EFFICIENT AND ROBUST SLAM: INTEGRATION AND EVALUATION OF ORB-SLAM2 EXTENSIONS

A DISSERTATION SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF MASTER OF SCIENCE
IN THE FACULTY OF SCIENCE AND ENGINEERING

2020

Student id: 10456489

Department of Computer Science

Contents

Word Count: 16261

List of Tables

List of Figures

Abstract

EFFICIENT AND ROBUST SLAM: INTEGRATION AND EVALUATION OF ORB-SLAM2 EXTENSIONS
Jiasen Tian
A dissertation submitted to The University of Manchester
for the degree of Master of Science, 2020

In recent years, the Simultaneous Localization and Mapping (SLAM) has been a hot point of perception algorithms. There are lots of excellent SLAM algorithms have been published which have achieved good performance in different scenarios. In this project, three ORB SLAM2 extension algorithms (Dyna SLAM, Fisheye SLAM and GCNv2 SLAM) are optimized by integrating their advantages. Three new integrated SLAM algorithms are proposed to promote the performance of the original SLAM algorithms, which include stereo camera mode of GCNv2 SLAM, the monocular camera mode of Fisheye + GCNv2 SLAM and the RGB-D camera mode of Dyna + GCNv2 SLAM. Three achieved SLAM algorithms have been repeatedly tested and evaluated in the corresponding datasets (TUM, TUM VI, and EuRoC). The evaluation results demonstrate that the tracking error rates of three achieved SLAM algorithms have declined 2%, 6.2% and 8.1% respectively on average, compared with the original SLAM algorithms. Furthermore, the new achieved SLAM algorithms require less computation resource because the GCNv2 descriptor requires less computation cost than ORB descriptor. The results show that achieved SLAM algorithms have better accuracy and robustness than the original SLAM algorithms.

Declaration

No portion of the work referred to in this dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

My deepest gratitude goes first and foremost to my professor, professor Mikel, who continued to guide online even during the lockdown. Without his patient guidance and enthusiastic encouragement, the final version of this thesis will not be presented. I also want to thank the PhD students Mihai and Konstantinos, who helped me a lot. Furthermore, I would like thanks to the medical staff who guarded us during the epidemic. Finally, my gratitude would give to my beloved family for their considerations and great confidence in me all through the tough times.

Chapter 1

Introduction

Simultaneous Localization and Mapping (SLAM) is a widespread perception algorithm which is widely used in many applications, such as the auto driven car, drone or service robotics. It aims to establish the map for the unknown environments and estimate the posture or position for moving senor. There are kinds of typical SLAM technologies which base on the different sensors such as visual cameras, laser radar and ultrasonic wave radar. The visual SLAM is the most popular SLAM technology because the camera devices are stable, convenient, and easy to get.

There are three most extensive used camera types: monocular camera, stereo camera (binocular camera) and the RGB-D camera. The monocular camera has the advantages of cheap cost and easy to use, but the accuracy of SLAM results could be affected by lacking precise depth data. The stereo and RGB-D camera enquire this problem through their equipment advantages. The stereo camera calculates the depth information by triangulating the matching points from the right and left image. The depth camera obtains the depth data directly from itself depth sensor.

During these years, the research of SLAM algorithms has a rapid development. There are lots of excellent SLAM algorithms proposed for different cameras and environments. The ORB SLAM2 is most classic SLAM algorithms which have three camera modes to compatible for all the mainstream types of camera: monocular, stereo and RGB-D camera. The ORB SLAM2 uses the ORB descriptor to recognize the objects and track the camera pose. And it provides robust and efficient solutions with monocular, stereo and RGB-D camera modes. Because of its stable framework and high tracking performance, there are several outstanding derivatives presented for more specific usages. For instance, the Fisheye SLAM is proposed to provide a robust tracking solution for the omnidirectional camera. The Dyna SLAM offers a high accuracy mapping

and tracking solution for the dynamic scene. The GCNv2 SLAM proposes an efficient and low computation descriptor (GCNv2 descriptor), and the new GCNv2 descriptor gets a higher performance on the original ORB SLAM2 framework.

Even though these derivative algorithms of ORB SLAM2 have their highlights in different aspects, however, there is no SLAM algorithm try to fuse their advantages into a single SLAM algorithm. The Dyna SLAM provides an efficient solution to detect dynamic objects and reduce the influences of the active items in the tacking and mapping processes. But the mass computation cost is required to perform the dynamic detection algorithms. Furthermore, the accuracy of SLAM is also affected by removing a large number of feature points from the image. Fortunately, the advantages of GCNv2 SLAM could fill the defects of Dyna SLAM, which has a low computation feature extraction method and more efficient than ORB extraction algorithm.

Similarly, the Fisheye SLAM extends the ORB SLAM2 to compatible the omni-directional camera by adding the EUCM (Enhanced Unified Camera Model). However, larger FOV (Field of View) also means the SLAM algorithms could extract more feature points. More computation resources are required to extract the ORB feature points. And the GCNv2 SLAM provides a suitable alternative for Fisheye SLAM. The GCNv2 descriptor not only requires less computation cost but also provides a more accurate and efficient method of feature points extraction.

1.1 Objectives and contributions

The objectives of this project are to promote the performance of original SLAM algorithms by fuse other advantages. The focused aspects of improvement are in accuracy, robustness and computation cost. Therefore, three optimized SLAM algorithms have been achieved in this project, which has more accuracy, efficiency and robustness than the original SLAM algorithms. Three experiments are presented to test and evaluate the performance of new achieved SLAM algorithms. The key points of evaluation are the focus on the aspects of accuracy and robustness. Furthermore, the results of established maps are also compared with the corresponding original SLAM algorithms to evaluate the quality of tracking and mapping processes.

The main contributions of this project are that three new integrated SLAM algorithms are achieved which fuse the advantages of original SLAM algorithms, and the benefits of GCNv2 SLAM are used as the focuses of integrated SLAMs.

- The stereo camera mode of GCNv2 SLAM is achieved which purposes to extends the GCNv2 descriptor to the stereo camera tracking system since the original GCNv2 SLAM only accomplished the RDB-D camera case. The results of the evaluation show the quality of the established map is better than the ORB SLAM, there are more map points in GCNv2 SLAM map, and the points are denser and more evenly distributed then ORB SLAM map. Furthermore, the achieved SLAM has lower computation cost and higher accuracy and robustness than ORB SLAM2.

- The monocular camera mode of Fisheye + GCNv2 SLAM is proposed which aims to provide a better omnidirectional camera solution with higher accuracy and lower computation cost. As the results of the evaluation suggest, the accuracy of Fisheye + GCNv2 SLAM has improved at least 6.2% on average. Furthermore, it performs much better performance in aspect of mapping than the original Fisheye SLAM, since the Fisheye + GCNv2 SLAM is more precise and robust than Fisheye SLAM.

- The RGB-D camera mode of Dyna + GCNv2 SLAM has achieved which purposes improve the accuracy and efficient on the aspects of tracking, mapping and background inpainting function. According to the results of the evaluation, the accuracy of achieved SLAM algorithm has a great improvement compared with the original Dyna SLAM which the Relative Pose Error (RPE) has declined at least 8.1% on average. As the promotion of accuracy, the quality of background inpainting and map establishing process also have been improved. The results show that the Dyna + GCNv2 SLAM is more efficient, accurate and robust than Dyna SLAM.

1.2 Thesis Structures

The overall layout of this thesis is shown as follows. Chapter 2 gives the background and theory of relative SLAM algorithms in detail, which includes ORB SLAM2, Dyna SLAM, GCNv2 SLAM and Fisheye SLAM. Chapter 3 is the methodology part which contains the implementations of three integrated SLAM algorithms. Then Chapter 4 provides the results and evaluations for three achieved SLAM algorithms, respectively.

And the used datasets and evaluation tools are also introduced in chapter 4. Furthermore, the discussion is presented in chapter 5, which summaries the results and limitation for each achieved SLAM algorithms. Finally, chapter 6 contains the summary and conclusion of this project, and the future works are also involved in chapter 6.

Chapter 2

Background

In this chapter, the background and the related works are presented which are relevant to this project. Some foundation algorithms of feature extraction methods are provided firstly which are include Fast algorithm, Brief algorithm and ORB algorithm. Then, four related SLAM algorithms (ORB SLAM2, Dyna SLAM, Fisheye SLAM and GCNv2 SLAM) is deeply introduced and discussed. And the related works of each SLAM algorithm are also presented in the corresponding section.

2.1 ORB Feature Extraction

Rublee et al. [25] proposed the ORB algorithm, which is the short name of Oriented Fast and Rotated Brief, is a popular feature extraction method. It aims to detect key points from images and create feature vectors quickly for each key point, respectively. The key points are stand-out and difference from nearby pixels which has obvious pixel value changing, such as corners. Those key points vectors will be foundational factors of feature matches which is a vital part of objects recognition. The ORB algorithm is famous for its super-fast speed and less affected by noise and image transformations, such as rotation and scale transformations. More in details, the ORB algorithm contains Fast (Features from Accelerated Segments Test[23]) and Rrief (Binary Robust Independent Elementary Features [24]) algorithms, which are feature detection algorithm and vector creation algorithm. The introduction of each algorithm will be presented in this part, respectively.

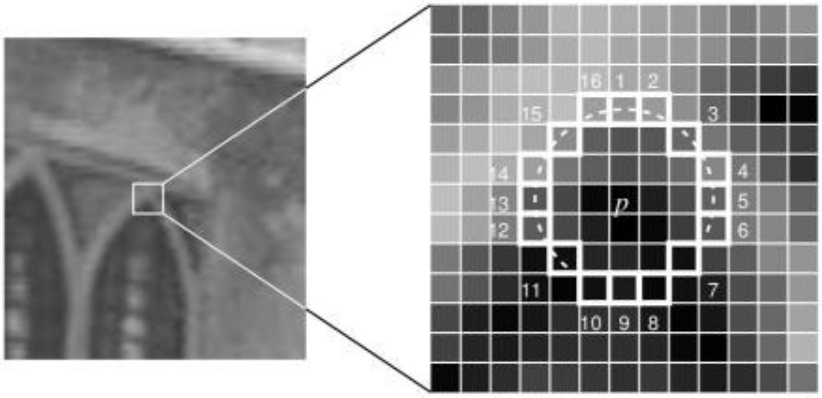


Figure 2.1: The corner detection of FAST algorithm which proposed by Rosten and Drummond [23]

2.1.1 Fast Algorithm

The first step of ORB feature detection is searching key points [4], and the Fast algorithm aims to extract the key points from the images, which is proposed by Rosten and Drummond[23]. The Fast algorithm has a highly robust and efficient performance.

The basic principle of the Fast operator [24] is that if there is a pixel whose value has a massive difference between the values of surrounding pixels, and the difference is larger than the specified threshold value. Then it can be considered as a feature point or corner. For the aspect of the grey image, fast operator thinks the value of the pixel and its difference between the value of neighbourhoods.

This algorithm can be described more clearly as the following processes. The process will iterate all the pixel points from the image. The pixel difference will be examined between a centre point and 16 surrounding pixel points, which located on the circle with a radius of 3 at this centre point. As Figure 2.1 shows, the centre pixel will be considered as a FAST feature point if its absolute grey value differs from 12 (or 9) consecutive pixel points, and the difference is higher or lower than a given threshold.

In order to improve the efficient of feature detection, the FAST algorithm [24] proposed the concept of segmentation test. The segmentation test doesn't iterate all 16 surrounding pixel points which are on a circle. Only 4 points are selected to test which locates on the vertical and horizontal directions of the ring, which include 1st, 9th, 5th and 13th pixels (As Figure 2.1 shows). The fast algorithm checks 1st and 19th pixels first. If their the difference of grey value satisfies the threshold, then it starts to check

5th and 9th pixels. If there are three or four pixels meet the threshold, then the centre pixels will be considered as a candidate of the corner. Contrarily, the centre pixel will be not labelled as a corner point. The candidate points, which have passed the segmentation test, will be processed in further testing. In FAST operator algorithm, the corner pixel requires at least 12 continuous different pixels on its surrounding circle. Therefore, if two or more of the four pixels from the vertical and horizontal directions do not meet the requirements, it can be directly determined that the point is not a FAST feature point. After preliminary filtering, the remaining eligible points are further tested by the FAST algorithm, and correct feature points are obtained after non-maximum suppression.

2.1.2 Brief Algorithm

The second step of the ORB algorithm is to transfer the key points into corresponding eigenvectors.

BRIEF is the short name for Binary Robust Independent Elementary Features (M. Calonder et al. [5]), which creates binary eigenvectors for a set of keypoints. Binary eigenvector, also known as a binary descriptor, which only contains 1 and 0. Each key point uses a BRIEF algorithm [5] to generate a binary eigenvector, which usually is 128 or 512 bits. Since the computer processes and runs the binary code inside, so that the most apparent advantage of using binary eigenvector is that it can be stored in memory very efficiently and computed quickly. Speed is critical for real-time applications. Therefore, binary eigenvector not only enables BRIEF algorithm very fast but also allows the BRIEF algorithm performed on devices which have limited computing resources.

Firstly, the BRIEF algorithm extracts a patch for a key point. The patch usually is a matrix around the key point which is in size of 9x9. And the matrix area performs the Gaussian fuzzy processing to avoid the influence of noise points. Then, many random pixel pairs are selected from this area. Each random pixel pairs contain two pixels: x and y, and both of them satisfy the Gaussian distribution. N is the length of eigenvectors. After selecting N pixel pairs in the matrix area, each pair compares the grey value of x and y. If the grey value of pixel x is higher than pixel y's, the amount of this pixel pair is 1. Contrarily, the value is 0. This comparison iterates N times, and the result of each pixel pair fills one bit of the eigenvector. Instead of calculating the descriptor, BRIEF [5] creates a binary eigenvector directly. Therefore, once binary eigenvector is obtained, the Hamming distance method could be used to perform the

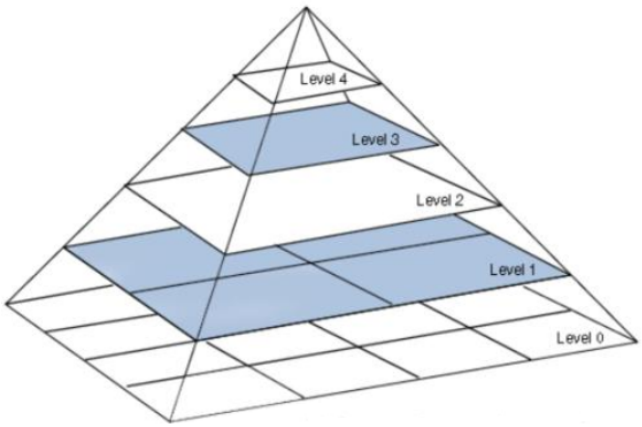


Figure 2.2: A image pyramid with five levels which proposed by Klein and Murry [14]

key points matching.

2.1.3 ORB Algorithm

2.1.3.1 Scale Invariance and Rotation Invariance

The original FAST and BRIEF algorithm [25] have advantages of low-computation and high-speed. However, the disadvantages are also visible. They are sensitive to noise and illumination and don't have the characteristic of rotation invariance and scale invariance. To increase the reliability of keypoint descriptor, ORB uses the image pyramid [14] and rBRIEF [5] to add features of scale invariance and rotation invariance.

2.1.3.2 Image pyramid and Scale Invariance

Image-pyramid [14] is a multiscale representation of a single image which is proposed by Klein and Murry [14]. An image pyramid consists of a series of different resolution versions of the original image. Each level of the pyramid is a downsampled version of the image from the previous level. Downsampling reduces the resolution and size of an original picture. Such as when the image downsamples to a 1/2 scale, the original image with a 4x4 pixel area will downsample to 2x2 pixel area. The downsampled image contains fewer pixels and is reduced in size by one-half, as Figure 2.2 shows.

