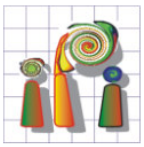


Gottfried Wilhelm Leibniz Universität Hannover
Institute of Photogrammetry and GeoInformation


# A novel stereo SLAM method with dense scene reconstruction

Qi Deng
10032934

Supervisor:
Dr.-Ing. Max Mehltretter

Examiner:
Prof. Dr.-Ing. habil. Christian Heipke
Dr.-Ing. Max Mehltretter

HANNOVER, DECEMBER 2022

## Statement

I declare that this thesis is the result of independent research conducted by me under the guidance of my supervisor. It does not contain the results of any other scientific research that has been published or written by any other individuals or groups, except for those already cited in the thesis. Furthermore, I state that this work in the same or a similar form has not been submitted to an examination authority.

Qi Deng

Signature

Hannover, 30/11/2022

Place, Date

## Abstract

In the field of mobile robotics and autonomous driving, simultaneous localization and mapping (SLAM) techniques are often used. Systems focusing on the dense mapping are an important branch of research as they can handle a variety of challenging robotic tasks such as 3D reconstruction or obstacle avoidance in challenging environments. Building dense maps often involves obtaining scene depth from RGB-D cameras. However, it is susceptible to lighting changes in outdoor scenes and the short range of measurement is not suitable for outdoor scenes. In contrast, binocular cameras are more robust in outdoor environments and can measure long distances based on stereo matching. Traditional stereo matching algorithms currently have some shortcomings, for example, disparity estimation is still challenging for low-texture areas and overlapping areas. Learning-based stereo matching has shown outstanding performance in recent years, both in terms of accuracy and efficiency. Therefore, in this work, a novel deep learning-based stereo SLAM method capable of real-time dense map building in outdoor environments is presented. The method is based on ORB-SLAM2, which flexibly combines end-to-end stereo matching networks in a loosely coupled manner and uses a point cloud fusion strategy based on covisible keyframes and probabilistic filtering to achieve globally consistent, smooth dense map building.

**Keywords** Stereo Visual SLAM, Dense Stereo Matching, CNN, 3D Reconstruction

# Contents

# 1 Introduction

Simultaneous localization and mapping (SLAM) technology is a commonly used technology in the field of mobile robots and autonomous driving, which solves the possibility of robots/mobile devices self-localizing in unknown environments while building a consistent map of the environment. SLAM technology is divided into laser SLAM and visual SLAM according to the sensors used. The former has been developed for a long time, and the latter has attracted the attention of many researchers with the help of the rapid development of computer vision in recent years. In this work, the main research object is visual SLAM.

The classic SLAM system is mainly divided into four modules, which are front-end visual odometer, back-end optimization, mapping, and loop closing. The main tasks of visual odometry are to estimate camera pose between adjacent frames and to construct local maps. Back-end optimization focuses more on optimizing the camera pose or the constructed map to obtain globally consistent trajectories and maps. Mapping is to create a map that meets the task requirements based on trajectory information, such as a sparse map or a dense map. Loop closing will determine whether the robot has returned to the place it has passed before. And pass this information to the backend, the purpose is to reduce the cumulative error and get a globally consistent estimate.

The main focus of different SLAM systems may be different. For example, some require high positioning accuracy, while others require the establishment of dense maps. For the mapping part, some SLAM systems such as the method based on the feature point, can accurately track camera pose and infer scene structure by establishing sparse correspondences between two/multiple views of the scene. However, the 3D map reconstructed by these methods is very sparse. This sparse map can meet the positioning tasks of robots to a certain extent, but it cannot be satisfied for obstacle avoidance and dense reconstruction, so dense 3D maps are needed in more complex robot work. In some dense reconstruction works (Izadi et al., 2011; Newcombe et al., 2011b,a; Keller et al., 2013; Whelan et al., 2015), RGB-D camera is used as a depth measurement sensor. RGB-D camera is an active depth acquisition sensor compared to a monocular or binocular camera, which can obtain more accurate measurements. Therefore, it is often used in dense SLAM or dense reconstruction work. However, the short measurement distance and light sensitivity are its shortcomings, so it is more often used in indoor environments. In this case, the binocular camera stands out for its features. Based on dense stereo matching, binocular cameras can obtain depth information for

most or all pixels by calculating the disparity. Moreover, the binocular camera can perform stereo matching by only one frame when it is stationary, which is a good solution to the initialization problem inside monocular SLAM. At the beginning of a monocular SLAM system, there is only one image, so stereo matching is not possible, so there must be a displacement to perform stereo matching and complete the initialization.

Only being able to obtain scene depth information is not enough of dense reconstruction. On the one hand, there is also an estimation of the camera pose, which is used to transform between image frames, and it is also an important prerequisite for building a globally consistent reconstruction. On the other hand, it is usually necessary to have a well-designed depth fusion strategy. Because the obtained scene-depth is often redundant and imperfect, it is not feasible to reconstruct all pixels of each frame image. Because the camera pose will have errors, and the depth estimation will also have errors, so the reconstructed model will have noise, and there will also be redundancy and overlap. Therefore, a good depth fusion strategy is crucial to achieving smooth, dense reconstruction with global consistency. In this kind of outdoor dense reconstruction tasks, such as common dense point clouds, the huge number of point clouds is the main reason for the huge amount of calculations, and it is for this reason that GPUs are often used to speed up calculations, especially progressive and real-time dense reconstruction work. Overall, exploring a CPU-only SLAM system capable of real-time dense mapping is challenging and valuable.

With the rapid development of deep learning in recent decades, deep learning has also performed very well in the field of dense stereo matching. Researchers have tried to use a learning network to replace the depth estimation function of the SLAM system. Some end-to-end stereo matching networks (Shaked and Wolf, 2017; Chang and Chen, 2018; Xu et al., 2022) show strong robustness, and their prediction results are complete and globally accurate. There are also some works (Wang et al., 2019; Bangunharcana et al., 2021; Shamsafar et al., 2022) that have higher efficiency on the premise of maintaining global accuracy. There is even some work (Mehltretter and Heipke, 2019) that quantifies the uncertainty in the stereo matching process as well, providing reliability for more advanced tasks later on. These learning networks provide a lot of room for capacity expansion to the SLAM system and can be well customized and adapted to the binocular SLAM system.

In my work, I propose a novel lightweight deep learning-based stereo SLAM system capable of densely mapping outdoor scenes in real time. Based on the characteristics of the stable visual SLAM scheme ORB-SLAM2 (Mur-Artal and Tardós, 2017), I combined it with the high-efficiency end-to-end dense stereo matching network AnyNet (Wang et al., 2019) flexibly. And the point clouds are fused between covisible keyframes using a weighted probability filter to achieve a dense, globally consistent outdoor scene reconstruction. The depth obtained from the prediction is also modeled with depth as the mean and uncertainty as the standard deviation (Mehltretter and Heipke, 2021). Gaussian filtering is proposed for simple and effective point cloud fusion. The main contributions

can be summarized as:

- A deep learning-based binocular SLAM method is proposed, which can maintain real-time dense reconstruction of outdoor scenes while relying only on the CPU.
- Designed a flexible combination of a stereo SLAM system and a dense stereo matching network. By loosely combining the SLAM system and the learning network, different stereo matching networks can be easily replaced according to the needs of different tasks.
- Based on the point cloud fusion framework proposed by Wang et al. (2018). The improved CNN network is used to estimate the depth map and the corresponding uncertainty map of the keyframe. Combined with the newly proposed Gaussian filter, the dense point cloud reconstruction with global consistency and smoothness is realized simply and effectively.

# 2 Theoretical Background

In this chapter, I present some of the theoretical backgrounds that are involved in my work. In the first Section 2.1, I introduce the basic concept of ORB-SLAM2, which is the baseline of my work. In Section 2.2 I introduce the concept of co-visibility, which is an important concept in the field of SLAM. Similarly, the use of a co-visibility graph in the ORB-SLAM2 system allows it to perform camera tracking and a local Bundle Adjustment in real time. In Section 2.3, I introduce the three Bundle Adjustment methods used in ORB-SLAM2. In Section 2.4, I introducd the end-to-end dense stereo matching network AnyNet and its main modules. In Section 2.5, I introduce an end-to-end deep learning network called CVA-Net, which quantifies the uncertainty in the stereo matching process and provides reliability for subsequent more advanced tasks. In Section 2.6, I introduce the Gaussian distribution and the Laplace distribution. The former provides the theoretical basis for the Gaussian fusion in the point cloud fusion part of my work. The latter is used in CVA-Net to assume an uncertainty model that allows implicitly learning uncertainty from the deviation between estimated and ground truth differences.

## 2.1 ORB-SLAM2

ORB-SLAM is one of the more well-developed and mature open-source visual SLAM systems available today. It is a feature-based monocular SLAM system that can run in real-time and is suited for a variety of scenarios, including indoor and outdoor scenes, and large and small scenes. It has a high level of resilience, can handle fast-moving images well, and has a significant margin of freedom for loop-closing control, repositioning, and even fully autonomous position initialization. ORB-SLAM2 is a successor to ORB-SLAM that supports monocular, binocular, and RGB-D cameras, as well as map reuse, loopclosing detection, and relocation. It is a very stable and accurate visual SLAM solution. The work of ORB-SLAM2 is mainly done by three threads. Three threads (*Tracking* thread, *Local Mapping* thread, *Loop Closing* thread) are running at the same time. The entire data stream is transformed into ORB feature points in the *Tracking* thread. The camera pose is then estimated based on the relationship between the frames and the current local map. In addition, more keyframes are generated using a relatively liberal keyframe creation approach, which also prevents tracking failure. Local bundle adjustment optimization is used in the *Local Mapping* thread to produce more precise camera poses and map point positions. The computational cost will be increased if there are too many map points and keyframes, so these must be eliminated.

