, no. 44, pp. 452–460, Apr. 2014.

[12] K. Di *et al.*, "Photogrammetric processing of rover imagery of the 2003 Mars exploration rover mission," *ISPRS J. Photogramm. Remote Sens.*, vol. 63, no. 2, pp. 181–201, Mar. 2008.

[13] A. Concha and J. Civera, "RGBDTAM: A cost-effective and accurate RGB-D tracking and mapping system," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Vancouver, BC, Canada, Sep. 2017, pp. 6756–6763.

[14] T. Whelan, S. Leutenegger, R. S. Moreno, B. Glocker, and A. Davison, "ElasticFusion: Dense SLAM without a pose graph," in *Proc. 11th Robot., Sci. Syst.*, Rome, Italy, Jul. 2015, pp. 1–9.

[15] P. Li, R. Wang, Y. Wang, and W. Tao, "Evaluation of the ICP algorithm in 3D point cloud registration," *IEEE Access*, vol. 4, pp. 1–19, 2020.

[16] J. Cheng, H. Zhang, and M. Q.-H. Meng, "Improving visual localization accuracy in dynamic environments based on dynamic region removal," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 3, pp. 1585–1596, Jul. 2020.

[17] R. Wang *et al.*, "A robust registration method for autonomous driving pose estimation in urban dynamic environment using LiDAR," *Electronics*, vol. 1, no. 8, pp. 43–63, Jan. 2019.

[18] M. S. Bahraini, A. B. Rad, and M. Bozorg, "SLAM in dynamic environments: A deep learning approach for moving object tracking using ML-RANSAC algorithm," *Sensors*, vol. 19, no. 17, pp. 3699–3718, Aug. 2019.

[19] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet++: Fast and accurate LiDAR semantic segmentation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Venetian Macao, China, Nov. 2019, pp. 4213–4220.

[20] J. Zhou, Y. Huang, and B. Yu, "Mapping vegetation-covered urban surfaces using seeded region growing in visible-NIR air photos," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 5, pp. 2212–2221, May 2015.

[21] J.-H. Kim, C. Cadena, and I. Reid, "Direct semi-dense SLAM for rolling shutter cameras," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Stockholm, Sweden, May 2016, pp. 1308–1315.

[22] G. He, X. Yuan, Y. Zhuang, and H. Hu, "An integrated GNSS/LiDAR-SLAM pose estimation framework for large-scale map building in partially GNSS-denied environments," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–9, 2021.

[23] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Minneapolis, MI, USA, Jun. 2007, pp. 1–8.

[24] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1704–1716, Sep. 2011.

[25] E. Garcia-Fidalgo and A. Ortiz, "IBoW-LCD: An appearance-based loop-closure detection approach using incremental bags of binary words," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3051–3057, Oct. 2018.

[26] F.-A. Moreno, D. Zuñiga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez, "PL-SLAM: A stereo SLAM system through the combination of points and line segments," *IEEE Trans. Robot.*, vol. 35, no. 3, pp. 734–746, Jun. 2019.

[27] X. Gao, R. Wang, N. Demmel, and D. Cremers, "LDSO: Direct sparse odometry with loop closure," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Madrid, Spain, Oct. 2018, pp. 2198–2204.

[28] Y. Hou, H. Zhang, and S. Zhou, "BoCNF: Efficient image matching with Bag of ConvNet features for scalable and robust visual place recognition," *Auton. Robots*, vol. 42, no. 6, pp. 1169–1185, Aug. 2018.

[29] Y. Yubing and L. Zhanping, "Research of fast and safe large files transmission based on socket," in *Proc. 7th Int. Conf. Comput. Sci. Educ. (ICCSE)*, Melbourne, VIC, Australia, Jul. 2012, pp. 483–485.

[30] W. Qinghua, H. Bingbing, and S. Runjie, "Application research of an efficient communication method in the wind-turbine vibration monitoring system based on socket protocol," in *Proc. Int. Conf. Comput. Intell. Commun. Netw. (CICN)*, Jabalpur, India, Dec. 2015, pp. 604–606.

[31] K. Wang, N. Shu, L. Li, and F. Lv, "A geo-information data transmission method in socket-based mobile device," in *Proc. Int. Conf. Comput. Intell. Softw. Eng.*, Chengdu, China, Sep. 2010, pp. 1–4.

[32] S. Schwarz *et al.*, "Emerging MPEG standards for point cloud compression," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 1, pp. 133–148, Mar. 2019.

[33] J. Li, C. Zhang, Z. Liu, W. Sun, W. Hu, and Q. Li, "Demo abstract: Narwhal: A DASH-based point cloud video streaming system over wireless networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Toronto, ON, Canada, Jul. 2020, pp. 1326–1327.

[34] G.-R. Jang, Y.-D. Shin, J.-H. Park, and M.-H. Baeg, "Real-time point-cloud data transmission for teleoperation using H.264/AVC," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot.*, Hokkaido, Japan, Oct. 2015, pp. 2374–3247.

[35] Y. Bahat, N. Efrat, and M. Irani, "Non-uniform blind deblurring by reblurring," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 3306–3314.

[36] R. J. Mstafa, Y. M. Younis, H. I. Hussein, and M. Atto, "A new video steganography scheme based on Shi-Tomasi corner detector," *IEEE Access*, vol. 8, pp. 161825–161837, 2020.

[37] R. Li, D. Shi, Y. Zhang, K. Li, and R. Li, "FA-Harris: A fast and asynchronous corner detector for event cameras," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Macau, China, Nov. 2019, pp. 6223–6229.

[38] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, Nice, France, Oct. 2003, pp. 1470–1477.

[39] M. Burri *et al.*, "The EuRoC micro aerial vehicle datasets," *Int. J. Robot. Res.*, vol. 35, no. 10, pp. 1157–1163, Nov. 2016.

[40] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vilamoura-Algarve, Portugal, Oct. 2012, pp. 573–580.

[41] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *Proc. IEEE Int. Conf. Robot. Autom.*, Saint Paul, MN, USA, May 2012, pp. 1691–1696.

[42] J. Engel, T. Schps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. Eur. Conf. Comput. Vis.*, Zürich, Switzerland, 2014, pp. 834–849.