ORB (Oriented Fast and Rotated Brief) algorithm [25] creates an image pyramid for the current frame and uses the FAST algorithm [24] to find key points in each resolution level of the image pyramid. Each level of the pyramid contains a smaller

version of the original image so that the size of any object reduces in higher pyramid levels. Objects must be unlikely to be precisely the same size in each image, especially dynamic objects that may be close to the camera at this time and far away from it at next time. In this way, the objects in the image will have a set of different scale of keypoints. By identifying key points at each level of the pyramid, the ORB algorithm can effectively discover key points for objects of various sizes, thus enabling partial scaling invariance.

2.1.3.3 RBRIEF and Rotation Invariance

For descriptors, the original RBRIEF features [5] do not have rotation invariance. To ensure the rotation invariance, the ORB uses the modified RBRIEF algorithm which is introduce by Rublee et al.[25].

Before generating descriptors, a size of 9x9 patch is selected around each key point, and the main direction θ of the key point is determined by calculating the zero-order and first-order moments. As formula (1) shows below, where $M_{01}$is zero-order moments of key points, M_{10} is first-order moments, x and y is selected pixel and the function I is predicate function.

$$M_{00} = \sum_{x,y} I(x,y), \qquad M_{01} = \sum_{x,y} yI(x,y), \qquad M_{10} = \sum_{x,y} xI(x,y)$$

$$\theta = atan2(m_{01}, m_{10}) \qquad (1)$$

In the patch, N pixel pairs are selected to make up a pixel pair matrix S. The rotation invariance (S_θ) can be obtained by rotating the matrix S to the direction of R_θ, as formula (2) shows.

$$S_\theta = R_\theta S \qquad (2)$$

After rotating the point pairs, the binary values are extracted according to the RBRIEF algorithm. However, instead of comparing one-pixel point with another, the RBRIEF chooses a patch around the key point to reduce the impact of noise.

Except for the ORB algorithm [25], there are lots of prominent places recognition algorithms. For example, Lowe [19] proposed the Scale-Invariant Feature Transform method (SIFT) which using the fast nearest-neighbor algorithm, the Hough transform

and least-squares to obtain the feature points. The SIFT descriptor contains the orientation information, and it has high accuracy on the feature detection. The Speeded up robust features (SURF) is another well know feature detection method which is mentioned by Bay et al. [1]. The SURF algorithm uses Hessian matrix to detect the features which improve the efficiency of extracting. As the experiments shows in [1], the SURF speed is three times of the SIFT, but its performance is comparable to SIFT. The SURF is good at handling blurred and rotated images, but not good at handling the changing of viewpoint and lighting. Even though the SURF and SIFT algorithm have satisfied extracting accuracy, however, the huge computation of SURF and SIFT leads to bad real-time performance. In comparison, the ORB algorithm has better performance of accuracy and real-time.

2.2 SLAM Algorithms

2.2.1 ORB SLAM2

ORB-SLAM2 is a famous real-time localization and mapping algorithm. ORB-SLAM2 algorithm was published by R. Mur-Artal and J. D. Tardós [22], which is well known for its high performance of robust and real-time. Comparing with the early version ORB-SLAM, ORB-SLAM2 also supports calibrated binocular camera and RGB-D camera. Most of SLAM algorithm researches bases on the ORB-SLAM2 framework. The ORB-SLAM2 is the foundational framework of three derivation SLAM algorithms: Dyna-SLAM, Fisheye-SLAM and GCNv2-SLAM, which are research points of this dissertation. Therefore, the framework and essential processes of ORB-SLAM2 will be introduced and analyzed briefly in this section.

The framework of ORB-SLAM2 consists of Tracking, Mapping, Relocalization and Loop closing module [22]. Those modules are implemented in three threads [33]: *Tracking*, *Local Mapping* and *Looping Closing*, as Figure 2.3 shows.

In the thread of tracking, the main tasks include extract ORB features [25] from the current frame, estimate the pose according to the last frame or initialize the pose with global relocalization. After obtaining the current pose, the tracking thread starts to track the reconstructed local map, optimize the current pose and figure out whether to insert the new keyframe.

In the part of Local Mapping, it mainly implements the construction of the local map. The tasks consist of inserting the new keyframe, validating and filtering the newly

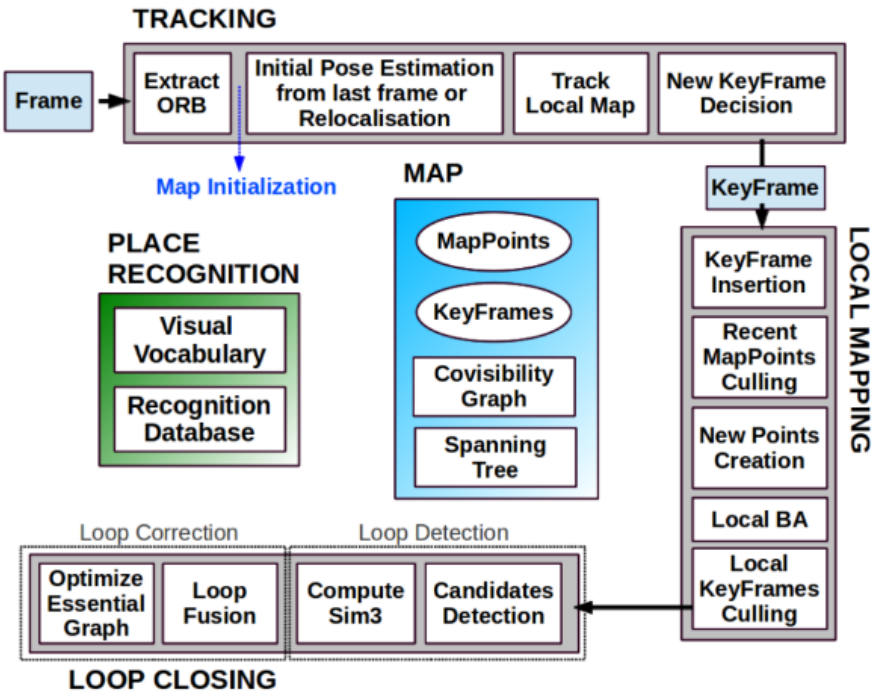


Figure 2.3: The framework diagram of ORB-SLAM2 proposed by by R. Mur-Artal and J. D. Tardós [22]

generated map points, creating the new map points for the local map and processing the local BA (Local Bundle Adjustment). Finally, filtering the inserted keyframes to remove the redundant keyframes.

The Loop Closing thread can divides two processes: Closed-loop Detection and closed-loop correction. Closed-loop Detection uses bag-of-word (BOW) [21] to find the closing loops and uses the sim3 algorithm to calculate the similarity transformation. In the next section, the flows and theories of three SLAM threads will be analyzed deeply.

2.2.1.1 Tracking

The tracking thread is the key point of this project, and the details of the tracking thread are introduced in this section. The introductions include pre-processing, tracking mode, tracking state, initialization for the monocular camera, module selection, recovery the camera pose, bundle adjustment (BA), stereo and RGB-D camera initialization, recovery the camera poses and relocalization.

Pre-processing

The SLAM system preprocesses the new input data before going into the tracking thread. The monocular camera mode requires the single RGB image. Binocular camera mode needs two RGB photos which are from each lens of the camera. RGB image and corresponding depth image is used for RGB-D mode [28]. The preprocessing consists typically of transferring the RBG image into a grey image, creating a new frame for the original image. Furthermore, it extracts the ORB features [25] and generates the image pyramid [14], which is the preparation of corner matching. Finally, using the correcting matrix to correct the position of feature points which on the new frame. After preprocessing the original image, the new frame is ready to enter the tracking thread.

Tracking mode

There are two tracking modes in the visual odometer system: Only Tracking and Simultaneous Localization and Mapping mode. In the Only Tracking mod, there are no key points and keyframes added to the local map. There is no local mapping and loop-closing in this mode and only tracking the points in the current existing map. Oppositely, the system in Simultaneous Localization and Mapping mode as default executes all the threads: *tracking*, *local mapping* and *loop closing*.

Tracking state

There are four tracking states in the tracking thread: *NO_IMAGES_YET*, *OK*, *LOST* and *NOT_INITIALIZED*. The tracking thread executes different operations according to the various state.

NO_IMAGES_YET means the system is waiting for a input of new image. Once the original image obtained, the state changes to *NOT_INITIALIZED*. The state of *NOT_INITIALIZED* shows the tracking thread is not initialized, and the system performs different initialization for the monocular camera and binocular camera or RGB-D camera. More initialization details will be introduced in the next section. After thread initialization, the tracking state changes into *OK* state. Then the system begins to pose estimation and map points tracking. The thread will always be in *OK* state if there is no frame loss or reset. The *LOST* state is detected, which presents fail to track the last frame. The current frame will trigger relocalization operations.

Initialization for monocular camera

In the tracking thread, the initialization of monocular camera and the binocular camera has a huge difference [33]. Because the depth of the monocular camera is unknown, and it needs to process a special initialization. Even after the initialization of the monocular camera, there is still a scale problem. Because, in initialization, the displacement of the first frame is taken as the unit length, and the depth and movement of the latter frames are computed and estimated according to this standard, so that scale problem is a theoretical defect of monocular SLAM.

The initialization of monocular SLAM requires two frames: the first frame becomes the reference frame and the second frame as the current frame. The initialization calculates the camera pose according to the matching points of the current frame, and the corresponding reference frame—furthermore, the RANSAC algorithm [38] is used in the initialization process. RANSAC is short for RANdom SAmple Consensus, and it eliminates wrong matches and optimizes the initialization results. In every iteration, the RANSAC synchronous computes the homography matrix (H) [8] and Fundamental matrix (F) [33].

The Homography Matrix (H) suits to deduct the matched points in planer space. As formula (3) shows, where X_c and X_r is the feature matrix of the current frame and the reference frame.

$$X_c = H_{cr} X_r \qquad (3)$$

As formula (4) shows, the Fundamental Matrix (F) normally computes the matched points in non-planer space, where X_c and X_r is the feature matrix of the current frame and the reference frame.

$$X_c^T F X_r = 0 \qquad (4)$$

The iteration times of RANSAC has a fixed number. Each iteration calculates a score S for the H and F models (As formula (5) shows below), respectively.

$$S_M = \sum_i \{\rho M(d_{cr}^2(x_c^i, x_r^i, M)) + \rho M(d_{cr}^2(x_c^i, x_r^i, M))\} \qquad (5)$$

$$\rho_M(d^2) = \begin{cases} \Gamma - d^2, \text{ if } d^2 < T_M \\ 0, \text{ if } d^2 \geq T_M, \end{cases}$$

Where M is module of Fundamental (F) or Homography (H), T_M is the distance threshold, the Γ is also a predetermined threshold. All the S_M scores are calculated respectively, And the scores are determined by calculating the sum of projection errors. The higher the score, the more trustworthy of a matrix under the current model is after completing all the iteration steps, choosing the modules with the highest score as the result of Homography (SH) and Fundamental module (SF) respectively.

Module selection (Homography matrix (H) or Fundamental matrix (F))

If the scene is planar, near planar, or has a low disparity, it can be solved by a Homography matrix. Conversely, for non-planar views where has enough disparity, the calculation can be done using the Fundamental matrix. So that the computing uses the following powerful heuristics (6).

$$R_H = \frac{S_H}{S_H + S_F} \qquad (6)$$

If Rh > 0.45. The homography matrix will be used to recovery of relative pose and 3D map points. Otherwise, using the fundamental matrix.

Recovery the camera pose

Once the best module obtains, the selected module can recovery the movement state to get the relative pose (R and T).

Suppose the homography matrix is the best module from RANSAC iterations, the eight motion hypotheses method, which is proposed by Faugeras and Lustman [9], will be used to recover the relative pose. It generates eight motion hypotheses (Eight R and T combination), and the best solution will be set as the pose parameters. The best solution has the minimum reprojection error and the maximum number of recovery 3D points.

In the case of the Fundamental matrix, the Essential matrix of the F matrix is decomposed by the singular value decomposition method [38]. It will obtain four motion hypotheses, and all the motion solutions triangulate the feature points [9]. Then it checks whether there is the best solution that all the points are in front of both cameras. If there is only one optimal solution with enough number of recovered points and enough disparity, the optimal solution will be taken as the relative pose of two frames.

After confirming the relative pose of the camera (R, t), we use the triangulation method to compute the three-dimensional position of matched points. Therefore, for each matching point, we have x = P * X, x' = P' * X. And x', X' presents the pixel

coordinates of the matching point in the first frame and second frame, respectively. And P, P' is the camera transformation matrix. Because of the scale problem, we use the homogeneous coordinates to compute the solution. The SVD method is used to get the homogeneous form of the three-dimensional point coordinates. The SVD decomposition has a least-squares effect to solve the problem caused by pose noise. Then the three-dimensional coordinates of the matching points are obtained by converting the homogeneous coordinates to the non-homogeneous coordinates.

Bundle Adjustment (BA)

Once the 3D position of matching points is obtained, the Bundle Adjustment (BA) [15] is applied to optimize the pose of camera. BA is essentially an optimization problem with the aim of minimizing the reprojection error. The reprojection error refers to the difference between the projection of a real three-dimensional spatial point on the image plane (The coordinate of real matching pixel points) and the reprojection of virtual points (The coordinate of 3D points we calculated) on the image plane. After optimizing the map points, the initialization of monocular camera tracking is finish and thread start to track.

Stereo and RGB-D camera initialization

Since binocular and RGB-D cameras do not require two adjacent frames to recover the map point depth. Therefore, the initialization process is very similar and simple. However, the stereo and RGB-D camera are quite different from the obtain method of depth.

In the case of RGB-D camera initialization, the depth information obtains from the depth map in a very easy way. Then the initialization will process series of operations which include building up the keyframe, adding the map points in the keyframe, calculating the BOW (Bag Of Word) vector [10] of the keyframe, adding the keyframe into the global map, building up map points and set its attributes (the keyframes who can detect this point, the optimal descriptor of this map points, the average observation direction and depth range of map points). Then adding those points to the global map, updating the common visual relationship between keyframes, adding keyframes to the local map thread, updating the local keyframes set and the local map points set.

In the case of stereo camera initialization, most of the initialization processes are similar, except the depth computation. The stereo camera uses the stereo matching algorithm to compute the depth information [22]. In the frame initialization process, the

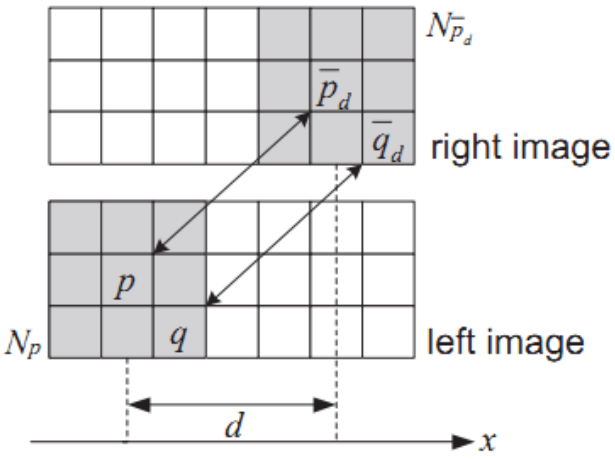


Figure 2.4: The SAD mathing algorithm: Reference searching windows (Np) and target searching window(Npd) when dp = d. (Proposed by Calonder et. al) [5]

algorithm receives two images (left and right camera image) and computes the image pyramids for each image. Then it uses two threads to extract ORB feature points from the left and right image, simultaneously. The OpenCV correction function is applied to correct the feature points by internal parameters. After obtaining the corrected feature points, SAD image matching algorithm is used to compute the matching points.