The keyframes are then delivered to the *Loop Closing* thread following a tight keyframe filtering procedure. Finding a loop-closing that eliminates the camera pose and mapping error brought on by camera motion is the aim of the *Loop Closing* thread, which aims to ensure that the entire system efficiently prevents accumulated errors and may be promptly recovered after being lost. Three threads allow for effective tracking and mapping, guaranteeing that the global trajectory and map are consistent.

## 2.2 Covisibility Graph

The co-visibility graph is an important tactic in the ORB-SLAM family. Real-time operations are made possible by the use of a co-visibility graph, which enables optimization issues in tracking and local mapping to concentrate only on a locally covisible area (Mur-Artal et al., 2015). The co-visibility graph is simply described it is a system maintained covisible state and made up of the covisible state of every keyframe in the entire system.

The current keyframe is presented in green in Figure 2.1 on the left, and all keyframes created during the tracking process are shown in blue. All of the map points that the current keyframe and its covisible keyframes can see are represented by the red 3D spatial map points, which are referred to as local map points. Additionally, the co-visibility information between keyframes is expressed as an undirected weighted graph as seen in Figure 2.1 on the right. If two keyframes observe some of the same map points (at least 15), then these two keyframes constitute two nodes, and the weight of the edge between them is the number of points observed together (Mur-Artal et al., 2015). In the tracking thread of ORB-SLAM, the camera pose is obtained by tracking the reference keyframe (matching the nearest keyframe) or tracking the motion model (matching the previous frame). In this process, the technology of Bags of Words (BOW) (Gálvez-López and Tardos, 2012) is used to convert the descriptor of the ORB feature point into a BOW vector, and then search between two frames to find the closest vector. This approach speeds up the confirmation of conjugate feature points. If the number of conjugate points is too small (below 15 pairs), it is considered that the common view between two frames is too small to provide a robust precondition for obtaining the camera pose.

The current local covisible area consists of the current keyframe, its covisible keyframes, and the map points visible to these keyframes. This is shown in Figure 2.2. The current keyframe $kF$ and its covisible keyframes $kF_{i-1}, kF_{i-2}$ together with the map points (red points) that can be observed by these three keyframes form the current local covisible area.

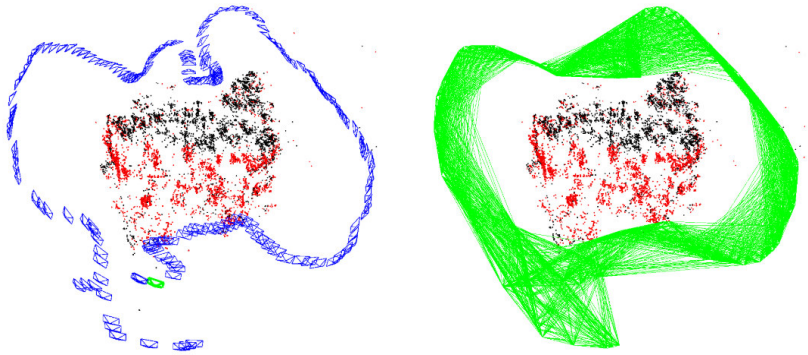


Figure 2.1: The co-visibility graph (Mur-Artal et al., 2015). Left figure: all keyframes (blue and green), current keyframe (green), map points (red and black), local map points (red); Right figure: the co-visibility graph. Connecting each keyframe with other keyframes that have a covisible relationship with the current keyframe to form a co-visibility.

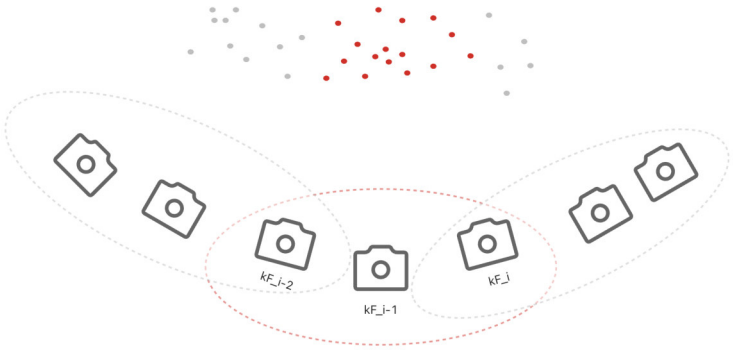



Figure 2.2: The local covisivility area. The red points are the map points that can be seen in the current covisible area. The current keyframe $kF_i$ and its covisible keyframes $kF_{i-1}, kF_{i-2}$ can see these map points or some of them (at least 15).

## 2.3 Bundle Adjustment

Bundle adjustment (BA) is the problem of refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameter (camera pose and/or calibration) estimates (Triggs et al., 1999). BA is frequently used in 3D reconstruction and SLAM systems. For example, in ORB-SLAM2, performing motion-only BA to optimize the camera pose in the *Tracking* thread, performing local BA to optimize all the points and the camera pose in a local map in the *Local Mapping* thread, and performing full BA to optimize all keyframes and points in the *Loop Closing* thread.

When estimating the camera pose, the estimated camera pose is often solved by constructing a Perspective-n-Point (PnP) problem, or PnP for short, whose goal is to establish the position $t \in \mathbb{R}^3$ and orientation $\mathbf{R} \in SO(3)$ of the camera in light of its intrinsic properties $K$ and a collection of $n$ correspondences between 3D points $\mathbf{X}^i \in \mathbb{R}^3$ and their projections $\mathbf{x}^i_{(\cdot)}$ (Lepetit et al., 2009).

**Motion-only BA**, whose main job is to project the 3D points onto the reference keyframe through the transformation matrix and then optimize the error between the projected points and the corresponding feature points, which do not overlap because of the noise. During the iterative optimization process, the position of the 3D points is kept constant with the internal parameters of the camera, and the camera pose is optimized iteratively to obtain the minimum error. This process is illustrated by the following Equation 2.1:

$$\{\mathbf{R}, \mathbf{t}\} = \underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}} \sum_{i \in \mathcal{X}} \rho \left( \left\| \mathbf{x}^i_{(\cdot)} - \pi_{(\cdot)} \left( \mathbf{R}\mathbf{X}^i + \mathbf{t} \right) \right\|^2_{\Sigma} \right) \tag{2.1}$$

where $R, t$ represents the rotation matrix and translation vector, respectively. They are used to represent the camera pose to be estimated. $\rho$ is the robust Huber cost function, its goal is to transform the squared measure of the error term into a function that does not grow so fast while maintaining its own smoothness. $\sum$ is the covariance matrix associated with the scale of the keypoint. What it means is that when quantifying the reprojection error, it is necessary to keep the feature points at the same scale. This is because, when extracting ORB feature points, an image pyramid is constructed, so different feature points may be located at different scales. Projection function $\pi_{(\cdot)}$, which has two cases. It is $\pi_{(m)}$ when a monocular camera is used and $\pi_{(s)}$ when a binocular camera is used. They are defined as Equation 2.2:

$$\pi_{\mathrm{m}} \left( \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \right) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \end{bmatrix}, \quad \pi_{\mathrm{s}} \left( \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \right) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \\ f_x \frac{X-b}{Z} + c_x \end{bmatrix} \tag{2.2}$$

where $f_x, f_y$ is the product of the focal length $f$ and the scaling factor $\alpha, \beta$ of the $x, y$ axes between the physical imaging coordinate system and the pixel coordinate system, respectively. $c_x, c_y$ represents how many pixels the origin has been translated on the pixel coordinate system. These

parameters are also intrinsic to the camera, and they can be obtained through the calibration of the camera. When using binocular cameras, $b$ is the baseline, which is the distance between the optical axes of the two cameras.

**Local BA** optimizes a set of covisible keyframes $K_L$ and all points seen in those keyframes $P_L$. All other keyframes $K_F$, not in $K_L$, observing points in $P_L$ contribute to the cost function but remain fixed in the optimization. Defining $X_k$ as the set of matches between points in $P_L$ and key points in a keyframe $k$, the optimization problem is the following (Mur-Artal and Tardós, 2017):

$$\left\{\mathbf{X}^i, \mathbf{R}_l, \mathbf{t}_l \mid i \in \mathcal{P}_L, l \in \mathcal{K}_L\right\} = \underset{\mathbf{X}^i, \mathbf{R}_l, \mathbf{t}_l}{\operatorname{argmin}} \sum_{k \in \mathcal{K}_L \cup \mathcal{K}_F} \sum_{j \in \mathcal{X}_k} \rho(E(k,j)) \tag{2.3}$$

$$E(k,j) = \left\| \mathbf{x}^j_{(\cdot)} - \pi_{(\cdot)} \left( \mathbf{R}_k \mathbf{X}^j + \mathbf{t}_k \right) \right\|^2_{\Sigma} \tag{2.4}$$

**Full BA** is to consider all keyframes and all map points that these keyframes can see, and optimize them.

## 2.4 Any-Net

The precision of the computation output and overall efficiency must often be traded off in current deep learning-based stereo depth estimation efforts because network parameters are numerous and GPU-dependent. In light of these flaws, AnyNet shows competitive performance in terms of efficiency and accuracy, even when running only on the CPU. The network diagram is illustrated in Figure 2.3.