A SAD algorithm is a short form of Sum of absolute differences, which is the simplest matching method. Image block matching always uses it to evaluate the similarity of image blocks. It presents the sum of absolute value that the difference of all the corresponding pixels. Smaller SAD value means more similarity of image blocks.

In the ORB-SLAM2 stereo camera matching method, the first step is to build up the corresponding searching range table for the feature points of the left image, which presents a feature point will search a matching point in this strip area. Because it is possible that the matching points are not in the same polar line so that the match searching is not only on a horizontal line but also on a horizontal search band. As Figure 2.4 shows.

For each feature point of the left image, a matching point is found in the strip search area of the right image by the descriptor, and then subpixel matching is done by the SAD method. It traverses all possible matching points of the right image to find the best matching point (The minimum distance of the descriptor subpixels). The matching points will be removed if it has a large SAD matching deviation (Outliers).

Only successful matched points compute the corresponding depth. The depth information is stored in the mvDepth vector. Once the depth of map points is obtained, the similar initial operations as RGB-D camera processes in the stereo camera initialization.

Camera pose tracking

The system tracking state turns to OK after initialization processes, which presents the tracking thread allows to pose estimation and map points tracking [22].

The method of map points replace checking is utilized for the last frame. It aims to determine whether there are map points that could replace the map points in the current frame. If it has some matching map points, the two map points will be fused in one map point. After fusing the map points of the last frame and current frame. The tracking thread uses two pose estimation method: tracking with the reference keyframe and tracking with motion model.

The first method is using reference keyframe to trace. Firstly, computing the BOW vector [10] for the current frame, then matching this BOW vector with the reference keyframe's BOW vector. If the matched points are less than 15, the tracking will fail. When the match is greater than 15, the last frame's pose is used as the initial value of the current frame's pose, and the matching points and the pose will be optimized by using least squares. Finally, discarding the useless miscellaneous points (outliers), and tracking will succeed when the number of matches is greater than or equal to 10. Generally, the reference keyframe tracking method is performed when the motion model has not been established, or the relocalization has just been completed.

In the case of the motion model tracking [20], it assumed that it is a uniform speed motion model. The current frame pose is estimated using the pose and speed of the last frame by PnP method. Then it projects the last frame' s map points onto the current frame. If the number of matching points is not enough, the sampling size will be double on the current frame. Once the matching number satisfies the threshold, it will use the BA [15] and least-squares method to optimize the camera pose. After removing the optimized outlier points, the rest points will be utilized in the next tracking.

Even though the initial pose has obtained by the motion model or the reference keyframe, the initial pose is still not reliable. Because It only uses the information of two frames data. If the accuracy of the last frame is in low quality, the estimated pose of the current frame will not reliable either. Therefore, it is essential to take more information from frames. The current frame requires further matching and optimizing

with the local map and the local map is a set of the keyframes. As the movement of the camera, the new keyframes and 3D map points are added to the local map to maintain the local map. In this way, even if a lousy frame appears in the tracking process, we can still figure out the correct positions of the subsequent frames, using the local map. In the same way, the local map is also performed in pose optimization, and this is what the function *TrackLocalMap* do.

In the *TrackLocalMap* process, it updates the local map, which includes the local keyframes and local map points. Furthermore, the local map points are projected to the current frame. The *isInFrustum* function will check whether the projecting points are in the frustum of the current frame. The points will be removed if they are not in the frustum and the rest points will process the feature matching by projection. Once the new matching points are found, the further camera pose optimization will be performed by nonlinear least squares. If the number of matched interior points is greater than the threshold, the local map tracking is considered as successful.

Even though these steps are like *TrackWithMotionModel*, except that the *TrackWithMotionModel* matching pairs are obtained by inter-frame matching, while TrackLocalMap finds new and more matching points from the local map to optimize the position and posture further.

Relocalization

In the case of LOST state, it presents the tracking of the last frame has failed, and the current frame will perform the relocalization operation.

In relocalization implementation [20], it computes the BOW vector of the current frame and searches the similar candidate keyframes according to the BOW vector. After finding the candidate keyframes, it matches the current frame with the found candidate keyframes and computes the camera pose Tcw by RANSAC [38] and PnP algorithm [16]. In the PnP algorithm, several P4P and RANSAC iterations are performed, and the nonlinear least-squares method is used to optimize the pose. The iteration will stop until the iteration time has reached the limited number or a camera pose with enough inlier points is found.

If local map tracking is successful, the motion model and map points will be updated according to the new camera pose. The outlier map points will be removed from the map, and the reference keyframe will be updated (The reference keyframe is the keyframe with the most significant number of co-visible map points among all the keyframes). After updating the keyframes, the next step is to detect whether it is

necessary to add a keyframe currently. The determining is according to the following processes:
1. Check the tracking mode, whether it is the ONLY_TRACKING. The ONLY_TRACKING mode does not need to create a new keyframe.
2. If the local map has been suspended or received the delayed request, the local map will not add the keyframe.
3. The relocalization just has done, and there are not enough frames. So that the local map will not add the keyframe.
4. The frame limitation has reached since the last keyframe inserted. The local map allows adding the keyframe if it has long time without newly inserted keyframe.
5. The tracking thread is weak, and the local map expects the new keyframe to maintain the tracking.
6. If the local map is in idle state and it does not exceed the limited number of keyframes, the local map will add the new keyframe.

The *CreateNewKeyFrame* function is executed when a keyframe is allowed to insert. The main work of this function is to create a keyframe, add its interior points to the global map. Then the local mapping thread will receive the new keyframe. Furthermore, this keyframe will also be passed to loop closing thread after the processes of the local mapping thread.

2.2.1.2 Local Mapping

The local mapping [20] thread obtains the new keyframe from tracking thread, and the local mapping thread performs the further steps, which include processing and inserting the keyframes, eliminating the redundant map points and keyframes, and processing the local BA (Bundle Adjustment) to optimize the keyframes. as the figure shows below.

Once the new keyframe is detected, the function *ProcessNewKeyFrame* will perform several operations such as update the keyframe, compute the BOW of the keyframe and insert the keyframe into the map. But not all the map points in the current keyframe satisfy the condition. If at least three keyframes in the map could detect this point, the point will be matched to the points in other keyframes and created by the triangulation method. Those map points, which can't be observed by other keyframes, will be culled. After creating the new map points, the thread performs the local BA (Bundle Adjustment) to optimize the current keyframe, and it detects the redundant keyframes

and removes them. In this way, it is easier to control the compactness of the reconstructions and the complexity of the bundle adjustment.

Finally, keyframes are recorded in the database list at the end of all steps, and the tracking thread is allowed to add the keyframes.

2.2.1.3 Loop Closing

As the camera moves, the calculated camera pose and triangulated point cloud locations have the accumulative error, even though the local BA (Bundle Adjustment) and the global BA are performed to optimize the pose. The most effective way to eliminate the error is to find the closed-loop and optimize all the results based on the loop closing. The loop closing is a stronger and more accurate constraint than BA.

When the loop closing thread receives a new keyframe, the thread computes the similarity of the BOW vector of keyframe with the existed keyframe BOW vectors. It aims to determine whether the loop closing is found. Then it selects the continuous candidate keyframes from the keyframe database. The candidate keyframes are used to compute the similarity transformation matrix (Sim3) [29] with the current keyframe. If a suitable Sim3 is obtained, which means the loop closing is detected and the thread will perform further optimization such as loop closing fusion [14] and essential graph optimization [20].

There are also several outstanding visual SLAM algorithms which are similar to the ORB-SLAM2. For instance, Strasdat et al. [30] provided a monocular SLAM which is used by the large-scale camera. It uses the FAST algorithm to extract the features and performs motion-only BA to optimization. Furthermore, the pose graph optimization is used to adjust the scale drift of the camera. Song et al. [27] proposed a visual odometry algorithm and it uses the ORB descriptor to extract the features. The temporal sliding window BA method is used to optimize the pose. However, the global reconstruction is not implemented in this visual odometry system, and the global map is not reused in the loop closing process. The stereo large-scale direct SLAM (LSD-SLAM) is presented by Engel et al. [7]. Unlike the ORB-SLAM, the LSD-SLAM doesn't use the feature points to track, and it is a semi-dense direct algorithm. It aims to perform robust tracking in the motion blur or poorly textured environments. The benefit of the semi-dense direct algorithm is that the tracking will not be affected by the quality of feature points. However, it could be affected by the unmodeled effects such as rolling shutter.

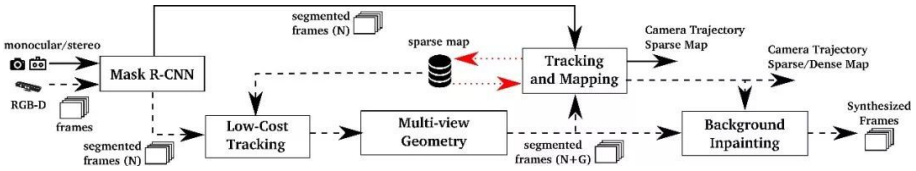


Figure 2.5: The framework diagram of Dyna SLAM proposed by Bescos et al. [2]

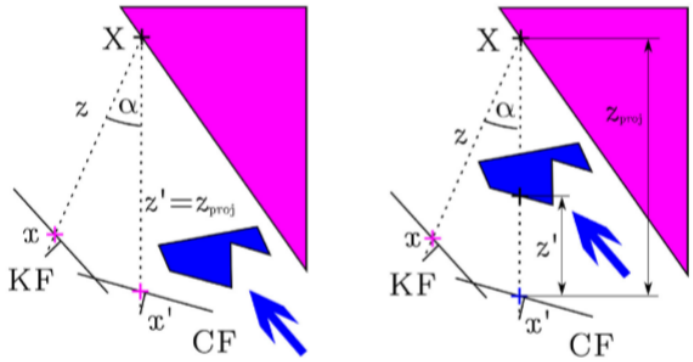
2.2.2 Dyna SLAM

The Dyna-SLAM is a derivative version of ORB-SLAM2 [22], which proposed by Bescos et al. [2]. It bases on the ORB-SLAM2 framework to track and mapping and performs the MASK R-CNN and the multi-view geometry to avoid the effect of the dynamic object. According to the experiment result, the Dyna-SLAM gets high-quality performance tracking and mapping results in the dynamic scene.

In the Dyna-SLAM system (As the Figure 2.5 shows), it proposes three highlights according to the scene with dynamic objects.

The first improvement is only using the convolution neural network (CNN) to segment the potential dynamic objects. The Mask R-CNN [12] is performed to find the semantic segmentation and instance tags of the dynamic objects. The input of Mask R-CNN is the original RGB image with the size of L * W * 3(Where L and W is the length and width of the image, respectively). The output result is a matrix with the size of L * W * N, where the N presents the number of detected dynamic object. In the current CNN module, the potential targets include person, car, plane, boat and most type of animals [2].

The second method is using Mask R-CNN and multi-view geometry [34] to segment the dynamic object. The multi-view geometry method aims to find those dynamic objects which are not contained in the potential objects of Mask R-CNN such as active books or bottles. In the multi-view geometry method of active object detecting (As the Figure 2.6 shows), when a new frame is received, some previous frames with the most overlap will be collected, and their key points x will be projected onto the current frame, and projected points names x'. Meanwhile, the depth of projecting points z_{proj} and corresponding 3D points X are also obtained. The points x, x' and X can form the angle xXx' called α. When the angle of α is greater than 30 degrees, the point will be ignored which could affect the detecting. The depth information z' of the remained points could be obtained from depth measurement or depth map. The projected point x' will be considered as part of the dynamic object if the difference of z' and z_{proj} is



(a) Keypoint x' belongs to a static object ($z' = z_{proj}$).

(b) Keypoint x' belongs to a dynamic object ($z' \ll z_{proj}$).

Figure 2.6: The multi view geometry algorithm Dyna ORB-SLAM proposed by Bescos et al. [2]

greater than the set threshold. In the implementation of multi-view geometry [2], the threshold is set as 0.4 with the most accuracy detecting result.

The third highlight point is background painting [2]. It repairs the obscured background area with static information from the previous frames. In the experiment, 20 previous frames are selected to repair the background of the current frame. However, the repairing process relies heavily on the estimated pose, if the estimated pose is inaccurate, there will be a large gap between the results of the repair and the actual situation.

Similarly, there are several types of research that aim to perform the robust SLAM system in the dynamic scene. Tan et al. [35] presented a new monocular SLAM system which suits the dynamic scene. The algorithm projects the keyframe onto the current frame to identify the dynamic object. This method reduces the effect of the occlusions so that the performance is improved on the dynamic tracking.

Sun et al. [32] proposed an RGB-D SLAM which adapts the activity object. It uses the ego-motion compensated method to detect the differencing of the dynamic objects. Then the particle filter is used to enhance the motion tracking. Finally, the Maximum-a-posterior (MAP) estimator is used to optimize quantized depth images. And the SLAM performs the robust result in the dynamic scene.

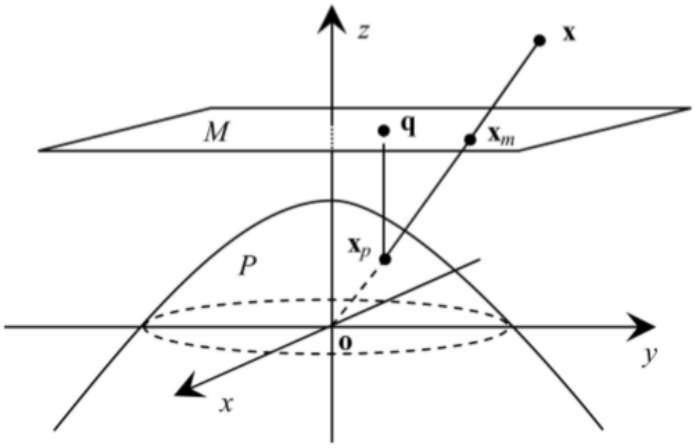


Figure 2.7: The enhanced unified camera model proposed by Khomutenko et al.[13]

2.2.3 Fisheye SLAM

Liu et al. [18] provides an omnidirectional camera SLAM solution called Fisheye SLAM. It modifies the projection function of ORB-SLAM2 [22] with the enhanced unified camera model (EUCM) [13]. The EUCM module makes the SLAM system to suit the omnidirectional camera such as the fish-eye camera.

Comparing the normal pinhole model, the EUCM model has more projection parameters (α and β) which present the radial distortions. And the EUCM model doesn't need further distortion projecting. In the EUCM model (As the Figure 2.7 shows), the 3D point x is projected onto the omnidirectional camera lens which is a second-order projection surface P. The location of the projected point called x_p. And the x_p is further mapped to q, which is on the M plane. The position of q can be expressed as formula (7).

$$q = \begin{bmatrix} \frac{x}{\alpha\rho+(1-\alpha)z} \\ \frac{y}{\alpha\rho+(1-\alpha)z} \\ 1 \end{bmatrix} \qquad \rho = \sqrt{\beta(x^2+y^2)+z^2} \qquad (7)$$

Where x, y, z is the camera coordinates of the 3D point x, α and β present the radial distortions of the camera lens.

To get the position of image point u(u, v), the point q, which on the M plane, is mapped onto the image plane. The mapping function uses the normal pinhole camera

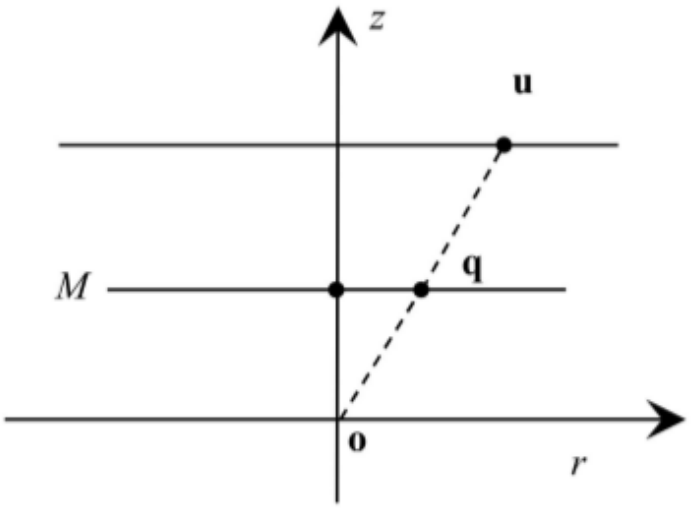


Figure 2.8: The pinhole camera projection model proposed by Liu et al. [18]

model currently. As the Figure 2.8 and formula (8) shows, where f_x, f_y are the focal lengths, and c_x, c_y are the principal points.

$$u = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x, 0 \\ 0, f_y \end{bmatrix} \times \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \tag{8}$$

Therefore, the position of image points can be computed by a 3D point coordinate as the formula (9) shows, where where f_x, f_y are the focal lengths, and c_x, c_y are the principal points and α and β present the radial distortions of the camera lens.

$$\pi(\vec{x}) = \vec{u} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x, 0 \\ 0, f_y \end{bmatrix} \times \begin{bmatrix} \frac{x}{\alpha\rho+(1-\alpha)z} \\ \frac{y}{\alpha\rho+(1-\alpha)z} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \tag{9}$$

And the unprojection function could be expressed as formula (10), where f_x, f_y are the focal lengths, and c_x, c_y are the principal points and α and β present the radial distortions of the camera lens.

$$\pi^{-1}(\vec{u}) = \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix}$$

(10)

$$\text{where } m_x = \frac{u-c_x}{f_x}, \quad m_y = \frac{u-c_y}{f_y}, \quad m_z = \frac{1-\beta\alpha^2 r^2}{\alpha\sqrt{1-(2\alpha-1)\beta r^2}+(1-\alpha)}$$

where f_x, f_y are the focal lengths, and c_x, c_y are the principal points and α and β present the radial distortions of the camera lens.

In the fisheye-SLAM system, the author implements a monocular case of omnidirectional camera SLAM. The projection function is changed to EUCM model in the Map initialization and relocalization algorithm. In the experiments [18], the Fisheye-SLAM shows a better robust performance than the original ORB-SLAM2, on the omnidirectional camera database.

Several omnidirectional camera SLAMs were proposed during these years. And the indirect method is a popular research area. J. Li et al. [17] transplants a spherical model in a SLAM system, and this system is compatible with the full-view images, but the speed performance is not good.

S. Wang et al. [39] provides an extended version of ORB-SLAM framework with EUCM model and the semi-dense depth map. However, the performance of the system is not stable.

2.2.4 GCNv2 SLAM

The GCNv2 SLAM is also based on the ORB-SLAM2, which is provided by Tang et al. [36]. The main contribution of GCNv2 is using the more effective GCN descriptor to replace the original ORB descriptor. The GCN descriptor is generated by the geometric coherence network (GCN). The GCN is used to extract feature points and descriptors from images. The network framework in the GCNv2-SLAM consists of two parts: the SuperPoint network [6] and the deep metric learning.

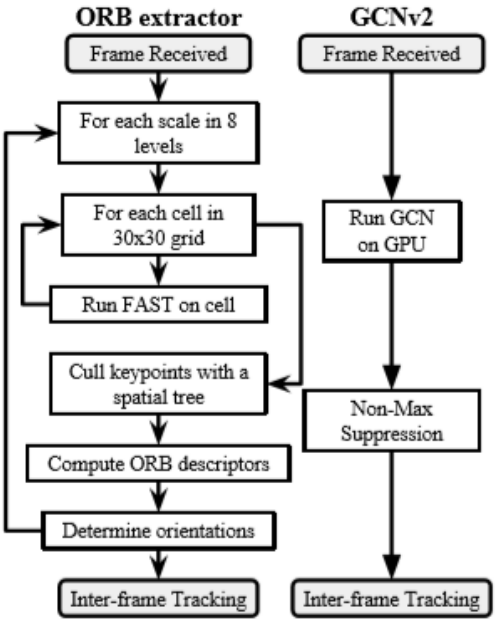


Figure 2.9: Illustration of the original ORB-SLAM2 and GCN-SLAM proposed by Tang et al. [36]

The SuperPoints network is a feature point detection and descriptor extraction method based on self-monitoring training, which was proposed in an article published by DeTone et al. [6] in 2018. Since the loss function is calculated by pose, the extracted feature points and descriptors have more advantages in the pose calculation.

The deep metric learning aims to feature association [36]. The metric learning bases on the ground-truth information which is provided by the camera dataset. Since the accurate pose provided by ground-truth is used in the training process, the accuracy and stability of pose calculation are better.

The GCN descriptor is a binary feature vector with 256 bits which is same as the ORB feature. Comparing the ORB extraction process (As Figure 2.9 shows), the GCNv2 extraction reduces the complex computation of image pyramid and ORB descriptor which greatly improves the efficiency of computation.

As the results of the experiments Tang et al. [36], the GCNv2 performs high effect on the embedded low power platform Jetson TX2. And it has proved that the GCNv2 descriptor not only remains the accuracy of tracking but also reduces the computation of feature extraction.

The GCNv2 SLAM is a variant of GCN SLAM which is also proposed by Tang et

al. Tang et al. [37]. The main difference is the structure of the GCN network. The GCN SLAM network consists of the CNN (Convolutional Neural Network) and an RNN (Recurrent Neural Network) network. The CNN network is used to extract the feature and feature association. The RNN is performed to predict the position of the features. Comparing the performance with GCNv2 SLAM, the accuracy of GCN is the same as GCNv2. However, since the computation of GCN is much larger than GCNv2 so that the speed performance GCNv2 is much better than GCN.

Because the GCNv2 descriptor has the benefit of low computation cost, high accuracy and efficiency, therefore, the advantages of GCNv2 SLAM are the focuses of SLAM advantages integration.

Chapter 3

Methodolody

3.1 Implementation

3.1.1 The stereo and monocular case of GCNv2 SLAM

The original GCNv2 SLAM only implements the tracking of RGB-D camera, which is the simplest tracking mode. Since the RGB-D camera datasets consist of the RGB images and the depth images, the system could extract the depth information from the depth images directly. However, the tracking mode of stereo and monocular camera are quite different, since the depth data can't be obtained instantly. The stereo camera mode could obtain the depth data by computing the disparity from the left image and corresponding right image. And the monocular camera tracking mode gets the depth data by calculating the related pose from the initial pose. Therefore, it is worth to extents the stereo and monocular mode of GCNv2 SLAM. In this work, the stereo and monocular camera mode of GCNv2 SLAM are implemented, which are performing the same GCNv2 models to extract the GCN descriptors for the images.

The framework of GCNv2 SLAM is same as the ORB SLAM2, and it consists of Tracking thread, Local Mapping thread and Loop Closing. The implementation is mainly in the tracking thread. More specifically, the optimization part is feature extraction.

The ORB SLAM2 implements the *ORBExtract* function to detect the ORB feature points from the current frame. The image pyramid is generated when the frame is initialized for an input image. The image pyramid contains different scales of the original image. The ORB SLAM2 performs the ORBExtract function to extract the feature points from each level of the image pyramid. In this way, the ORB descriptors

offer the property of scale-invariant. However, the GCNv2 SLAM uses the geometric coherence network (GCN) to generate the GCN descriptors directly, and the GCN descriptor has the same size as the ORB descriptors (256 bits). The GCNv2 SLAM inherits the mostly framework of ORB SLAM2, and it modifies the feature extraction methods and its relative function. Therefore, the implementation of the stereo camera mode also similar to the ORB SLAM2 framework.

For the convenience of comparison, two feature point extraction methods are implemented in the stereo camera mode of GCNv2 SLAM. Suppose the value of parameter "USE_ORB" is 1, the system will use ORB feature points to recognize the corners. Conversely, the tracking will perform the GCNv2 feature points. Once the tracking mode is confirmed as the GCNv2 tracking mode, the system initializes the new frame for the corresponding image. Because of GCNv2 doesn't perform the image pyramid, so that the frame initialization method is modified. In the frame initialization process, the *ExtractGCN* function is performed to extract the GCNv2 descriptors from the left image and the right image, respectively. The system puts the original image into the GCNv2 network to get the GCNv2 descriptors. Therefore, the GCNv2 key points don't contain the information about the image pyramid, such as pyramid level, scale factors. In other words, only the original image is processed by the GCNv2 model. The difference is that ORB descriptors are obtained by computing from each level of an image pyramid. In this way, the GCNv2 SLAM reduces the computation requirement of key points extraction.

The main task of *ExtractGCN* function, which is similar to ExtractORB in ORB SLAM2, aims to extract the GCNv2 key points from the image and create the corresponding 256 bits descriptor. GCNv2 SLAM provides models for two different resolutions (320x240 and 640x480). The default mode uses is the 320x240 model, which is less computationally-demanding and was found to perform well enough for most applications. The input image will be re-sized into the required size before putting it into the neural network model. The GCNv2 model will output the key point set and a corresponding descriptor set. After extracting the GCNv2 key points, the system continues initializing the frame.

Because the structure of GCNv2 key points differs from the ORB key points, so the *computeStereoMatches_GCN* function is implemented to initialize of the stereo camera frame. It aims to find the match pairs of the GCNv2 feature points from the left image and the right image. And the depth data could be computed from the matched pairs. The used stereo matching algorithm is the same as the matching algorithm in

the original ORB SLAM, which has been introduced in the background part. Since all the GCNv2 key points are extracted from the original image so that the value of scale factor always equal to 1, and the pyramid level defaults as 0. Therefore, in the matching algorithm, the window size of the sliding window is still in the same size, which differs from the ORB stereo matching algorithm. In the ORB stereo matching algorithm, the size of the searching window is according to which level of pyramid that the feature point is. If the ORB key point is extracted from a high layer of the image pyramid, the size of the searching window will smaller.

As same as the RGB-D camera tracking of GCNv2 SLAM, the frame matching algorithm method is further modified. In the original ORB SLAM2, the SearchByBoW function is performed to find the matching relationship between the keyframes and the current frame. The new matching method is used in the GCNv2 tracking mode, which is implemented by the standard nearest neighbor searching algorithm. This algorithm uses the descriptors of the keyframe and the current frame to compute the matching directly. If the hamming distance of the candidate matching pair is less than the limited threshold, this pair will be considered as the matching pairs. It works for testing the accuracy of the GCNv2 descriptors and reducing the computation of frame matching. This new matching algorithm is performed in the *Relocalization* process and the *TrackReferenceKeyFram* process.

The implementation of the monocular camera mode is similar to the stereo camera mode. However, since the monocular camera can't get depth information from the single image, so the initialization of monocular camera mode of GCNv2 SLAM is differing from the initialization of stereo camera mode.

In the model of a monocular camera, the initialization processes are also modified because the structure of GCNv2 key point is not the same as the original ORB key points. As introduced in the background, the monocular initialization uses two images to estimate the pose and orientation of the monocular camera.

The GCNv2 network generates the GCNv2 descriptors for two images respectively. Most of the initialization processes use the image pyramid parameters to perform initialization operation. For instance, the function *SearchForInitialization*, who finds the matching pairs from the reference frame and the current frame, uses the octave level information to filter the unreasonable candidate pairs. The inside function *GetFeaturesInArea* uses octave level and scales factor to select the key points, which is localizing in the assigned frame area, and so on. All the monocular initialization processes are modified to adapt GCNv2 key points. In the GCNv2 mode, the value of the octave

level is always 0, and the scale factor is 1.

The most modification takes place in the tracking thread. The framework of loop closing thread and local mapping thread is as same as the ORB SLAM2. This is other advantages of the GCNv2 SLAM that it is well organized and easy to transplant.

3.1.2 The RGB-D camera case of Dyna + GCNv2 SLAM

The Dyna + GCNv2 SLAM was implemented which bases on ORB SLAM2. The Dyna SLAM uses the Mask RCNN to detect the potential dynamic objects from the current image, and the multi-view geometry algorithm is performed to detect the non-potential objects. The feature points which locates in the objects area will be removed from the key point set so that the dynamic objects will not affect the map establish. However, removing the feature points leads to the reducing of the number of key points obviously. Furthermore, the performance of tracking is also affected by the quality and quantity of key points. Therefore, to improve the performance of the Dyna SLAM, it is worth to apply a new feature point with higher quality and effectiveness. The GCNv2 feature point, which has the properties of low computation and high correspondence prediction, is a suitable replacement choice.

The RGB-D camera mode of Dyna SLAM performs both of Mask RCNN module and multi-view geometry algorithm. So that it requires vast computation to detect the dynamic objects, and it removes lots of feature points from the key point set, and the system could fail to track because there is not enough key points to use. Therefore, RGB-D camera mode of Dyna + GCNv2 SLAM is implemented, which aims to perform the GCNv2 key points to improve the performance of the original Dyna SLAM. The Dyna + GCNv2 SLAM consists of the Mask RCNN network, multi-view geometry algorithm and the GCNv2 network. As same as the ORB SLAM2, it consists of tracking thread, local mapping thread and loop closing thread. When the system is initialized, it performs several processes such as initializing three threads and Mask RCNN model and loading the binary GCNv2 vocabulary. The binary GCNv2 vocabulary file replaces the original text vocabulary file. The vocabulary file generates the BOW vector of the frame, which is beneficial for the frame matching. The binary GCNv2 vocabulary suits the GCNv2 descriptors, and the binary vocabulary provides more efficiency than the text vocabulary.

The main modification is in the tracking thread. When the system receives a new RGB image and the depth image, the RGB image is processed by the trained Mask RCNN model, and the network outputs the masks of the potential objects. All images

will be resized to satisfy the input requirement of the GCNv2 model. Then the resized image, the depth image and corresponding masks are used to initialize the frame. In the frame initialization process, the GCNv2 network model is used to extract the GCNv2 feature points from the RGB image.

The GCNv2 network outputs the GCNv2 descriptor for each detected key point. After extracting the GCNv2 key points, the system iterates each of the key point to check out whether its position is in the mask area. If so, this key point is considered belonging to a dynamic object, and it will be deleted from the key points set. The set of descriptors also has the same operation. The rest GCNv2 key points are used to tracking and mapping. Then the GCNv2 key points extract the corresponding depth data from the depth image. Once a new frame created, the tracking thread performs the multi-view geometry algorithm to further dynamic object detection. As introduced in the background part, this algorithm aims to detect the dynamic objects which can't be detected from the Mask RCNN model. The new detected active GCNv2 feature points and descriptors are removed too. Up to now, the key points set, and corresponding descriptors set only contains the feature points information of static object or background. Then the normal tracking process is performed to track the camera pose according to the current keypoints.

Furthermore, like the stereo camera tracking mode of GCNv2 SLAM, the frame matching method is changed in the *Relocalization* and the *TrackReferenceKeyFrame* process. Finally, the *InpaintFrames* function is performed to repaint the dynamic object area using the static background from the previous frames. The local mapping thread and the loop closing thread remains as the ORB SLAM2 formation.