The entire process of estimating disparity in the network can be broken down into four stages as time goes on. The disparity maps obtained at various stages become sharper over time. The input stereo image is delivered to a U-Net (U-Net feature extractor) before the first step in order to extract feature maps with various resolutions (1/16, 1/8, 1/4). because feature maps with various resolutions can acquire information at various scales (Wang et al., 2019).

To create a low-resolution disparity map, only the lowest-resolution (1/16) feature map is fed into the Disp-Net to generate a low-resolution disparity map. The entire stage 1 computation only takes a few milliseconds because of the low resolution of inputs (Wang et al., 2019). It moves into stage 2 as time goes on and continues to use U-Net to get feature maps with a slightly higher resolution (1/8). The disparity map obtained in stage 1 is just corrected in stage 2 as opposed to being completely recalculated using the newly acquired feature map with greater resolution (1/8) through Disp-Net. The disparity map is upscaled during correction to match the resolution in stage 2. Calculating a residual map that includes minor adjustments that indicate how much each pixel

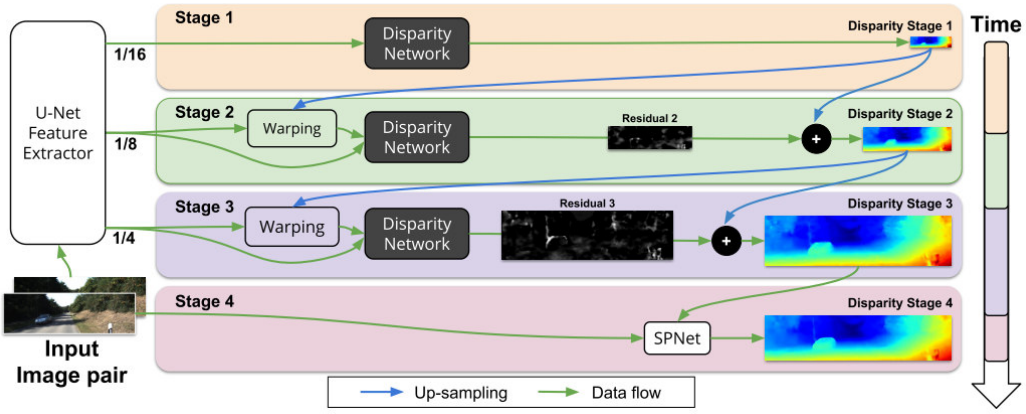



Figure 2.3: Network diagram of AnyNet (Wang et al., 2019). The left and right images are fed into U-Net to extract feature maps with resolutions of 1/4, 1/8, and 1/16 of the original image. The 1/16 feature map is passed through Disp-Net in stage 1 to obtain a low-resolution disparity map. In stage 2, the 1/8 feature map is combined with the disparity map of the previous stage to obtain a residual map through Disp-Net. This residual map is used to guide the adjustment of the disparity map of the previous stage and output the disparity map of the stage 2. The stage 3 process is similar to stage 2. In stage 4, the disparity map is further refined through SP-Net by combining the disparity map of the previous stage and the original image.

should increase or decrease (Wang et al., 2019). In stage 4, by applying a local filter whose weights are forecast by applying a small CNN on the left input picture, the SP-Net sharpens the disparity map (Liu et al., 2017).

## 2.5 CVA-Net

Although the deep learning method estimated disparity meets the task requirements in terms of global accuracy, there are still errors. Then, if the probability of the accurately estimated result can be determined, a more accurate disparity can be artificially screened. For example, in some confidence estimation work, pixels are estimated one by one with their confidence level, which represents the probability that the pixel is assigned the correct disparity value. That is, the higher the probability, the higher the probability that it has been assigned the correct disparity value. Then local outliers, such as pixels with very low confidence, can be removed, thus adjusting the ratio of density to reliability. Additionally, information about the uncertainty of individual disparity estimates is used to support the depth reconstruction process itself, such as by adjusting the impact of disparity assignments on their local neighborhoods (Mehltretter and Heipke, 2021).

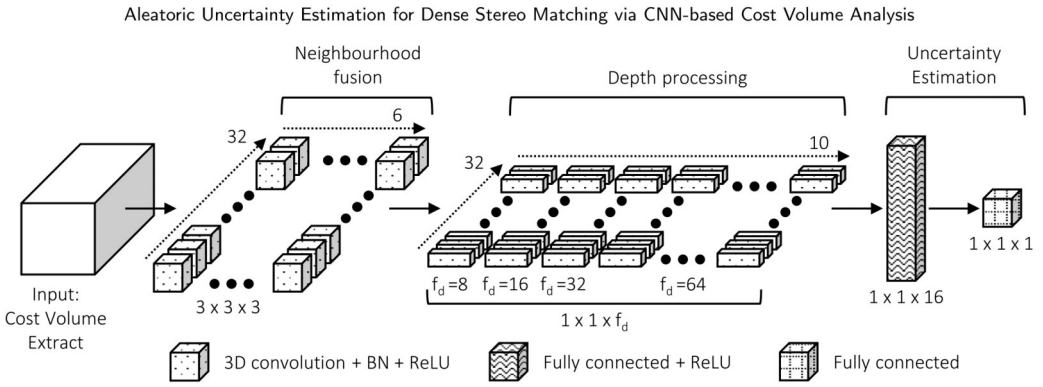



Figure 2.4: CVA-Net (Cost Volume Analysis Network). The network, which consists of three primary components, initially combines cost volume extractions into a single cost curve before further processing it along the disparity axis. These two components are developed via 3D convolutions, with batch normalization (BN) and a ReLU nonlinearity coming after each. In the first portion, the convolution kernel's size is constant throughout all layers, whereas in the second part, the filter depth d is variable. At the end of the network, the fully connected layer estimates the arbitrary uncertainty pixel by pixel. (Mehltretter and Heipke, 2021).

Taking into account the reliability of disparity estimation. Depth reconstruction methods may fail to identify the correct correspondence for all pixels, especially in difficult conditions (insufficient texture, fast camera movement, poor image quality, etc.). In other words, the disparity estimation is inaccurate and there is uncertainty, so measuring this uncertainty is the task of uncertainty estimation. CVA-Net was first proposed for confidence estimation (Mehltretter and Heipke, 2019). On this basis, the network was adapted and used for probabilistic modeling (Mehltretter and Heipke, 2021), to be able to learn uncertainty implicitly from the deviation between the estimated and ground truth discrepancy in the absence of uncertainty in the real reality.

CVA-Net is divided into three sections: neighborhood fusion, depth processing, and classification. Figure 2.4 depicts the detailed network framework. The network accepts cost volume extracts of size $N * N * D$ as input. $N$ denotes the size of the perceptive field, and $D$ denotes the maximum disparity taken into account. In the first part, the information contained in cost volume extracts is merged into a single cost curve via neighborhood fusion. The basic idea of this process is similar to that of most region-based matching methods: including neighborhood information improves robustness. The merged cost curve is further processed in the following depth processing section to derive high-level features that characterize the curve. In the third section, the final uncertainty estimation is performed by using the fully connected layer to estimate the uncertainty.

## 2.6 Probability Distributions

The **Gaussian distribution** $x \sim \mathcal{L}(\mu, \sigma^2)$, also known as the normal distribution, is a very common continuous probability distribution, with $\mu$ is the location parameter and representing the mean value of expectation. $\sigma$ is the scale parameter and representing the standard deviation. The probability density function is defined as

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.5}$$

The Gaussian distribution is linear, so after adding two Gaussian distributions, the Gaussian distribution is still satisfied. After the two Gaussian distributions are multiplied, the Gaussian distribution is also satisfied, and the mean and variance after the multiplication satisfy the following relationship.

$$\mu = \frac{\mu_2\sigma_1^2 + \mu_1\sigma_2^2}{\sigma_1^2 + \sigma_2^2}, \quad \sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \tag{2.6}$$

The related mathematical derivation is not discussed here.

The **Laplacian distribution** $x \sim \mathcal{L}(\mu, b)$ is a continuous distribution with location parameter $\mu$ and scale parameter $b$. The mean of expectation of the Laplacian distribution is $\mu$, and the variance is $2b^2$. The absolute difference from the mean of expectation $\mu$ is used to express the Laplacian density. The probability density function is defined as

$$p(x \mid \mu, b) = \frac{1}{2b} \exp(-\frac{|x-\mu|}{b}) \tag{2.7}$$

Considering the relationship between its standard deviation $\sigma$ and the variance $2b^2$, and replacing $b$ in the above formula with $\sqrt{2}b = \sigma$, the following formula can be obtained.

$$p(x \mid \mu, b) = \frac{1}{\sqrt{2}\sigma} \exp(-\frac{\sqrt{2}|x-\mu|}{\sigma}) \tag{2.8}$$

In this way, the probability density function, i.e. the likelihood can be described directly by the mean $\mu$ and the standard deviation $\sigma$.

# 3 Related Work

In this section, I will discuss related research in three main areas: several classical visual SLAM methods, deep learning-based dense stereo matching, and SLAM methods combined with deep learning.

## 3.1 Visual SLAM

Visual SLAM has long been a focus of researchers due to its price advantage and potential to acquire richer information for more complex tasks. To classify SLAM methods in terms of sensor categories, there are monocular camera-based, binocular camera-based, and RGB-D camera-based. When classified based on methodology, there are feature point-based and direct methods. Kinect Fusion (Izadi et al., 2011; Newcombe et al., 2011a) realize indoor-scale 3D reconstruction with real-time, dense volume reconstruction using only one RGB-D camera, and GPU. And a basic pipeline of reconstruction is proposed. 1. depth measurement. 2. update reconstruction. 3. surface prediction. 4. pose prediction. Keller et al. (2013) made several improvements, such as using a point-based representation that works directly with inputs obtained from distance/depth sensors without the overhead of converting between representations. While the previous methods are based on RGB-D cameras, LSD-SLAM (Engel et al., 2014) is a direct method SLAM based on monocular cameras, which uses a depth filter to complete the estimation of the depth of the pixel points with large gradients in the image, and calculates the uncertainty of that depth assumption, optimizes it by new observations, and incorporates it into the global map when the depth values converge. A semi-dense map building on the CPU in real-time is achieved. Mur-Artal et al. (2015) proposed to use three threads to divide the functions of each module of the SLAM system. The use of ORB feature points and the covisible graph strategy allow a good balance of accuracy and efficiency both in the process of estimating camera poses and in the process of optimization. ORB-SLAM2 (Mur-Artal and Tardós, 2017), as an upgraded version of ORB-SLAM, not only supports a monocular camera, but also a binocular camera and depth camera as input. It is because the ORB feature points are extracted from all the input images and then used as data streams in the later work. ORB-SLAM is lightweight, long-term, and capable of constructing sparse maps with global consistency. The three BA methods enable the estimated camera poses with high accuracy. Nevertheless, in some task scenarios such as robot obstacle avoidance or scene modeling, sparse maps are difficult to satisfy the task. Thus, lightweight SLAM with dense reconstruction is in

demand.

## 3.2 Deep Learning-based Dense Stereo Matching

In recent years deep learning has been introduced to depth estimation, and many works (Eigen et al., 2014; Zbontar and LeCun, 2015; Liu et al., 2015; Eigen and Fergus, 2015; Li et al., 2015; Shaked and Wolf, 2017; Chang and Chen, 2018) have demonstrated more robust and accurate estimation compared to these traditional methods. Cross-based cost aggregation and semi-global matching are used to refine the matching cost (Zbontar and LeCun, 2015), followed by a left-right consistency check to eliminate errors in the occluded regions. On the super-pixel level, Liu et al. (2015) introduced a deep learning strategy that learns the unary and pairwise potentials of continuous CRF in a unified deep CNN framework. A new highway network (Shaked and Wolf, 2017) was designed, based on multilayer weighted residual shortcuts, trained with a hybrid loss that supports multilevel comparison of picture patches, and has been developed for determining the matching cost at each potential disparity. A second deep convolutional neural network is then used in a new post-processing stage. In PSM-Net (Chang and Chen, 2018), spatial pyramid pooling (SPP) was set up to collect hierarchical contextual information. To combine feature information along with disparity and spatial dimensions, and to gain additional context information, a stacked hourglass architecture is utilized. However, all these algorithms mentioned above need to choose terms of accuracy and computation time. In AnyNet (Wang et al., 2019), the disparity is first estimated at the lowest resolution (1/16 of the original resolution) at the full disparity range, this stage is usually completed in a few milliseconds. Then, starting from a low resolution, the resolution of the disparity map is successively increased by upsampling and subsequently correcting errors that are apparent at higher resolutions. To ensure the reliability of disparity estimation, Mehltretter and Heipke (2019) further consider the confidence of stereo matching, which is also called confidence estimation. Using CNN, the confidence in the disparity estimation process is learned end-to-end from the cost volume. On this basis, different uncertainty models (Mehltretter and Heipke, 2021) are used to quantify uncertainty, which provides a more robust premise for more advanced tasks after disparity estimation, such as map building. Disparity estimation in stereo matching tasks, to recover depth information, can be very directly helpful for SLAM tasks, such as map building. Therefore, the cooperation between stereo matching networks and SLAM systems can be naturally facilitated.

## 3.3 Deep Learning-based SLAM

Deep learning has been widely used in SLAM systems in recent years. In the work of (Tateno et al., 2017), CNN-SLAM looks into the usage of deep neural network-predicted depth maps for precise and dense monocular reconstruction. It is suggested to combine the direct monocular SLAM depth

measurements with the dense depth maps predicted by CNN. A framework for depth fusion is presented in this work. For each new keyframe generated, its depth map is estimated by the CNN module and its uncertainty map are also determined, where the uncertainty is represented by the squared distance of the depth difference between the current keyframe and the nearest neighboring keyframes. While using the uncertainty map, the newly generated depth map is fused with the already generated depth map. Not only for end-to-end estimation of depth but CNN networks are also used to extract image features. However, due to scale inconsistency, CNN-SLAM can only be trained on datasets with the same camera intrinsic parameters, which limits its generalization usage. To expand to training on mixed datasets with varied intrinsic parameters. Wang et al. (2018) developed a deep prediction network with adaptive loss. At the same time, loosely combines depth prediction with monocular SLAM and frame-by-frame point cloud fusion based on the weighted probability filter. In his work, the confidence of disparity estimation is used to guide depth fusion. During this process, the confidence is also updated, and the confidence level after fusion is used to determine whether the point cloud is in a stable state, and if it is stable, it is retained, and if it is unstable, subsequent fusion or selective rejection is performed. However, the determination of the initial confidence level is based on experience. The above work uses the combination of the direct method SLAM and CNN. On the one hand, it can construct a semi-dense or dense map, which is easier to integrate with the learned depth. On the other hand, its calculation cost is also compared with feature point-based SLAM methods are lower. However, based on the assumption of invariant grayscale, the direct method SLAM is sensitive to light changes, making its use in outdoor environments challenging. SLAM based on the feature point method has more advantages in this regard. Because the depth information generated by the feature point-based SLAM is typically quite sparse, cooperation between it and predicted depth by CNN is more difficult than that of the direct method SLAM. To create dense monocular reconstructions, Ji et al. (2018); Ye et al. (2020) fully utilized the sparse depth samples and the depth inferred by the CNN and suggested a depth fusion architecture. Using sparse feature map points and dense maps estimated by CNN for fusion, and guided by color keyframes, a depth reconstruction model is developed to fuse heterogeneous depth data and simultaneously reconstruct high-quality dense depth maps. But in his work, the confidence map is obtained by considering both the depth of the sparse feature points and the depth obtained from the CNN estimation, in three cases to model the confidence. No further consideration is given to the sources of uncertainty generated during disparity estimation. Confidence and uncertainty estimation during disparity estimation is shown to improve the robustness of the system and provide the reliability of the predicted depth for subsequent 3D reconstructions (Mehltretter and Heipke, 2019, 2021). In the work of Ding et al. (2020), the uncertainty of the noisy data was estimated by modeling this uncertainty in a new loss function. Finally, the predicted depth maps and the corresponding depth uncertainties are combined into a monocular reconstruction system. The implementation allows for smoother and denser model reconstruction on a variety of scenes.

In this literature, most of the works are based on monocular depth estimation and monocular SLAM systems. While monocular cameras have an advantage in cost and size. However, a specialized initialization process is required in monocular SLAM due to the scale uncertainty. Moreover, the monocular camera must have a translational motion to be able to perform depth estimation, which is not possible in the case of pure rotation. In contrast, the binocular camera is based on stereo matching and can initialize and estimate depth even when it is stationary, and it can measure a much longer distance and can be used outdoors. At the same time, since depth estimation and dense mapping require a lot of processing resources, many of the above tasks use GPU resources. Therefore, in my work, I propose a novel lightweight stereo SLAM method based on deep learning, which is achieved by flexibly combining an end-to-end stereo depth estimation network with the ORB-SLAM2 system.

# 4 Methodology

In this section, I present the methodology in my work, including a statement of the my proposed method and the overall framework 4.1, the three main modules involved in the system: the pose estimation module 4.2, the depth estimation module 4.3, and the mapping module 4.4.

## 4.1 Proposed Method

The new SLAM method I propose is based on the framework of ORB-SLAM2. Adjustments and additions have been made on this basis. Due to the stability and accuracy of ORB-SLAM2, it is widely used as a framework by researchers to propose new work. The framework of my proposed method is shown in Figure 4.1.