3.1.3 The monocular case of Fisheye + GCNv2 SLAM

The Fisheye + GCNv2 SLAM also bases on the structure of the ORB SLAM2, which further implants the GCNv2 model and the enhanced unified camera model (EUCM). In the original Fisheye SLAM, it presents the monocular camera SLAM algorithm which adapts the omnidirectional cameras. It performs well on the fisheye camera database. The most apparent advantage of the omnidirectional SLAM is that the broad field of view (FOV) camera could catch more objects and scenes than the ordinary camera, which means more feature points could be extracted from the same location because the SLAM algorithm performs the tracking and mapping operations according to the extracted feature points. Therefore, the more detected feature points are found,

the more accuracy and robust performance the system will have. Even though the Fisheye SLAM uses the EUCM to compatible the omnidirectional camera, but the more feature points means the higher computation demand. High computation also presents low efficient and low portability. Therefore, the original fisheye camera SLAM is not well compatible with the low computation platform. However, the GCNv2 SLAM could provide a high-quality solution for low computation platforms. In this project, the monocular camera mode of the fisheye + GCNv2 SLAM is completed to satisfy high accuracy, low computation and omnidirectional camera compatibility simultaneously.

As introduced in the background, the difference between GCNv2 SLAM and ORB SLAM2 is that the ORB algorithm computes the image pyramid to extract the feature points. Every pixel in each level of the pyramid is iterated to extract the feature points, which requires a vast computation to satisfy the real-time and accuracy. However, the GCNv2 descriptors are extracted only from the original image by the GCNv2 network model. As the GCNv2 SLAM result shows, the GCNv2 descriptors not only has the property of the low computation but also obtains a high accuracy performance. Therefore, GCNv2 descriptor is an ideal solution for the Fisheye SLAM to reduce computation and improve the performance of accuracy and speed.

The Fisheye + GCNv2 SLAM inherits the most of ORB SLAM2 frameworks, except the modification of the tracking thread. The map initialization and relocalization parts are modified with the EUCM. Furthermore, similar to the other implementation, Fisheye + GCNv2 SLAM also presents two key points extraction methods (GCN and ORB), which is convenient to compare the difference of performance.

When the tracking thread initializes, the system starts to initialize the pose of the monocular camera and the position of the map points. The monocular camera initialization requires two images. The first image is set as the reference frame, and the second image is set as the current frame. The GCNv2 network model is performed to extract the GCNv2 feature points and descriptor from the two frames (The processes are the same as the previous implementation). The initialization uses feature points matching algorithm to find the matching pairs from two frames (The reference frame and the current frame). When enough matching pairs are obtained, the initialization processes will use matching pairs to recovery the initial pose. The corresponding homography matrix and the essential matrix are computed to select the best recovery matrix. The camera poses will be calculated from the chosen matrix. Once the initial pose is confirmed, the map points are also triangulated into real-world coordinate. In map

initialization processes, the mapping functions use the EUCM projection method (as the background introduced) rather than the standard pinhole camera mapping methods. Some operations, which contain the projection or reprojection methods, are modified to the EUCM model, such as the computing the homography matrix and the essential matrix, checking the map point whether it is in the area of fisheye camera FOV, triangulating the map points into the real-world coordinate, and so on.

Another application of EUCM projection in the tracking thread is the relocalization process. The relocalization process aims to find a keyframe from the candidate keyframes to restore the camera pose. And this keyframe is most similar to the current frame. The frame matching algorithm is by projecting the points of the current frame onto the candidate frames and check out which candidate frame has the most significant number of the matching inliers since the original projection method adopts the pinhole camera model so that the projection method is changed into EUCM mode, to compatible the omnidirectional camera.

In the local mapping thread, the EUCM projection method is performed to create the map points in the local map. The projection method is the same as the initialization by using the triangulation method to compute the position. And the local BA is also modified to fit the EUCM model. The situation is similar to the loop closing thread, the bundle adjustment (BA) projection function changes to EUCM model.

Chapter 4

Evaluation

In this chapter, the stereo camera mode of GCNv2 SLAM, monocular camera mode of Fisheye + GCNv2 SLAM and RGB-D camera mode of Dyna + GCNv2 SLAM are evaluated. Three open-source datasets are used to evaluate the performances of achieved SLAM algorithms which includes TUM RGB-D datasets, TUM Visual-Inertial Odometry datasets (TUM VI) and EuRoC MAV datasets. Corresponding SLAM algorithms run each sequence of selected datasets at least five times, and the median results are used to calculate the performance of each SLAM algorithm. The SLAM evaluation tool, EVO, is performed to evaluate the estimated trajectories. Furthermore, the results of new achieved SLAM algorithms will compare with the results of original SLAM algorithms. In this section, the introductions of the three datasets are presented. Then the evaluation tool (EVO) and its essential metrics are introduced in detail. Furthermore, three achieved SLAM algorithms are evaluated and analyzed respectively.

4.1 The introduction of datasets

4.1.1 TUM RGB-D Dataset

The TUM RGB-D dataset [31] is used to evaluate the performance of the RGB-D camera mode of GCNv2 SLAM. The TUM datasets are captured from the Kinect camera, and the dataset consists of color images, depth images and the ground-truth trajectories. All the color images and depth images are PNG images which in the size of 640x480 in each image. The Ground-truth trajectory records the real poses of the motion sensor. It consists of the timestamps, corresponding position and orientation of the camera optical center. The Ground-truth trajectory files are formatted into standard

TUM dataset formation ("*timestamp tx ty tz qx qy qz qw*"). The timestamp provides the time metric to identify the pose and orientation. The *tx, ty, tz* presents the real-world position of the camera. The *qx, qy, qz, qw* offers the direction of the camera optical center. Four dynamic TUM RGB-D sequences are used to evaluate the RGB-D camera case of the Dynamic + GCNv2 SLAM which includes *setting_xyz*, *walking_halfsphere*, *walking_rpy* and *walking_xyz*.

4.1.2 TUM Visual-Inertial Odometry Dataset

The TUM Visual-Inertial Odometry [26] (TUM VI) dataset provides several types of omnidirectional camera sequences for the Visual Intertial odometry algorithms. The sequences are captured from different environments, such as the room, corridor, outdoors and slides. All the sequences consist of consistent timestamps, IMU and ground truth of the camera trajectories and omnidirectional camera images. The ground-truth trajectories are stored into the standard TUM formation ("*timestamp tx ty tz qx qy qz qw*"), which is same as the TUM RGB-D dataset. And the omnidirectional camera image is in the size of 512x512. Most of the sequences only have ground-truth poses for the start and end segments, which is not suitable to evaluate the entire trajectory. Therefore, four rooms sequences (*Room1*, *Room2*, *Room3* and *Room4*) are used to assess the performance of Fisheye + GCNv2 SLAM, because their ground truth poses are available in the entire trajectory.

4.1.3 The EuRoC MAV Dataset

The EuRoC MAV dataset [3] is performed to evaluate the performance of stereo camera mode of GCNv2 SLAM. The datasets are collected on-board a Micro Aerial Vehicle (MAV). It consists of stereo images, synchronized IMU measurements, and accurate motion and structure ground-truth. The sequences of EuRoC dataset are collected from two different scenes: Machine Hall and Vicon Hall. Four sequences (*MH01*, *MH02*, *V101* and *V201*) are selected to evaluate the performance of stereo camera mode of GCNv2 SLAM.

4.2 Evaluation tools and metrics

EVO [11] is a popular SLAM evaluation tool which could compare and evaluate the trajectory of the SLAM algorithms. It generates the trajectory comparison diagrams

and computes two accuracy indicators: the absolute pose error (APE) and the relative pose error (RPE).

Absolute Pose Error, which is known as the absolute trajectory error, compares the estimated and reference trajectory and calculates statistics for the entire track, which is suitable for evaluating the global consistency of tracking result. The EVO algorithm first aligns the estimated value and the ground truth value, which bases on the timestamps of the real-world position. Then it calculates the difference between each pair of positions. The differences are called absolute pose error.

Differ from the absolute pose error, the Relative Pose Error (RPE) doesn't compare the absolution positions, and it only compares the amount of motion changing (posture increment). Relative positional errors can provide local tracking accuracy, such as the drift or rotation drift per meter of the SLAM system. The relative pose error is obtained by calculating the difference between the amount of pose change on the same two timestamps. Similarly, after aligning with the timestamps, the amount of change of the ground-truth and estimated position are calculated at the same time interval, respectively. Then it computes the difference of changes to obtain the relative pose error. This standard is suitable for evaluating the drift of the camera system.

The EVO algorithm provides the multiply evaluation methods to presents the errors for RPE and APE. The result contains Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Standard Deviation (SD), Sum of Squared Errors (SSE), Max value of errors (max), Mean value of the errors (Mean), Median value of errors (Median), Minim value of errors (min). In this project, the RMSE, Mean and SD of relative pose errors are used as the evaluation standard.

Root Mean Square Error (RMSE) is the square root of the mean square of error deviations. It is used to measure the deviations between estimated values and ground truth values. As Formula (11) shows below, the less RMSE value means the accuracy of tracking is better, where m is number of the estimated poses, the x_i present the i th estimated position and y_i represents the i th ground truth position.

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_i - y_i)^2} \qquad (11)$$

Mean Absolute Error (MAE/Mean) is the average of absolute errors, which is another metric to reflect the actual situation of estimated errors. The expression is as Formula (12) shows. Where m is number of the estimated poses, the x_i present the i th estimated position and y_i represents the i th ground truth position.

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|x_i - y_i| \qquad (12)$$

Standard deviation (SD) is the arithmetic square root of the variance. It is used to measure the degree of dispersion of a set of errors. It presents the stability of drift errors in this project. The expression is as Formula (13) shows below. Where m is number of the estimated poses, the x_i present the i th estimated position and u represents the mean of estimated positions.

$$SD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - u)^2} \qquad \text{where } u = \frac{1}{N}\sum_{i=1}^{N}x_i \qquad (13)$$

In this project, the metrics of relative pose error (RPE) is used to evaluate the performance of the SLAM algorithm. Because the RPE contains not only the translation drift errors but also the rotation drift errors. And the APR only provides the translation drift errors. Furthermore, the initial position could affect the APE results obviously, and the estimated position and ground truth position is typically not in the same coordinate. Then the achieved SLAM algorithms require both of the translation errors and rotation errors to evaluate the performance. Therefore, the Root Mean Square Error (RMSE), Mean and Standard Deviation (SD) of the relative pose error (RPE) are performed to test and evaluate the SLAM algorithms.

4.3 Evaluation of achieved SLAM algorithms

4.3.1 The evaluation of stereo camera mode of GCNv2 SLAM

The stereo camera mode and monocular camera mode of GCNv2 SLAM are realized in this project. The implementation of monocular case mode is like the Fisheye + GCNv2 SLAM so that the evaluation of monocular camera mode of GCNv2 SLAM is not processed. Therefore, only the evaluation of stereo camera mode of GCNv2 SLAM is performed in this section.

Four EuRoC stereo camera datasets are used to test and evaluate the stereo camera mode of GCNv2 SLAM, which include the sequence of *MH01*, *MH02*, *V101*, *V201*. The quantitative comparisons are performed between the stereo camera mode of ORB SLAM2 and GCNv2 SLAM by selecting the different feature point extraction mode. All the sample cases are run five times, and the median trajectory result is assigned to calculate the relative pose error among the ground truth trajectory.

Table 4.1: COMPARISON OF THE MEAN, RMSE AND SD OF THE RPE OF ORB SLAM2 AGAINST GCNv2 SLAM FOR STEREO CAMERA, WITH FULL TRANSFORMATION POSE RELATION (TRANSLATION AND ROTATION) WITH UNITS IN METERS(m)

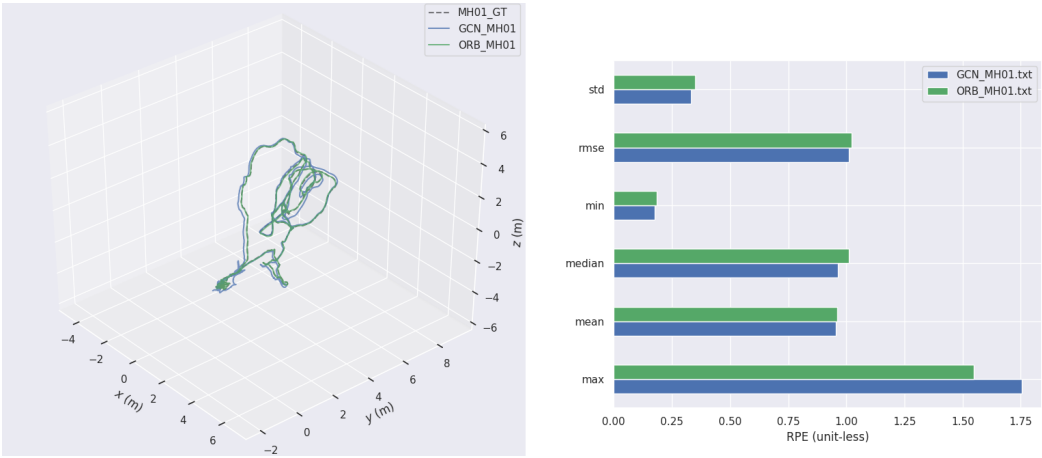
Metrics	MEAN		RMSE		SD	
Descriptor	ORB	GCNv2	ORB	GCNv2	ORB	GCNv2
MH01	0.959899	**0.953924**	1.021960	**1.010426**	0.350709	**0.333186**
MH02	**0.864955**	0.881632	**0.944155**	0.950140	0.378525	**0.354230**
V101	1.237284	**1.232852**	1.312057	**1.309289**	**0.436603**	0.440777
V201	1.157378	**1.142645**	1.278279	**1.267957**	0.552655	**0.550482**
Average	1.054879	**1.052763**	1.139112	**1.134453**	0.429623	**0.419668**

The Mean Absolute Error (MEAN), Root Mean Square Error (RMSE) and Standard Deviation (SD) of RPE results are recorded. And the RPE results of ORB SLAM and GCNv2 SLAM are presented in the Table 4.1. Furthermore, the average result of each metric is also provided in the Table 4.1. The lower error results are bolded in the comparison table.

As Figure 4.1a shows, both of ORB SLAM2 and GCNv2 SLAM get the high accuracy estimated trajectories. According to the average value of MEAN, RMSE and, the GCNv2 SLAM outperforms lightly better than the ORB SLAM2. All the average metrics of GCNv2 SLAM are better than the ORB SLAM2, and the value of MEAN, RMSE and SD decrease 0.002, 0.005 and 0.010 respectively (Figure 4.1b).

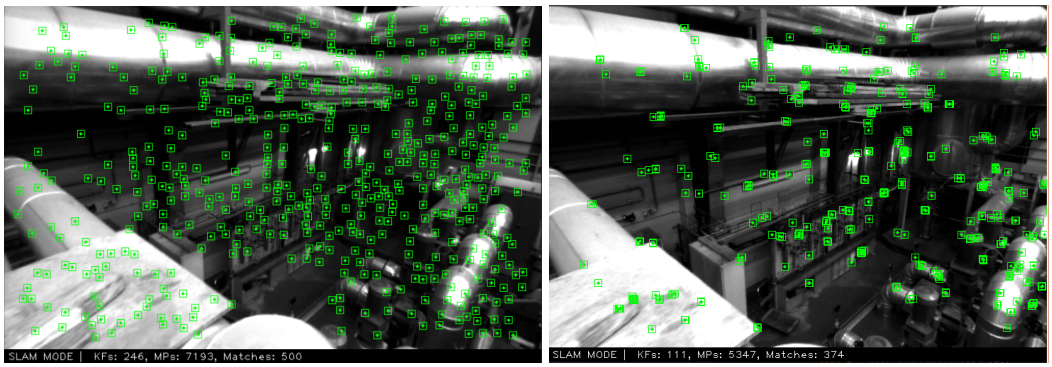
The result presents the performance of GCNv2 SLAM is better than the ORB SLAM in most cases. As Figure 4.1b shows, in the results of *MH01* sequence, the RMSE, Median and STD of GCNv2 are less than the metrics of ORB, which reduces 0.017, 0.119 and 0.02 respectively, which means the accuracy of GCNv2 SLAM is better than ORB SLAM2, in the sequence of *MH01*. And the GCNv2 SLAM has better stability because it has a lower standard deviation and mean.