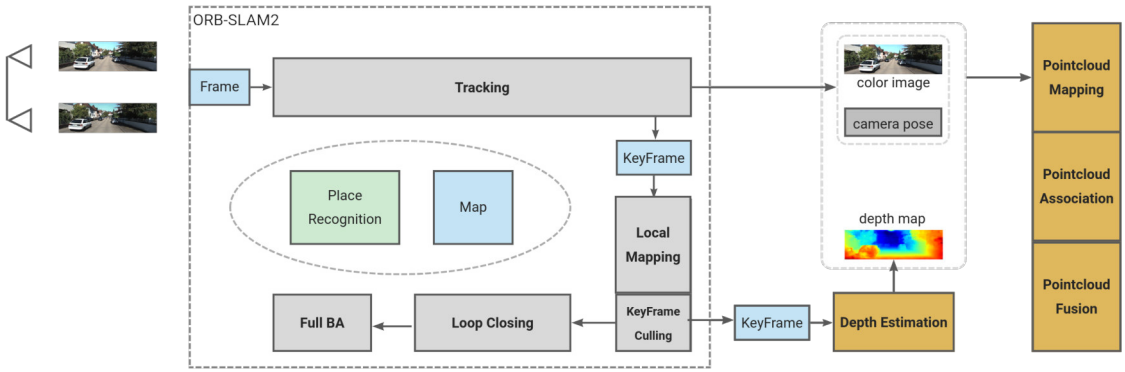



Figure 4.1: The framework of the proposed method consists of three main parts: the pose estimation module, the depth estimation module, and the dense mapping module. Estimating the pose of the camera in the tracking thread and generating keyframes according to the generation strategy. In the local mapping thread, the redundant keyframes are filtered and then passed to the loop closing thread and the depth estimation thread. Using the learning network, the depth map of keyframes is estimated in the depth estimation thread. Then in the dense mapping thread, the poses of the keyframes, the input RGB images, and the depth maps of the keyframes are combined to incrementally build a dense point cloud.

The system is mainly composed of three parts, namely the pose estimation module, the depth estimation module, and the dense mapping module. A more detailed description of the pose estimation module is the tracking thread and local mapping thread in ORB-SLAM2. ORB feature points are extracted first, and then the current pose of the camera is estimated according to the matching between frames, and keyframes are generated according to the generation strategy. Keyframes are passed into the local mapping thread and filtered to reduce redundancy. The reason why the keyframes generated by the tracking thread are redundant is to ensure the robustness of the tracking so that keyframes will be generated as frequently as possible. But redundant keyframes are a pressure on subsequent depth estimation and mapping tasks. The number of filtered keyframes is much smaller and evenly distributed among the input frames. It is then passed to the depth estimation thread of the depth estimation module. The original input stereo image is located according to the keyframe, and the corresponding depth map is estimated by using the learning network. Next, the depth map is passed to the dense mapping module. After the first keyframe is passed in, all pixels with reasonable disparity values, combined with the original RGB image and the keyframe pose, are mapped into a 3D point cloud. Afterward, according to the point cloud fusion strategy, the point cloud is incrementally updated to construct a globally consistent and smooth point cloud.

## 4.2 Pose Estimation Modul

ORB-SLAM2 can accurately estimate the pose of the camera, so its positioning in my work is a module of pose estimation. The process of pose estimation and other details are worth explaining.

### 4.2.1 Camera Tracking

It will be initialized when the system is just started. Since stereo input is selected, stereo initialization is performed here, which refers to the construction of the frame and the initial map. In the whole system, the flow of data is realized through the frame object. Each pair of stereo images is converted into a frame. In the process of converting to frame, ORB feature points are simultaneously extracted on the left and right images, and stereo matching is performed to obtain the number of matched point pairs. This information and the index corresponding to the input RGB original images will be stored in the object of this frame.

The premise information of the initialization is the object of the frame, and the system will judge whether the number of successfully matched feature points contained in the current frame is greater than the threshold. If the conditions are met, continue to try to initialize, which includes a series of steps. The main action is to set the pose of the current frame as the origin, convert the current frame to the initial keyframe and insert it into the map, and map the feature points of the successful stereo matching in the current frame through triangulation to obtain depth to the

map, etc. This is also the advantage of binoculars over monoculars. It can directly obtain depth from a single frame and does not need to be initialized based on the theory of small baselines like monocular cameras. That is to say, since the monocular camera cannot obtain depth through a pair of images, when the monocular camera is performing the translational motion, assuming that the motion gap is small, it can be assumed that this small gap is similar to the baseline in stereo matching. In this way, stereo matching can be performed based on this assumption to obtain depth. Subsequent pose estimation and map expansion are based on the results of stereo initialization.

When the system is initialized, the second frame is ushered in at the same time, and then the inter-frame matching is performed to estimate the pose of the camera. There are two main ways of estimation: one is to estimate the pose by tracking the reference keyframe, and the other is to use the motion model, which essentially tracks the previous frame to estimate the pose. Because the pose of the current frame is obtained on the basis of the previous frame or the previous reference keyframe through inter-frame matching, this process is described as tracking.

If the motion model has not been established, that is, when the current camera velocity is 0 or the system has just been relocated, the former method will be used to complete the tracking. For the second frame, the velocity is 0 at this time, the reference keyframe, which for this place is actually the first frame, will be tracked. By matching the feature points of the current frame with the reference keyframe, it can be associated with the 3D map point corresponding to the feature point on the reference keyframe, that is to say, it has the 2D information on the current frame and the 3D information of conjugate point on the reference keyframe. So a PnP problem can be constructed in which the pose of the previous frame is the initial pose, and get the camera pose through motion-only BA 2.3 by minimizing the reprojection error. The process is illustrated in Figure 4.2.

In the present case, the conjugate points $p_1$ and $p_2$ are obtained by feature point matching, and the 3D map point is known as $\boldsymbol{P}$. What needs to be solved is the camera pose $\boldsymbol{R}, \boldsymbol{t}$, can also be expressed in homogeneous form as $\boldsymbol{T}$. Now suppose the coordinates of the 3D point $\boldsymbol{P}_i = [X_i, Y_i, Z_i]^T$. The projection of $\boldsymbol{P}$ on the current frame is $\hat{p}_2$. The conjugate point $p_2$ expressed in image coordinate system $\boldsymbol{u}_i = [u_i, v_i]^T$. Their relationship is

$$s_i \boldsymbol{u}_i = \boldsymbol{K}\boldsymbol{T}\boldsymbol{P}_i \tag{4.1}$$

where $s$ is the scale factor. There is an error in this equation due to the unknown camera pose and the noise of the observation point. Therefore, we sum the errors, construct a least squares problem, and then find the best camera pose that minimizes it. The process can be illustrated in Equation 4.2.

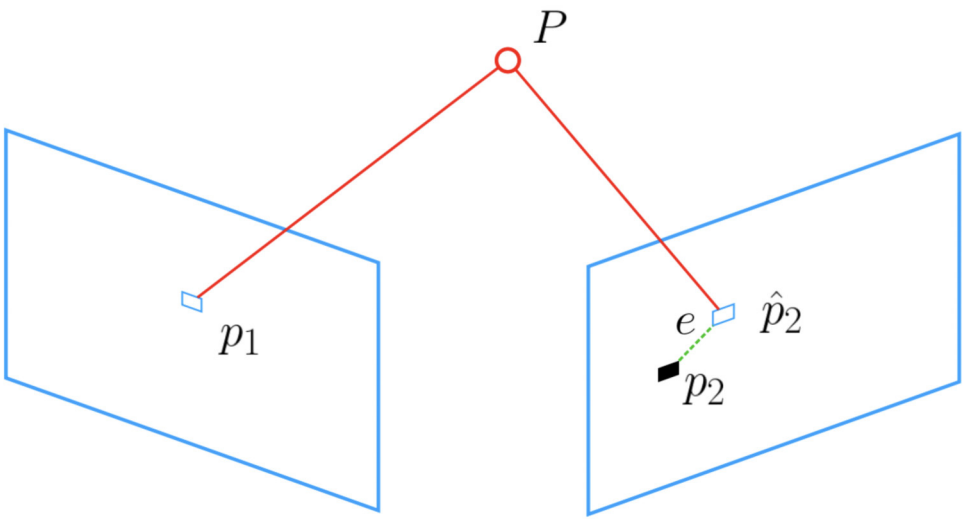



Figure 4.2: Reprojection error. The 3D point $P$. Conjugate image points $p_1, p_2$. The projection $\hat{p}_2$ of $P$ on the image. The optimal estimate of the camera pose is obtained by iteratively reducing the error $e$

$$\boldsymbol{T}^* = \arg\min_{\boldsymbol{T}} \frac{1}{2} \sum_{i=1}^{n} \left\| \boldsymbol{u}_i - \frac{1}{s_i} \boldsymbol{K} \boldsymbol{T} \boldsymbol{P}_i \right\|_2^2 \tag{4.2}$$

This error term is the difference between the projected position of the 3D point and the observed position, so it is called the reprojection error. So adjusting the pose of the camera to make this erroe smaller. However, since this adjustment needs to consider many points, the error at the end of each point is usually not exactly zero.

For the case of not tracking the reference keyframe, it is the second case, tracking the motion model. The motion model is based on the assumption of a constant velocity motion model. That is to say, the motion between the current frame and the previous frame is the same as the motion between the previous frame and its previous frame. Therefore, the pose of the current frame is set as the initial value according to the motion model, and then the PnP problem is constructed and solved by motion-only BA 2.3.

### 4.2.2 Local Map Tracking

In the tracking thread, camera poses are estimated through inter-frame matching, optimizing the 3D-2D reprojection of error to obtain the pose of the current frame. The pose obtained in this way is not reliable. It only uses the information of two frames of data. If the quality of the previous frame is too poor, for example, there is a large error for the pose of the reference frame or the number of feature points is small, the reliability of the pose obtained is low. Therefore, to utilize

more information, it is necessary to further match and optimize the current frame with the local map. As the camera moves, we add new keyframes and 3D map points to the local map to maintain the local map, so that even if there is a problem with a frame during the tracking process, using the local map, we can still find the correct pose for those frames afterward. In the motion model, the 3D points of the previous frame are projected to the current frame. However, in local map tracking, the 3D points in the local map (excluding the 3D points of the previous frame) and within the field of view of the current frame are projected to the current frame to construct the 3D-2D reprojection error and optimize the solution.

### 4.2.3 Keyframe Creation

The final step is to determine if the current frame is generated as a new keyframe. Keyframes are essentially transitions from normal frames. The purpose of generating keyframes is to appropriately reduce information redundancy because during the back-end global optimization process, optimizing the map points will also optimize the pose of the camera, that is, the pose of the frame, so it will be from the normal frame select some representative keyframes among them, reduce the optimization variables and reduce the amount of calculation.

In ORB-SLAM (Mur-Artal et al., 2015), the strategy for deciding whether to insert a new keyframe is:

1. More than 20 frames must have passed from the last global relocalization.
2. The local mapping thread is not busy, or more than 20 frames have passed since the last keyframe was insertion.
3. The current frame tracks at least 50 points.
4. The number of points tracked in the current frame is less than 90% of the reference keyframe. This means that the current frame and the reference keyframe share less than 90% of the visible points of the reference keyframe.

In general, the selection of keyframes should satisfy both the need for good quality of the current frame itself, such as having enough feature points and as even a distribution as possible and the need to achieve both constraints and as little redundancy as possible between the current frame and other keyframes.

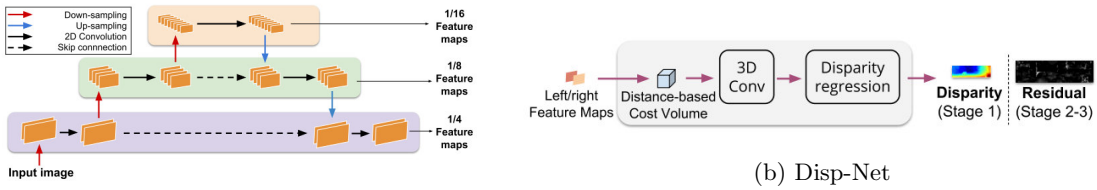
## 4.3 Depth Estimation Module

In the depth estimation thread, the keyframe transmitted from the local mapping is used as input here. The index of the original RGB image is recorded in the keyframe, so the corresponding stereo

image is located through the keyframe and input to the AnyNet 2.4.

### 4.3.1 Disparity Estimation

U-Net is used as a feature extractor to extract the features of the left and right images by sharing weights. Prior to applying convolution filters, the input image is first downscaled using max pooling or stridden convolution. The previously acquired lower-resolution feature maps are up-sampled and integrated into the current feature map before being sent into the final convolutional layer as illustrated in Figure 4.3.




(a) U-Net

(b) Disp-Net

Figure 4.3: U-net and Disp-net diagram (Wang et al., 2019). U-Net is used to extract feature maps of different resolutions. Disp-Net is used to estimate the disparity map in stage 1, and the residual map is calculated in stage 2&3.

When employing a CNN to predict a disparity map, the computational cost is typically cubic with image resolution and linear with maximal disparity taken into account (Kendall et al., 2017). To achieve the minimum calculation time, the feature map with the minimum resolution (1/16) produced by U-Net is used to construct the 3D cost volume in Disp-Net 4.3. In this process, the same spatial scaling factor is considered, that is, the maximum considered disparity range is 1/16 of the full range. In Disp-Net, each pixel $(i, j)$ in the left image can be represented as a feature vector $\mathbf{p}^L_{i,j}$, where dimension $\alpha$ corresponds to the pixel $(i, j)$ in the $\alpha^{th}$ input feature map associated with the left image. Similarly, $\mathbf{p}^R_{i,j-k}$ can be defined. The distance-based cost volume $C_{i,j,k}$ is then as the $L_1$ distance between the two vectors that shown in the following Equation 4.3:

$$C_{i,j,k} = \left\| \mathbf{p}^L_{i,j} - \mathbf{p}^R_{i,j-k} \right\|_1 \tag{4.3}$$

The distance-based cost volume is often not perfect, even if the learned features are used in the feature extraction phase. Therefore, after obtaining the cost volume, it is also expected to improve the disparity estimation by learning regularization functions to take contextual information into account in the cost volume. The use of 3D convolution operation is used to filter and refine the cost volume quantity. It will be recovered by weighted averaging 4.4 if a disparity $k$ clearly has the lowest cost, i.e. it is the only good match. The average of the viable alternatives will be the result

if there is ambiguity Wang et al. (2019).

$$\hat{D}_{i,j} = \sum_{k=0}^{M} k \times \frac{\exp\left(-C_{i,j,k}\right)}{\sum_{k'=0}^{M} \exp\left(-C_{i,j,k'}\right)} \tag{4.4}$$

Three stages and three Disp-Net modules do not add a lot of extra calculations due to the small number of three-dimensional convolutional layers. In stage 1, Disp-Net is used to obtain the disparity map, and in stage 2&3, it is used to calculate the residual maps. The low-resolution disparity map output in stage 1 is upsampled to the size in stage 2 and then used to warp the feature map(1/8) of stage 2. The wrapped right feature maps should match the left feature maps if the disparity estimate is accurate. However, due to errors, there are still mismatched pixels. Then this error can be improved by calculating the residual map. The calculation of the residual map is similar to the calculation of the disparity map, which is formulated in the following equation:

$$C_{i,j,k} = \left\| \mathbf{p}_{i,j} - \mathbf{p}_{i,j-k+2} \right\|_1 \tag{4.5}$$

After 3 stages, the resulting disparity map (1/4) will eventually be upsampled back to the original resolution. Due to the limitation of hardware conditions. Here, SP-Net 2.4 is not used to further refine the disparity map, which is completely optional. Disparity map with full resolution, which will be used in the next dense mapping module.

### 4.3.2 Uncertainty Estimation

CVA-Net 2.5 is treated as a separate module and is loosely coupled into AnyNet. In stage 1 of AnyNet, two cost volumes with different resolutions are generated at the same time, and the low resolution (1/16 of the original resolution) still serves for the disparity estimation mission, according to its original workflow. The high-resolution (1/4 of the original resolution) is input into the CVA module, and the output is the uncertainty map of the same size as the input. To achieve the same size as the full resolution disparity, it is upsampled. The full resolution cost volume is not chosen here because it is necessary to not consume too many computing resources as much as possible. The use of an uncertainty map is introduced in Section 4.4.4.

### 4.3.3 Loss

All stages are trained concurrently, but the losses are weighted differently, with weights of $\lambda 1 = 1/4, \lambda 2 = 1/2, \lambda 3 = 1$ respectively (Wang et al., 2019). When not using the CVA module to predict uncertainty. A supervised learning approach is used to train the model, using ground truth data as labels. Using L1 loss 4.6 to describe the gap between the predicted disparity value and the ground truth.

$$\mathcal{L}_1 = \frac{1}{n} \sum_{i=1}^{n} |y_i - f(x_i)| \tag{4.6}$$

When using the CVA module, the tasks of predicting disparity and predicting uncertainty are performed jointly. For a Bayesian interpretation of the task of learning to predict uncertainty. This is typically accomplished by defining a probability distribution to represent the uncertainty of data, the likelihood of which is increased during training. These parameters of the probability distribution are estimated disparities and uncertainty values, and its observations are ground truth disparities. By using this formula, an arbitrary uncertainty can be implicitly learned as the variance or standard deviation (depending on the probability distribution that is selected), avoiding the need to refer to the uncertainty (Mehltretter and Heipke, 2021). Therefore, the Laplace distribution is chosen to describe the uncertainty here. Therefore, the Laplace distribution is chosen to describe the uncertainty here.

To enable the use of common optimizers, the objective of the deep learning-based optimization process is formulated as the negative log-likelihood of this distribution:

$$-\log p\left(\hat{d}_i \mid d_i\right) \propto \frac{\sqrt{2}}{\sigma_i}\left|d_i-\hat{d}_i\right|+\log\left(\sigma_i\right) \tag{4.7}$$

where $d$ is the estimated and $\hat{d}$ the ground truth disparity, while $\sigma$ is the standard deviation of the assumed Laplace distribution representing the uncertainty for the respective pixel. In the loss function, $s = log(\sigma)$ is substituted. With this modification, the network is trained to predict the log standard deviation, which increases the numerical stability of the training process and prevents the loss function from being divided by zero (Mehltretter and Heipke, 2021). Thus, the Laplacian loss function is defined as:

$$\mathcal{L}_{lap}=\frac{1}{N}\sum_{i=1}^{N}\frac{\sqrt{2}}{\exp\left(s_i\right)}\left|d_i-\hat{d}_i\right|+s_i \tag{4.8}$$

## 4.4 Mapping Module

Contrary to ORB-SLAM2, which maps sparse feature points to a 3D map, continuous dense depth measurements must be accumulated into a continuously improving 3D model here. Finding the relationship between depth maps, removing outliers, and fusing them into the overall model is the main task here. Existing online techniques either compromise on the scale to produce better reconstructions of small objects or scenes. Limit the scope of active reconstruction or trade real-time performance and/or quality to handle larger scenarios (Keller et al., 2013). The point-based approach is used as the best compromise between precision and memory cost, as well as the practicality of integration with the most common SLAMs, among the voxel-based, point-based, and depth-based approaches (Wang et al., 2018).