On the other hands, the stereo camera mode of GCNv2 SLAM has a better performance on the feature points extraction and map establishing as Figure 4.2a, and 4.2b shown. The GCNv2 SLAM obtains more matched points at the same frame in the sequence of *MH01*. The GCNv2 SLAM finds 500 matched points but the ORB SLAM finds 374 matched points in the same frame. The GCNv2 extraction algorithm performs higher efficiency than then ORB extraction algorithm.



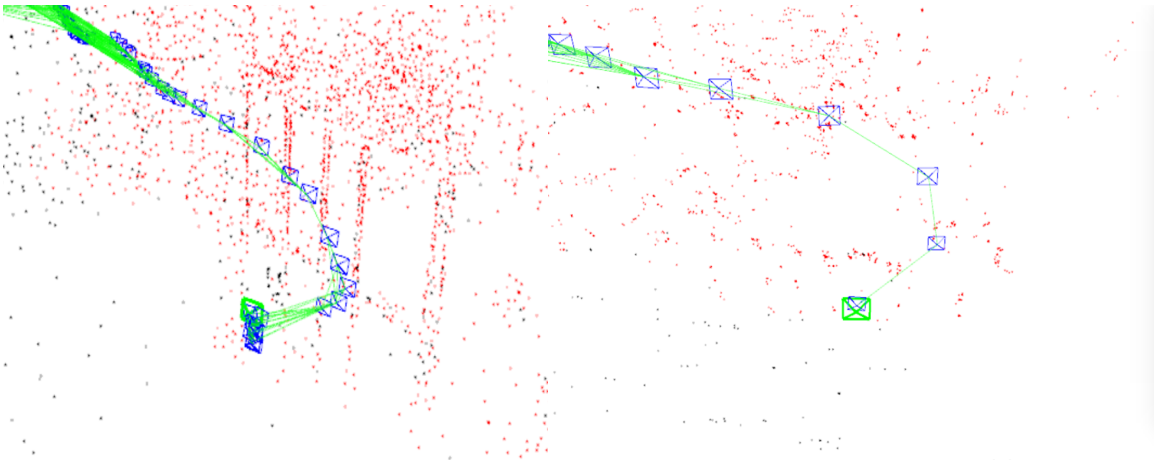
(a) (b)

Figure 4.1: (a) Estimated trajectories of GCNv2 SLAM, ORB SLAM2 and the ground truth. Dataset: *MH01*. (b) The RPE comparation results of ORB SLAM and GCNv2 SLAM. The green raw represents the ORB SLAM and the blue raw represents the GCNv2 SLAM Dataset: *MH01*.



(a) (b)

Figure 4.2: System running diagrams of GCNv2 SLAM and ORB SLAM2. Dataset: *MH01*. (a)The stereo camera mode of GCNv2 SLAM. (b) The stereo camera mode of ORB SLAM



(a) (b)

Figure 4.3: Map diagrams of GCNv2 SLAM and ORB SLAM2. The green line represents the constraints between the keyframes. Dataset: *MH01*. (a)The stereo camera mode of GCNv2 SLAM. (b) The stereo camera mode of ORB SLAM.

Furthermore, the matched points of GCNv2 SLAM are dense and evenly distributed on the frame. However, the matched points of ORB SLAM2 sparsely distributed on the frame with the same timestamps. The quality and quantity of the feature points are essential for the mapping process so that the dense and evenly spread of the points are more beneficial to the map establishment. As Figure 4.3a and Figure 4.3b shown. There are two establishing maps with the same timestamp which are reconstructed by GCNv2 SLAM and ORB SLAM respectively.

The map of GCNv2 SLAM has dense distributed key points in the frame. It is easy to recognize the shape and texture of factory equipment in the map of GCNv2 SLAM. Conversely, the map of ORB SLAM2 is hard to describe the real-world environments since its key points are sparse and messy. Furthermore, the number of green lines in the GCNv2 map is more than in the ORB map. The green line presents the constraints of keyframes (The blue squares in the diagram are keyframes), which also means there are co-visibility points between keyframes. The larger number of greens line presents the stronger constraints between the keyframes, which could reduce the effect of the drift errors. Therefore, the GCNv2 SLAM has a more robust and accurate performance than ORB SLAM2 in the *MH01* sequence.

Even though the general accuracy of SLAM has improved by using the GCNv2 descriptor, particularly in the sequences *MH01*, *V010* and *V201*, however, the performance of ORB SLAM is better than GCNv2 SLAM in the sequence *MH02*. Both the MEAN and RMSE of ORB SLAM2 (0.864955 and 0.944155) are better than the

metrics of GCNv2 SLAM (0.881632 and 0.950140) respectively. The GCNv2 model performs well in the sequences which the movements (Rotation and Translation) are gentle changes. However, the sequence of *MH02* has several actions whose moving states change dramatically in a short period of time. So that, the GCNv2 model is hard to extract enough feature points from the current frame whose movement state is changing dramatically. And it could lead to lost to track and localization. Therefore, the stereo camera mode of GCNv2 SLAM performs worse than ORB SLAM2 in the *MH02*.

In general, the performance of stereo mode of GCNv2 SLAM is better than ORB SLAM2 in most of EuRoC stereo sequences, which the relative pose error decreases about 2.4% on average. The stereo camera mode of GCNv2 SLAM gets a robust and efficient performance on the tracking and mapping parts. Benefits from the high efficiency and low computation of GCNv2 model, more feature points are extracted from the frame which is used in tracking and mapping processes. Comparing the established maps between GCNv2 SLAM and ORB SLAM2, GCNv2 SLAM obtains the better mapping results than ORB SLAM2, because its map points are dense and well organized.

4.3.2 The evaluation of Fisheye + GCNv2 SLAM

The monocular camera mode of Fisheye + GCNv2 SLAM is realized in this project. The purposes of Fisheye + GCNv2 SLAM are to improve the accuracy of tracking and reduce the computation required to compatible the common computation platform. The property of low computation of GCNv2 SLAM has been proved and introduced in the research of [16]. Therefore, in this section, the evaluation of Fisheye + GCNv2 SLAM focuses on the accuracy performance.

The evaluation uses four TUM Visual-Inertial datasets (TUM VI) to test the Fisheye + GCNv2 SLAM. As same as the evaluation of GCNv2 SLAM, each sequence is run for at least five times and the median trajectory is selected to calculate the RPE between the estimated trajectory and ground-truth trajectory. The MEAN, RMSE and SD results are stored to compare the performance of original Fisheye SLAM and Fisheye + GCNv2 SLAM. The results are shown in Table 4.2. The better metrics are bolded in the table.

According to the RPE comparison, the performance of Fisheye SLAM has been significantly improved by adding GCNv2 model. The relative pose errors of Fisheye + GCNv2 SLAM reduces 0.046, 0.0459 on the MEAN and RMSE respectively, on

Table 4.2: COMPARISON OF THE MEAN, RMSE AND SD OF THE RPE OF FISHEYE SLAM AGAINST FISHEYE + GCNv2 SLAM FOR MONOCULAR CAMERA, WITH FULL TRANSFORMATION POSE RELATION (TRANSLATION AND ROTATION) WITH UNITS IN METERS(m)

Metrics	MEAN		RMSE		SD	
Descriptor	ORB	GCNv2	ORB	GCNv2	ORB	GCNv2
Room1	3.468542	**3.432498**	3.575794	**3.528402**	0.869204	**0.817055**
Room2	3.409335	**3.208450**	3.641353	**3.435517**	1.279018	**1.228263**
Room3	**3.182867**	3.373335	**3.340210**	3.565286	**1.013096**	1.154070
Room4	3.037194	**2.852832**	3.139529	**2.983838**	**0.795043**	0.874436
Average	3.274484	**3.216778**	3.424221	**3.378260**	**0.989090**	1.018445

average. And the average SD of Fisheye SLAM is nearly the same as the Fisheye + GCNv2 SLAM. Comparing with the original Fisheye SLAM, the Fisheye + GCNv2 SLAM gets the most apparent accuracy improvement on the sequences of *Room2*. The value of MEAN, RMSE and SD reduces 0.20, 0.250 and 0.05, respectively. As Figure 4.4 shows, the accuracy of Fisheye + GCNv2 SLAM increases 6.2% and 6.8% respectively on the metrics of Mean and RMSE in the *Room2* sequence. Therefore, the Fisheye + GCNv2 SLAM has more robust and stable performance because it has lower sd and means than the original Fisheye SLAM.

Furthermore, the GCNv2 model also obtains a better key points extraction result on the omnidirectional camera. As Figure 4.5a, and Figure 4.5b shown. There are 302 matched points found by the Fisheye + GCNv2 SLAM. However, the original Fisheye SLAM only obtained 145 matched points from the frame, at similar timestamps.

This situation not only shows that the GCNv2 model has higher effectiveness of keypoints extraction but also presents the GCNv2 descriptor has the better property of rotation invariance. Therefore, the Fisheye + GCNv2 SLAM obtains the higher accuracy on the tracking of rotation movements.

There is another advantage of Fisheye + GCNv2 SLAM is that it suffers less influence of illumination. For example, there are several illumination changing situations happened in the sequence of *Room2*. And the Fisheye + GCNv2 SLAM has a better performance than the original Fisheye SLAM in the Room2 sequence.

As Figure 4.6a and Figure 4.6b shows, the Fisheye + GCNv2 SLAM obtains 86 matched key points from the current frame which is in lousy illumination condition.

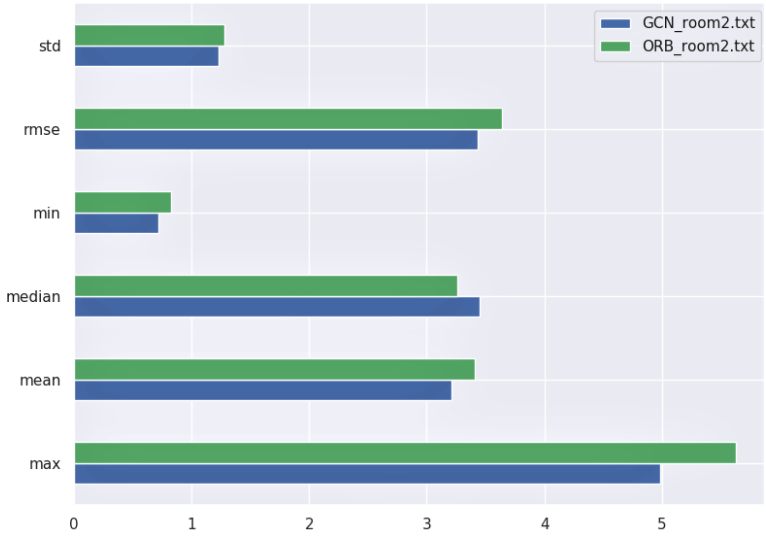
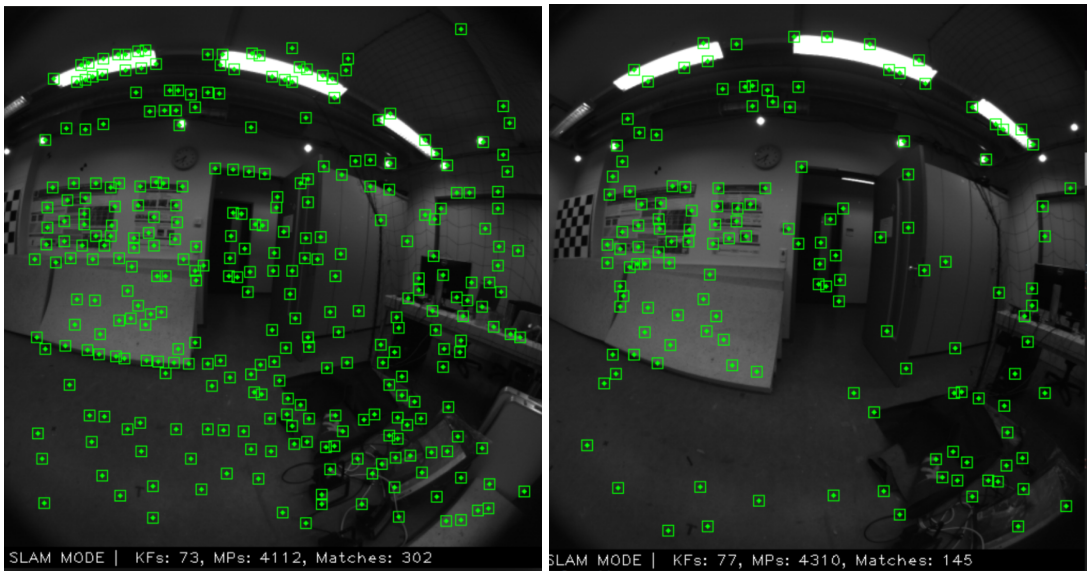
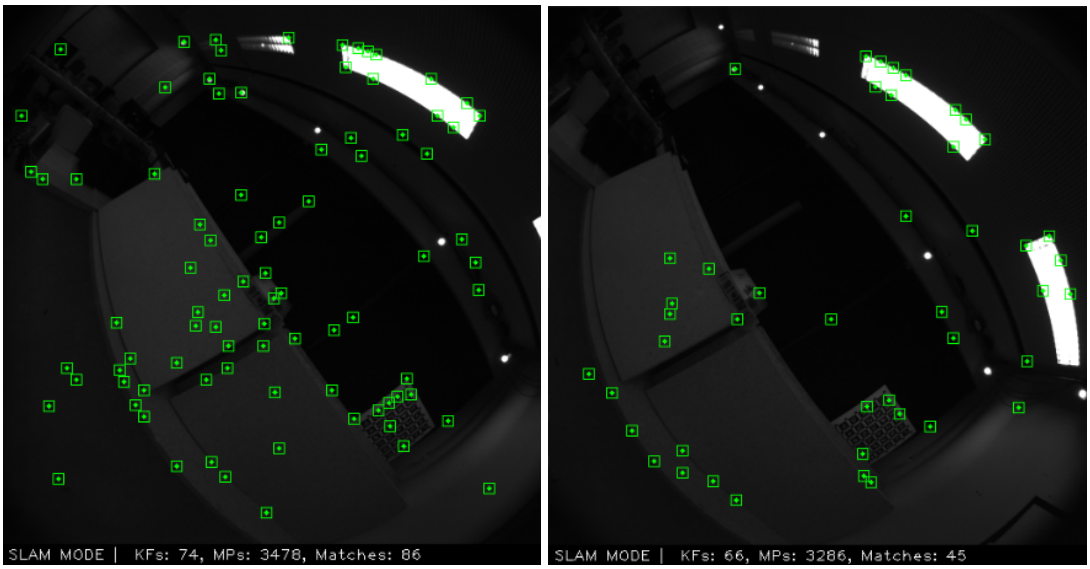


Figure 4.4: The RPE comparation results of ORB SLAM and GCNv2 SLAM. The green bar represents the ORB SLAM and the blue bar represents the GCNv2 SLAM. Dataset: *room2*.



(a) (b)

Figure 4.5: System running diagrams of Fisheye + GCNv2 SLAM and Fisheye SLAM. Dataset: *Room2*. (a)The Fisheye + GCNv2 SLAM. (b) Fisheye SLAM.