### 4.4.1 Pointcloud Mapping

It is simple to determine the mapping relationship between the 2D image and the 3D point cloud based on the pinhole imaging concept if the depth information and camera pose are known. I only map keyframes in my work because of the need for real-time performance and the limited amount of computational power. After the keyframe is filtered in the local thread, it will be passed to the depth estimation thread to estimate its depth. After obtaining the depth, it will be passed to the dense mapping thread for the parallel mapping point cloud. The mapping formulation is shown in Equation 4.9,

$$\mathbf{P}_w = T_{cw}^{-1}\pi(\mathbf{u}, d), \tag{4.9}$$

where $\mathbf{u} = (u, v, 1)^T$ is the homogeneous representation of a pixel, $d$ is depth, and $\pi(\mathbf{u})$ is the back-projection from image to camera coordinate, i.e. $\pi(\mathbf{u}, d) = K^{-1}d\mathbf{u}$, where $K$ is the intrinsics matrix.

The relationship between disparity and depth is given by the stereo matching formula4.10. Under the premise that the camera intrinsic parameters $f$ have been obtained through camera calibration and the baseline distance $B$ between the two cameras is known, and for different data sets, the maximum disparity range considered is usually fixed. Therefore, the depth range that can be measured can be theoretically fixed. In terms of distance, the learned depth is substantially more erroneous. Set the maximum and minimum depth thresholds $d_{max}, d_{min}$ when mapping the point cloud to make sure it is as accurate as feasible. No mapping is done if the depth is greater than the maximum threshold or lower than the minimum threshold.

$$D = \frac{fB}{d} \tag{4.10}$$

### 4.4.2 Pointcloud Association

Combining the obtained poses, the global model created by mapping the depth map to a point cloud in the world coordinate system and then performing simple stitching is redundant and overlapping. Furthermore, applying the ICP-like algorithms for point cloud registration is challenging due to the imperfection of the depth map produced by the learning network (Wang et al., 2018). As a result, before point cloud fusion, point cloud association is a crucial stage.

The covisible keyframes of the current keyframe are kept as local keyframes and the points that the current keyframe and its covisible keyframes may see together are associated, which is modeled after the keyframe technique in ORB-SLAM2. By mapping pixels, $i$ in the present keyframe to all covisible keyframes $k$ for association in accordance with the transformation matrix $T_i^k$ may determine whether a certain point is visible in covisible keyframes. As the following Equation 4.11,

$$\mathbf{u}^k = g\left(KT_i^k\pi\left(\mathbf{u}^i, d\right)\right), \tag{4.11}$$

where $g(\mathbf{x}) = (x/z, y/z)^T$. Additionally, a mapping $M : \mathbf{u} \Rightarrow P_w$ is kept between the pixels on each keyframe and its corresponding 3D points. When a new keyframe $i$ is generated, the pixel $u^i$ on the keyframe and the corresponding 3D point $P_i$. Visibility on keyframe $k$ is checked by Equation 4.11. If it is visible, it means that the 3D point $P_i$ is associated with the 3D point $P_k$, namely via matching $P_i \Rightarrow u^i \Rightarrow u^k \Rightarrow P_k$.

### 4.4.3 Pointcloud Fusion

To produce a continuous, low-noise global 3D model. I used the fusion strategy in the work of (Wang et al., 2018). A confidence $\mu_c$, an average-weight $\omega$, and a global location $P_w$ are used to represent each 3D point. Whenever a new observation becomes available at keyframe $i$ a probabilistic filter derived from a voxel-based fusion method (Newcombe et al., 2011a) is used.

The process of point fusion can be simply described as updating the points that are covisible and adding new points that are not covisible, finally mapping the point with high confidence. When a new keyframe is generated, data association is performed in the covisible keyframes. If associated points are found, the points are merged with the new point using a weighted average. If no corresponding points are found, the new points are added temporarily. And finally, the confidence of each point determines whether the point is in a stable state. If the confidence of the point is greater than a certain threshold, the point is in a stable state, otherwise, it is in an unstable state. The threshold can be adjusted. The purpose of this is to further filter out points with high confidence and finally be mapped. As shown in the Equation 4.12,

$$\begin{aligned} \mathbf{P}_w^n &= \frac{\omega \mathbf{P}_w + \omega^0 \left(T_{cw}^i\right)^{-1} \pi\left(\mathbf{u}^i\right)}{\omega + \omega^0} \\ \mu_c^n &= \frac{\omega \mu_c + \omega^0 \left\| \left(T_{cw}^i\right)^{-1} \pi\left(\mathbf{u}^i\right) - \mathbf{P}_w \right\|}{\omega + \omega^0} \\ \omega^n &= \min\left(\omega + \omega^0, W_\varsigma\right) \end{aligned} \tag{4.12}$$

where $\mathbf{P}_w^n$ means the updated position, $\mathbf{P}_w$ and $\omega$ are the old position and weight of the associated point, respectively. $\omega^0$ is the initial weight of the new association point and configured to constant value 1 empirically. The new position of the association point is obtained by mapping a certain pixel $\mathbf{u}^i$ on the new keyframe. The initial confidence of the points is set to 0.8 empirically, and the confidence of the new associated point is the Euclidean distance between these two points. Both the updated location $\mathbf{P}_w^n$ and confidence $\mu_c^n$ are fused by weighted averaging. The updated weight is obtained by adding the old weight to the initial weight, while setting a truncation threshold $W_\varsigma$.

Figure 4.4 depicts the fusion procedure. Both the new keyframe $kF_i$ and the current keyframe $kF_k$ are covisible keyframes. This indicates that a portion of the points (red) are visible in both

frames. Naturally, the association between points is carried out by Equation 4.11 to check their co-visibility before knowing which points are covisible. Equation 4.12 is then used to execute the fusion and update for these covisible points. The other portion of the points (black, yellow) that keyframe $kF_i$ can see but $kF_k$ cannot see are known as non-covisible points, and they are immediately added to the map and updated upon the generation of the subsequent new keyframe $kF_{i+1}$. Finally, only mapping the stable points.

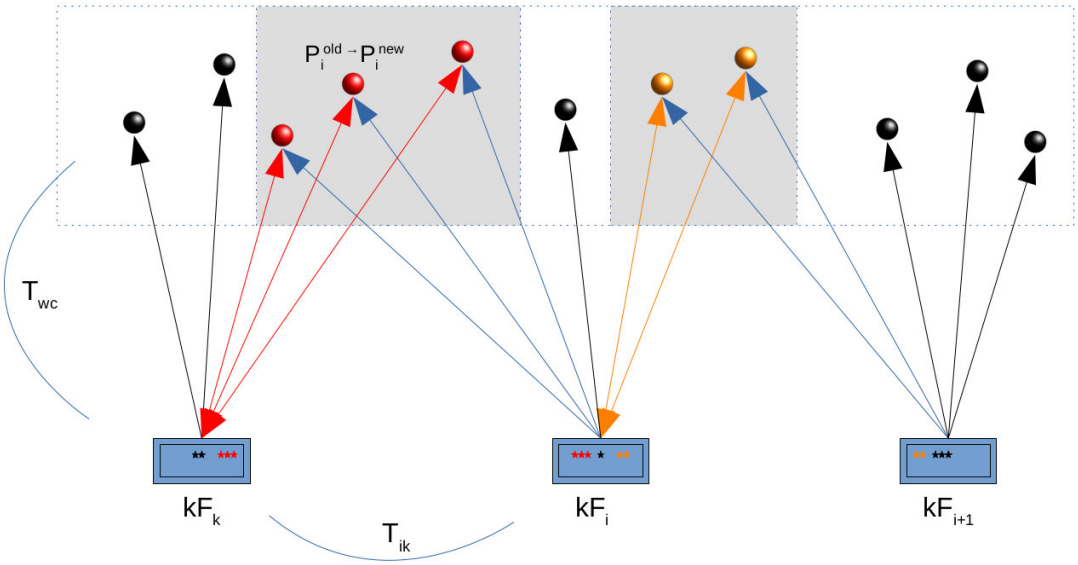



Figure 4.4: Point cloud fusion. The points (red) that can be seen by the current key frame $kF_k$ and the new keyframe $kF_i$ are fused. The points (black, yellow) that can be seen by keyframe $kF_i$ but cannot be seen by $kF_k$ are directly added to the map and updated after the next new keyframe $kF_{i+1}$ is generated.

### 4.4.4 Gausian Fusion

On the basis of the above, the uncertainty map estimated by CVA-Net can also be used in the fusion of point clouds. The principle is similar to the above fusion method, which is to fuse and update the existing and covisible point clouds and add non-covisible points. Finally, only mapping the stable with low uncertainty.