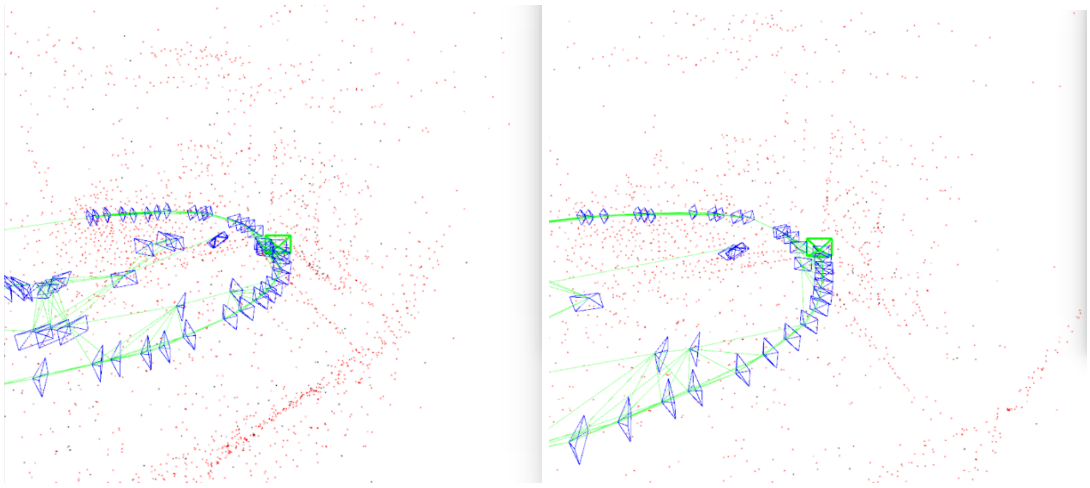
(a) (b)

Figure 4.6: System running diagrams of Fisheye + GCNv2 SLAM and Fisheye SLAM with bad illumination. Dataset: *Room2*. (a)The Fisheye + GCNv2 SLAM. (b) Fisheye SLAM.

However, the original Fisheye SLAM only finds 45 matched points in the same illumination condition because the GCNv2 model has more efficiency in extracting the GCNv2 feature points in the terrible illumination condition so that the Fisheye + GCNv2 SLAM dramatically reduces the influence of illumination in the sequence of *Room 2*.

Furthermore, the established map of Fisheye + GCNv2 SLAM is much more accuracy and precise than the original Fisheye SLAM. As Figuer 4.7a and Figure 4.7b show, since the GCNv2 mode has the better feature extraction efficiency than the ORB algorithm, so that extracted GCNv2 key points are dense and evenly distributed in the map. On the contrary, the ORB key points of original Fisheye SLAM are sparse and messy distributed in the map. Therefore, the recovered map of Fisheye + GCNv2 SLAM has more details and textures of the room environment.

The Fisheye + GCNv2 SLAM improves the property of robust and accuracy, which also can be observed from the Figure 4.7a and Figure 4.7b. In Fisheye + GCNv2 constructed map, the constraints (green lines) between the keyframes are more robust than the map of Fisheye SLAM, which means the system is more robust and accurate. Therefore, Fisheye + GCNv2 system suffers less influence caused by the drift and loss of keyframes.



(a) (b)

Figure 4.7: System running diagrams of Fisheye + GCNv2 SLAM and Fisheye SLAM. The green line represents the constraints between the keyframes. Dataset: *Room3*. (a)The Fisheye + GCNv2 SLAM. (b) The Fisheye SLAM.

The Fisheye + GCNv2 SLAM gets the better performances in most of the TUM VI sequences. Significantly, its trajectory relative pose error has decreased by 6.8% and 5.2% on the sequences of *Room2* and *Room4*, respectively. However, the original Fisheye SLAM performs better accuracy than the Fisheye + GCNv2 SLAM on the *Room3* sequence. The similar situation happens on *Room3* sequence. There are several movements involving violent rotation and translation, and the effectiveness of GCNv2 model is affected by these violent motions. For instance, there is a severe rotation fragment of the camera movement in the *Room3*, and the Fisheye + GCNv2 SLAM can't find enough matches points from the current frame. Then tracking is lost, and the system performs the relocalization process. However, the original Fisheye SLAM performs well on this movement fragment, even though the number of the matched points also reduces obviously.

In general, the performance of Fisheye SLAM has significant improvement by implanting the GCNv2 model. The Fisheye + GCNv2 SLAM obtains the high accuracy on these sequences whose movements are smooth and stable. Compared with the original Fisheye SLAM, the new achieved system has increased the accuracy of at least 5.2% in general omnidirectional camera datasets. Furthermore, benefiting from the high effectiveness of GCNv2 model, the Fisheye + GCNv2 SLAM not only extracts more high quality of key points from frames but also performs well in lousy illumination condition. Therefore, Fisheye + GCNv2 SLAM provides a better SLAM solution

Table 4.3: COMPARISON OF THE MEAN, RMSE AND SD OF THE RPE OF DYNA SLAM AGAINST DYNA + GCNv2 SLAM FOR STEREO CAMERA, WITH FULL TRANSFORMATION POSE RELATION (TRANSLATION AND ROTATION) WITH UNITS IN METERS(m)

Metrics	MEAN		RMSE		SD	
Descriptor	ORB	GCNv2	ORB	GCNv2	ORB	GCNv2
setting_xyz	0.024597	**0.018671**	0.025435	**0.019045**	0.006475	**0.003753**
walking_halfsphere	**0.043035**	bf0.045911	bf0.045273	0.050744	**0.014061**	0.028218
walking_rpy	0.085344	**0.077408**	0.106459	**0.096389**	0.063640	**0.057436**
walking_xyz	0.030722	**0.027183**	0.034548	**0.028312**	0.015802	**0.007917**
Average	0.045924	**0.041793**	0.052928	**0.048622**	0.024994	**0.024331**

for the omnidirectional camera, which requires the lower computation resource and has higher accuracy in tracking and mapping processing.

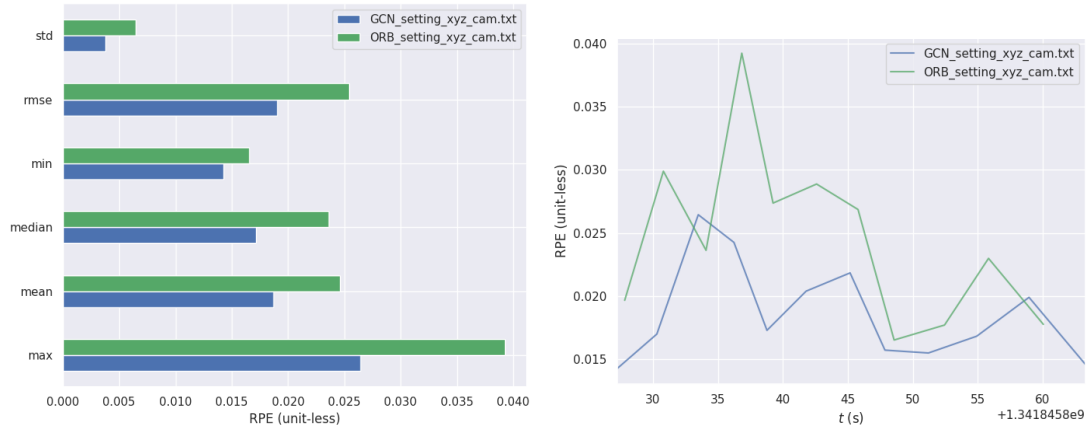
4.3.3 The evaluation of Dyna + GCNv2 SLAM

The RGB-D camera mode of Dynamic + GCNv2 SLAM is achieved in this project. It aims to reduce the required computation resource and improve the quality of tracking, mapping and background inpainting. Similarly, the key point of this section is to evaluate the accuracies of the Dyna + GCNv2 SLAM. The RPE results of Dyna + GCNv2 SLAM will compare with the results of original Dyna SLAM.

Four TUM RGB-D datasets are performed to obtain the established trajectories of Dyna + GCNv2 SLAM. The established trajectories are used to calculate the relative pose error (RPE) among with the provided ground-truth trajectories. The MEAN, RMSE and SD results are stored to evaluate the performance of original Dyna SLAM and Dyna + GCNv2 SLAM, which are shown in Table 4.3. The better metrics are bolded in Table 4.3.

According to the Table 4.3, the performance of Dyna + GCNv2 SLAM gets a greatly improved compared with the original Dyna SLAM, whose MEAN and RMSE of RPE decreases 8.9% and 8.1% respectively in average. Especially, in sequence of *setting_xyz*, the Dyna + GCNv2 SLAM decreases the mean of RPE for 24% (0.0059), and the RMSE of RPE reduces for 25% (0.0063). As Figure 4.8a and Figure 4.8b shows, the Dyna + GCNv2 SLAM obtains the most obvious performance improvement compared with other achieved SLAM algorithms.

According to the results of Figure 4.8a, all the metrics of RPE of Dyna + GCNv2



(a) (b)

Figure 4.8: The bar chart and line chart of RPE of Dyna + GCNv2 SLAM and DynaSLAM. The green represents the Dyna + GCNv2 SLAM and the blue represents the Dyna SLAM. Dataset: *setting_xyz*. (a) bar chart of RPE. (b) Line chart of RPE

SLAM are better than original Dyna SLAM, which presents the accuracy of tracking has improved a lot. And the Figure 4.8b shows that Dyna + GCNv2 SLAM is more stable and effective than Dyna SLAM in the entire trajectory. Almost the RPE of Dyna + GCNv2 SLAM is lower than Dyna SLAM in the whole tracking process, and the average RPE of Dyna + GCNv2 is much lower (about 24%).

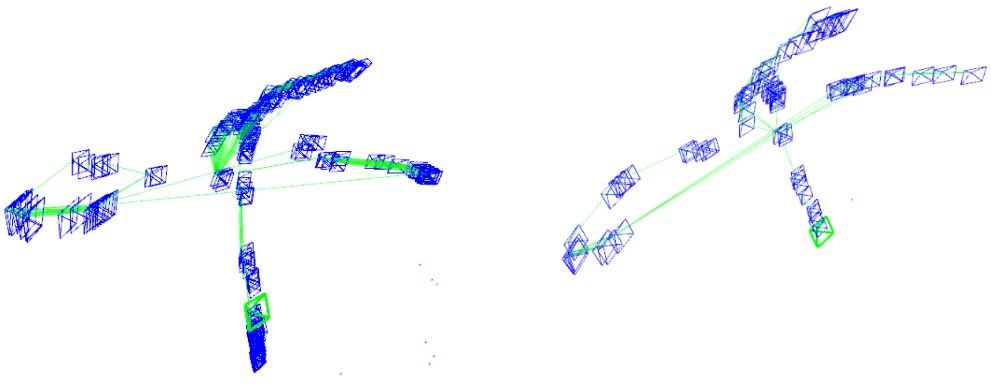
The improvement of performance is benefit from the GCNv2 model, which greatly improve the performance of feature extraction in the most sequences. It is also proved in the comparison of running diagrams.

As Figure 4.9a and Figure 4.9b shows, the Dyna + GCNv2 SLAM obtains 220 matched map points after removing the feature points of dynamic objects (Two setting person). However, the Dyna SLAM only finds 162 matches from the same frame. The matched points of Dyna + GCNv2 SLAM not only has more quantity but also are denser and evenly distributed in the map, which benefits for improving the performance of tracking and mapping processes.

The performance of the Dyna + GCNv2 SLAM outperforms the Dyna SLAM in the robustness and accuracy, which also can be proved from the established map. As shown in Figure 4.10a and Figure 4.10b, the Dyna + GCNv2 SLAM is more robust, and it suffers less influence by keyframes drift. There are more constraints (green lines) between the keyframes (blue squares) in the map of Dyna + GCNv2 SLAM, which are much stronger than the Dyna SLAM. Therefore, the relative positions of keyframes are more accurate and stable in the system of Dyna + GCNv2 SLAM.

(a) (b)

Figure 4.9: System running diagrams of Dyna+ GCNv2 SLAM and Dyna SLAM. Dataset: *setting_xyz*. (a)The Dyna+ GCNv2 SLAM. (b) Dyna SLAM.

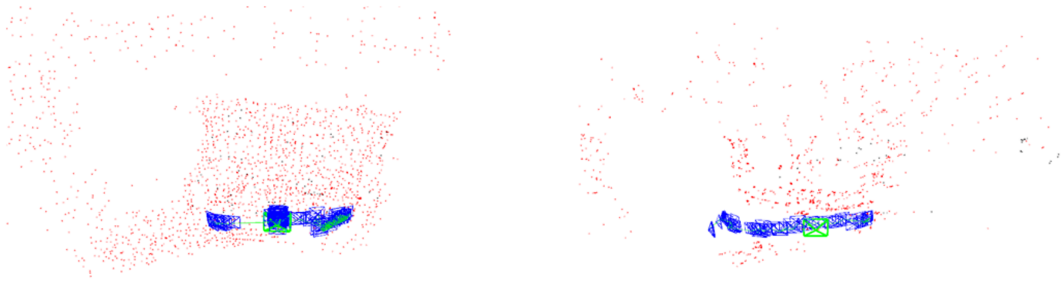


(a) (b)

Figure 4.10: System map diagrams of Dyna + GCNv2 SLAM and Dyna SLAM. The green line represents the constraints between the keyframes. Dataset: *setting_xyz*. (a)The Dyna + GCNv2 SLAM. (b) The Dyna SLAM.

(a) (b)

Figure 4.11: Background inpainting results of Dyna + GCNv2 SLAM and Dyna SLAM. Dataset: *setting_xyz*. (a)The Dyna + GCNv2 SLAM. (b) The Dyna SLAM.



(a) (b)

Figure 4.12: Established map diagrams of Dyna + GCNv2 SLAM and Dyna SLAM. Dataset: *setting_xyz*. (a)The Dyna + GCNv2 SLAM. (b) The Dyna SLAM.

As the accuracy of the Dyna + GCNv2 SLAM greatly improving, the result of background inpainting has significantly improved. As Figure 4.11a and Figure 4.11b shows, the left image (a) is the background inpainting result of Dyna + GCNv2 SLAM at the timestamp of 1341846451.502266. It is evident that the repaired image is more precise than the right-side image (b) which is generated by the original Dyna SLAM at the same timestamp. Because the background inpainting function requires previous images and high accurate relative poses information to fill the background area. Since the accuracy of the SLAM algorithm has been significantly improved, so that the quality of repaired background has a significant improvement.

Furthermore, for the performance of map establishing, Dyna + GCNv2 SLAM outperforms the original Dyna SLAM. As the Figure 4.12a and Figure 4.12b shows, the

Figure 4.13: System running diagrams of Dyna + GCNv2: Lost to track when the violent rotation happened. Dataset: *walking_halfsphere*.

map of Dyna + GCNv2 SLAM has more map points than Dyna SLAM. And the points of Dyna + GCNv2 SLAM are dense and evenly distributed on the map. However, the established map of Dyna SLAM contains fewer map points and the distribution of points are much sparser. Therefore, the established map of Dyna + GCNv2 SLAM has more details and textures then original Dyna SLAM.

Even though the average of accuracy has been improved obviously, but the performance of Dyna + GCNv2 SLAM gets minimal improvement in the sequence of *walking_halfsphere*. The RPE of Dyna + GCNv2 SLAM is nearly the same as the Dyna SLAM. Similar to the situation of Fisheye + GCNv2 SLAM and GCNv2 SLAM, the *walking_halfsphere* sequence contains several movements which have dramatic translation or rotation. As Figure 4.13 shows, the state of the camera is rotating rapidly so that image is indistinct. The Dyna + GCNv2 system fail to obtain enough matched points from the frame, and the relocalization process is performed.

According to the results of the evaluation, the performance of Dyna SLAM has been improved obviously by adding the GCNv2 model. The RPE of Dyna + GCNv2 SLAM has reduced at least 8.1% on average, compared to the original Dyna SLAM. The Dyna + GCNv2 SLAM extracts more high-quality feature points from the frame, which greatly improves the accuracy of tracking and mapping. Furthermore, the quality of the background inpainting function is also improved as tracking accuracy increased.

Chapter 5

Discussion

The results of achieved SLAM algorithms are discussed in this section. Furthermore, the limitations of achievements are also presented and discussed.

5.1 Results

The purposes of this project are to integrate the advantages of three existed SLAM algorithms (Dyna SLAM, Fisheye SLAM and GCNv2 SLAM). And the benefits of GCNv2 SLAM are used as the focuses of integration, which contains the advantages of low computation and high efficiency of feature point extraction. Therefore, three integrated SLAM algorithms are achieved in this project which are RGB-D camera mode of Dyna + GCNv2 SLAM, stereo camera mode of GCNv2 SLAM and the monocular case of Fisheye + GCNv2 SLAM.

After testing the achieved SLAM algorithms, the results of the evaluation show that the performance of three achieved SLAM algorithms have varying degrees of improvement compared with the original SLAM algorithms, especially in the aspects of accuracy, robustness and effectiveness.

The Dyna + GCNv2 SLAM has the most apparent improvement which declines the relative pose error (RPE) about 8.1% on average. And it obtains the best performance improvement in the sequence of *setting_xyz* which reduces at least 25.2% of the RPE. Because GCNv2 model has a higher efficient feature extraction algorithm, the Dyna + GCNv2 SLAM could find more useful feature points from the frame, even though a large number of points has been culled which belong to the dynamic objects. Benefits of the improvement of accuracy, the quality of background inpainting is also improved a lot, which the repaired images have less black gap error and more precise image

clarity. Therefore, the Dyna + GCNv2 SLAM succeed to optimize the original Dyna SLAM by fusing the advantages of GCNv2 SLAM.

The Fisheye + GCNv2 SLAM also presents the more robust and accurate than the original Fisheye SLAM, and it reduces about 6.2% of RPE on average. It also obtains a better tacking performance than original SLAM algorithm in bad illumination condition environment (*Room 2*), which the RPE has reduced at least 8.6%. The results also show that Fisheye + GCNv2 SLAM is more sensitive to the rotation movement since the GCNv2 descriptor is more efficient than the ORB descriptor and the GCNv2 descriptor has better rotation invariance so that the Fisheye + GCNv2 SLAM has achieved the original purpose which is using the advantages of GCNv2 SLAM to improve the performance.

As the results of the evaluation show, the stereo camera mode of GCNv2 also obtains better performance than ORB SLAM2 and it declines the RPE about 2.1% on average. In the running map, the constraints between the keyframes are stronger than ORB SLAM2 so that the property of robustness of the SLAM system has been promoted greatly. Since more map points are extracted and generated, the map points of GCNv2 SLAM is denser and evenly distributed in the constructed map. So that the map of GCNv2 SLAM has more details and textures and the map has higher quality than ORB SLAM2. Therefore, the stereo camera mode of GCNv2 succeeds to realize the optimization.

5.2 Limitation

Even though the achievements have better performance than the original SLAM algorithms, however, there are several limitations found in the evaluation processes.

Firstly, the efficient of GCNv2 model will reduce obviously, when the movements state of the sensor violently changes in a short time. Therefore, the accuracy of three achieved SLAM algorithms will decline obviously, even worse than the original SLAM algorithm if the camera movement is not gentle or smooth. For example, three achievements perform not much improvement in the sequence of *MH02* (EuRoC), *Room3* (TUM VI) and *walking_halfsphere* (TUM), respectively. There are similar situations happened in these sequences, and all of them contains some rotation or translation which the movement state changes violently in a short period. As the Figure 4.13 shows, the GCNv2 model hard to extract enough feature points at this state which the camera was rotating quickly. Therefore, the GCNv2 model more suits to extract the

feature points in the sequences whose movement state is stable and gentle changes.

Secondly, there are two GCNv2 models provided to fit the different resolution of images (320x240 and 640x480). However, some datasets contain the images with different size of resolution, such as the sequences of TUM VI whose image is in the scope of 512x512. The SLAM system resizes the images to satisfy the input formation of GCNv2 model, and the image will be recovered after extracting the feature points. So that the accuracy of results will be affected by twice resize operations. Therefore, the GCNv2 model should be modified to fit more different resolution images in future work. Thirdly, because the Dyna + GCNv2 SLAM uses dynamic detection methods which require massive computation so that it is hard to realize the real-time performance, using a GPU is an excellent solution to deal with this situation. However, because of the time-limited, so that this process has not been done.

As the results show, three achievements have varying degrees of improvement in the aspect of accuracy, robustness, computation efficient and quality of mapping. Therefore, three achieved SLAM algorithms have a better performance.

Chapter 6

Conclusion and future work

This section provides the overall summary of this project. The purpose of the project and achievements' results are summarized in this part. Furthermore, future works are presented in this section.

6.1 Conclusion

The purpose of this project is to integrate the advantages of three outstanding ORB SLAM2 extensions which include Dyna SLAM, GCNv2 SLAM and Fisheye SLAM. The Dyna SLAM uses the Mask RCNN and multi-view geometry method to detect the active items in the frame. The feature points, which belong to the dynamic objects, will be removed and not be used in tracking and mapping processes. In this way, the Dyna SLAM dramatically reduces the influence of active items, which improves the performance of the SLAM system in a dynamic scene. The advantage of Fisheye SLAM is that it broadened the FOV (Field of View) of the original SLAM system by adding the Enhanced Unified Camera Model. The Fisheye SLAM algorithm could extract more feature points from the larger FOV, which improves the accuracy of the tracking system and compatible the omnidirectional cameras. The GCNv2 SLAM proposes a new feature point extraction algorithm which is base on the Graph Convolutional Network (GCN) model. Comparing with the ORB descriptor, the GCNv2 descriptor requires lower computation and has more robustness and efficient. Therefore, the highlights of the GCNv2 SLAM are used as the focuses of advantages integration between Dyna SLAM, Fisheye SLAM and GCNv2 SLAM.

Even though three proposed SLAM algorithms have their own highlights, however, the defects of these SLAM algorithms are also obvious which still have some room for

improvement. Therefore, three achieved SLAM algorithms aim to fuse their advantages and improve the shortcomings. The results of three achieved SLAM algorithms are summarized to prove the improvement of the corresponding defects.

Dyna SLAM provides a satisfying SLAM solution for dynamic scenes. However, as the operation of active items detection and feature points removing are performed, the more computation cost is required. And the accuracy of the SLAM system reduces since many feature points are deducted. To improve the performance original Dyna SLAM, the Dyna + GCNv2 SLAM is achieved to fuse the advantages of GCNv2 descriptor because the GCNv2 model has a better efficiency in feature extraction and requires less computation cost. The results show that the performance of Dyna + GCNv2 SLAM has been significant improvement in the aspects of accuracy and robustness, which the relative pose error has declined 8.1% on average. As the accuracy of tracking has been improved dramatically, the quality of backing inpainting results also has been improved. In the background repaired images, the black gaps are less than the original image, and the newly restored image has a better image clarity. Furthermore, because the Dyna + GCNv2 SLAM obtain a more efficient ability of feature point extraction, the established map of Dyna + GCNv2 SLAM is denser and more precise than the original Dyna SLAM. Therefore, the Dyna + GCNv2 SLAM provides a better SLAM solution for dynamic scenes, with lower computation and better robustness.

Like the Dyna SLAM, the Fisheye SLAM costs mass computation resource to extract feature points from a large FOV frame. Therefore, the Fisheye + GCNv2 SLAM is implemented to deal with this situation, which aims to reduce the computation costs and improves the accuracy and robustness of the SLAM system. According to the evaluation results, the performance of Fisheye + GCNv2 SLAM also gets better performance in the aspects of accuracy and robustness. Compared with the original Fisheye SLAM, the Fisheye + GCNv2 SLAM has reduced the RPE at least 6.2% on average. Then the performance of Fisheye + GCNv2 SLAM gets a prominent promotion in the lousy illumination condition scenes which the ORB algorithm usually fails to extract enough feature points. Furthermore, the Fisheye + GCNv2 SLAM established a denser and more precise map than the original Fisheye SLAM, because the GCNv2 model performs a better efficient and robust feature extraction than ORB algorithm. The results show that Fisheye + GCNv2 SLAM is more robustness and accurate than the original Fisheye SLAM.

The stereo camera mode of GCNv2 SLAM is also achieved in this project. According to the evaluation results, the RPE of the stereo case of GCNv2 SLAM has slightly reduced about 2.1% on average compared with the stereo case of ORB SLAM2. The achieved SLAM algorithm not only requires less computation cost but also constructs a more precise and texture map because it fuses the advantages of GCNv2 descriptor. Even though the accuracy of stereo camera mode of GCNv2 SLAM has a small improvement, but the robustness of achieved SLAM algorithm is stronger than the ORB SLAM, because of the high efficiency of theGCNv2 feature extraction algorithm.

In a word, three achievement SLAM algorithms have been tested and evaluated in real-world datasets. The results present that, all the achieved SLAM algorithms have varying degrees of improvement in the aspects of accuracy, robustness and computation efficient. Therefore, the performance of original SLAM algorithms has been improved dramatically by integrating the advantages of SLAM algorithms.

6.2 Future work

Future work Even though three achieved SLAM algorithms have been improved in different aspects. However, there is still much room to improve performance. There are several future works listed for each achieved SLAM algorithms.

1. The Mask RCNN and multi-view geometry algorithm cost tremendous computation resource in the Dyna SLAM. Even though the Dyna + GCNv2 SLAM has reduced the requirement of computation, but it still hard to realize the real-time tracking and mapping om the CPU device. Therefore, this system is expected to be developed on the GPU, which could provide more powerful computation.

2. The GCNv2 SLAM is a focus of this project. The original GCNv2 SLAM only provides two models for different resolution image (320 x 240 and 640 x 480). The accuracy of the SLAM algorithm will be affected if the size of the frame is not included in these two sizes. For example, the TUM VI dataset provides the omnidirectional camera image in the size of 512 x 512, and the Fisheye + GCNv2 SLAM resizes the image into the required size, which influences the accuracy of feature extraction so that the GCNv2 model is expected to compatible more different resolution images.

3. The Fisheye + GCNv2 SLAM uses EUCM model to compatible the omnidirectional camera. However, there are several different projection models with better precise and efficient such as the Fisheye Camera Model (FCM). The Fisheye SLAM is expected to use other better projection model to improve the accuracy of tracking.

Bibliography

[1] H. Bay, A. Ess, T. Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *Computer Vision and Image Understanding*, 110:346–359, 01 2008.

[2] Berta Bescos, Jose Facil, Javier Civera, and Jose Neira. Dynaslam: Tracking, mapping and inpainting in dynamic scenes. 3:1–1, 07 2018.

[3] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 2016.

[4] Michael Calonder, Vincent Lepetit, and Pascal Fua. High-speed keypoint description and matching using dense signatures. 2 2009.

[5] Michael Calonder, Vincent Lepetit, and Pascal Fua. Brief: Binary robust independent elementary features. 12 2011.

[6] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. pages 337–33712, 06 2018.

[7] Jakob Engel, Jörg Stückler, and Daniel Cremers. Large-scale direct slam with stereo cameras. pages 1935–1942, 09 2015.

[8] Olivier Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern Recognition and Artificial Intelligence - IJPRAI*, 02, 09 1988.

[9] Olivier Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern Recognition and Artificial Intelligence - IJPRAI*, 02, 09 1988.

[10] Dorian Galvez-Lopez and Juan Tardos. Bags of binary words for fast place recognition in image sequences. *Robotics, IEEE Transactions on*, 28:1188–1197, 10 2012.

[11] Michael Grupp. evo: Python package for the evaluation of odometry and slam. `https://github.com/MichaelGrupp/evo`, 2017.

[12] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 06 2018.

[13] Bogdan Khomutenko, Gaetan Garcia, and Philippe Martinet. An enhanced unified camera model. *IEEE Robotics and Automation Letters*, 1:1–1, 12 2015.

[14] Georg Klein and David Murray. Improving the agility of keyframe-based slam. volume 5303, pages 802–815, 10 2008.

[15] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. pages 3607 – 3613, 06 2011.

[16] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81, 02 2009.

[17] Jianfeng Li, Xiaowei Wang, and Shigang Li. Spherical-model-based slam on full-view images for indoor environments. *Applied Sciences*, 8:2268, 11 2018.

[18] Liu, Guo Anfu, Guo Feng, and Yang. Accurate and robust monocular slam with omnidirectional cameras. *Sensors*, 19:4494, 10 2019.

[19] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 11 2004.

[20] Raul Mur-Artal, J. Montiel, and Juan Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31:1147 – 1163, 10 2015.

[21] Raul Mur-Artal and Juan Tardos. Fast relocalisation and loop closing in keyframe-based slam. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 846–853, 06 2014.

[22] Raul Mur-Artal and Juan Tardos. Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras. *IEEE Transactions on Robotics*, PP, 10 2016.

[23] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. volume 3951, 07 2006.

[24] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE transactions on pattern analysis and machine intelligence*, 32:105–19, 01 2010.

[25] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[26] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stueckler, and D. Cremers. The tum vi benchmark for evaluating visual-inertial odometry. In *International Conference on Intelligent Robots and Systems (IROS)*, October 2018. ¡a href="https://arxiv.org/abs/1804.06120" target="$_blank$" $>[arxiv]$.

[27] Shiyu Song, Manmohan Chandraker, and Clark Guest. Parallel, real-time monocular visual odometry. pages 4698–4705, 05 2013.

[28] Hauke Strasdat, Andrew Davison, J. Montiel, and Kurt Konolige. Double window optimisation for constant time visual slam. pages 2352–2359, 11 2011.

[29] Hauke Strasdat, J. Montiel, and Andrew Davison. Scale drift-aware large scale monocular slam. volume 2, 06 2010.

[30] Hauke Strasdat, J. Montiel, and Andrew Davison. Scale drift-aware large scale monocular slam. volume 2, 06 2010.

[31] J. Sturm, W. Burgard, and D. Cremers. Evaluating egomotion and structure-from-motion approaches using the TUM RGB-D benchmark. In *Proc. of the Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RJS International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[32] Yuxiang Sun, Ming Liu, and Max Meng. Improving rgb-d slam in dynamic environments: A motion removal approach. *Robotics and Autonomous Systems*, 89, 11 2016.

[33] Wei Tan, Haomin Liu, Zilong Dong, Guofeng Zhang, and Hujun Bao. Robust monocular slam in dynamic environments. pages 209–218, 10 2013.

[34] Wei Tan, Haomin Liu, Zilong Dong, Guofeng Zhang, and Hujun Bao. Robust monocular slam in dynamic environments. pages 209–218, 10 2013.

[35] Wei Tan, Haomin Liu, Zilong Dong, Guofeng Zhang, and Hujun Bao. Robust monocular slam in dynamic environments. pages 209–218, 10 2013.

[36] Jiexiong Tang, Ludvig Ericson, John Folkesson, and Patric Jensfelt. Gcnv2: Efficient correspondence prediction for real-time slam. *IEEE Robotics and Automation Letters*, PP:1–1, 07 2019.

[37] Jiexiong Tang, John Folkesson, and Patric Jensfelt. Geometric correspondence network for camera motion estimation. *IEEE Robotics and Automation Letters*, PP:1–1, 01 2018.

[38] Red W. and Arabic Portal. Multiple view geometry in computer vision. 03 2018.

[39] Senbo Wangl, Jiguang Yuel, Yanchao Dong, Runjie Shenl, and Xinyu Zhang. Real-time omnidirectional visual slam with semi-dense mapping. pages 695–700, 06 2018.