Depth estimation for a pixel can also be modeled as a transition estimation problem. As a simple assumption, I assume that its depth distribution satisfies a Gaussian distribution and the depth $d$ of a certain pixel obeys Gaussian distribution. The scope of application is limited here, and the influence of the direction of view is not considered.

$$P(d) = N(\mu, \sigma^2), \tag{4.13}$$

with $\mu$ is the mean representing the estimated depth, $\sigma$ is the standard deviation representing the estimated uncertainty. What needs to be explained is that the mean value described in CVA-Net (Mehltretter and Heipke, 2021) is an estimated disparity, which is represented by depth here. After obtaining the disparity, switch to the representation of depth and uncertainty. Since the concept of co-visibility, information fusion can be performed between the current keyframe and the co-visible keyframes, that is, the multiplication of two Gaussian distributions. It is known from Section **??**, after multiplying two Gaussian distributions, the distribution still satisfies the Gaussian distribution. The mean and variance after fusion are as follows:

$$\mu_{fuse} = \frac{\mu_{ref}\sigma_{cur}^2 + \mu_{curr}\sigma_{ref}^2}{\sigma_{ref}^2 + \sigma_{curr}^2}, \quad \sigma_{fuse}^2 = \frac{\sigma_{ref}^2\sigma_{curr}^2}{\sigma_{ref}^2 + \sigma_{curr}^2}, \tag{4.14}$$

with $\mu_{curr}$ and $\sigma_{cur}$ is the depth and uncertainty corresponding to the pixel on the current keyframe. $\mu_{ref}$ and $\sigma_{ref}$ are the depth and uncertainty of the corresponding pixels on the co-covisible keyframe obtained through the previous point cloud association. There are only observation equations and no motion equations, so the depth here only uses the information fusion part, and there is no complete prediction and update, like in the Kalman filter. Unlike the update process in Equation 4.12, here the fusion of "old" points and "new" points with the same weight.

# 5 Experimental Setup

In this Chapter, I present my experimental arrangement. Among them, the open-source dataset KITTI I used was introduced in Section 5.1. It is a frequently used dataset in the field of autonomous driving and mobile robotics. In Section 5.2, I introduce my training procedure for AnyNet. After fine-tuning the trained AnyNet, I saved it and exported it to the SLAM system. In Section 5.3, in order to reasonably evaluate my proposed SLAM system, I set up five variables to evaluate the performance of the system and locate the impact of different modules on the system. Section 5.4 introduces different evaluation metrics, which are used to quantitatively evaluate the performance of different modules of the system.

## 5.1 Dataset

The KITTI 2012 dataset (Geiger et al., 2012) was utilized in my research. This dataset consists of a stereo image stream captured by two binocular cameras (two grays and two colors) of a real scene. A Velodyne 3D laser scanner was used to obtain the semi-dense Ground Truth. 194 pairs of training images and 195 pairs of test images make up the disparity estimation project. In the odometry project, 22 stereo sequences are included. The ground truth of the trajectory is present in sequences 00–10, but not in 11–21. The dataset includes some surfaces with reflections, perspective, and some scenes with significant displacements, such as those with fast motion and various lighting effects. This dataset is therefore frequently used in research on robotics and autonomous driving.

## 5.2 Training Procedure

In the depth estimation module, the learning-based end-to-end network AnyNet is exploited to estimate scene disparity, and the CVA module is used to estimate the uncertainty. AnyNet is trained on the KITTI dataset. Since the test set in the KITTI dataset does not provide ground truth, I use 70% of the training set as the training set in my work, 20% as the validation set, and the remaining 10% of the data as the test set. On hardware, I use NVIDIA GeForce RTX 3090 GPU for my training work. The choice of deep learning framework is PyTorch.

The input data need to be preprocessed. Before being added to the pipeline, each pair of stereo images needed to be cropped to $1232 * 368$ and implemented with a normalization procedure using

mean $[0.485, 0.456, 0.406]$ and standard deviation $[0.229, 0.224, 0.225]$. The mean and standard deviation here is from Imagenet, and it is a common practice to use the mean and standard deviation of Imagenet (Deng et al., 2009). They are calculated from millions of images. Normalization can make the distribution of data more uniform, reduce the possibility of the model learning the data distribution, and improve the generalization ability of the model.

The maximum disparity is set to 192 pixels, corresponding to a Stage 1 cost volume depth of $M = 192/16 = 12$. The residual range in Stages 2 and Stage 3 is 2, corresponding to 16 pixels in Stage 2 and 8 pixels in Stage 3. All stages are trained concurrently, but the losses are weighted differently, with weights of $\lambda 1 = 1/4, \lambda 2 = 1/2, \lambda 3 = 1$, respectively (Wang et al., 2019). The optimizer Adam (Kingma and Ba, 2014) was used to carry out the optimization with an initial learning rate of $5e - 4$. Due to the small amount of dataset of the disparity estimation project, I set the initial training epochs to 300. After 200 epochs, the learning rate is divided by 10. The training batch size is empirically set to 6, however, this may obviously be changed depending on the hardware setup. I got this by starting at 8 and discovered that 6 could converge, as quickly as possible, without exceeding the limitation of GPU memory.

The optimum number of training epochs is determined by early stopping, which means that if the validation loss does not decrease in three consecutive epochs, the training procedure is terminated, and the set of parameters associated with the epoch with the lowest validation loss is used for testing.

In order to shorten the training time and also obtain a better convergent model, fine-tuning the pre-trained model is a common method. A pre-trained model provided in the work of AnyNet (Wang et al., 2019) and subsequently fine-tuned on the KITTI dataset is implemented. The pre-training process was done on the Scene Flow dataset (Menze and Geiger, 2015).

When AnyNet includes the CVA module, the model must be trained separately because uncertainty in the disparity estimation process must be considered. The KITTI dataset continues to be used. Fine-tune using a pre-trained model and the same hardware, hyperparameter, and optimizer settings as before. The difference is that only the optimization of the disparity estimation task is considered in the first 250 epochs, and the L1 loss function is used. The Laplace loss function is used for the final 50 rounds, which also takes into account the optimization of uncertainty estimation. In comparison to directly optimizing both at the same time, the goal of this is to better maintain the convergence behavior, resulting in better overall results (Mehltretter, 2022).

## 5.3 Evaluation Strategy

For evaluating the depth estimation module, I isolated this module for control experiments. Take SGBM as the control group, and Anynet and Anynet-cva as the experimental group. Each group of subjects entered the same ten pairs of stereo images, and performed the same experiment 5 times, to get the averaged results. Qualitative and quantitative evaluations of the three stereo-matching methods were performed, respectively.

For camera pose estimation, an intuitive experimental result is a trajectory and an accuracy of the pose. Since the front-end used in each variant, the pose estimation module, is the same, I did not set a control group. At the same time, in different scenarios, the accuracy of pose estimation is bound to be affected, but this is not the focus of my work, so I do not conduct an experiment on all sequences in the KITI dataset, take one group sequences are sufficient to evaluate the estimated pose.

Evaluation of dense reconstructions is challenging work. On the one hand, for large outdoor scenes, dense reconstruction often occupies a large amount of memory, which is not easy to evaluate qualitatively or quantitatively. On the other hand, the standards used are often different. For the purpose of reasonably evaluating the ability of dense reconstructions within my experimental resources. I first unified the experimental conditions, and I chose an equal number of input image sequences for dense reconstruction. Then in order to be reasonably able to target the evaluation of different modules, I set up four different variants. While obtaining the overall performance of the system, I can also locate the impact of different modules on the entire system. At the same time, an additional set of variants with uncertainty estimation is made in order to obtain the influence of the CVA module on the system.

1 **SGBM-concat**: the traditional SGBM algorithm (Hirschmuller, 2007) is used for disparity estimation, and in the reconstruction process, the point cloud fusion is not used, and the keyframe point cloud is simply mapped to the three-dimensional space and then stitched.

2 **Anynet-concat**: replace the SGBM algorithm and use the stereo matching network AnyNet for disparity estimation.

3 **SGBM-prob**: in order to optimize the reconstruction process, a point cloud fusion strategy based on co-visibility and probability filtering is used. This is because the reconstructed model obtained by directly splicing the point clouds of the keyframes has point cloud redundancy and contains noise and outliners.

4 **Anynet-prob**: using AnyNet and probability filter at the same time, which is also the new SLAM method I proposed.

**Anynet-cva-gausian**: finally, in order to verify the influence of the uncertainty obtained by the CVA module on the reconstruction effect, in the disparity estimation, the AnyNet with the CVA module is used to estimate the disparity and uncertainty at the same time, while using the Gaussian fusion. I compare it with variant 4 alone and directly draw the impact of the CVA module on the system.

## 5.4 Metrics

Whether it is for the disparity estimation module or the reconstruction module or the evaluation trajectory, the results of the quantitative analysis are more convincing than the results seen by the naked eye. I used assessment metrics that are often used in related disciplines.

In the depth estimation module, my main concern is the estimation accuracy and efficiency of AnyNet. The disparity map obtained by stereo matching, under the premise of a relatively more accurate disparity map (that is, ground truth), uses widely used evaluation indicators to quantitatively evaluate its estimation results.

- **Pixel Error**: the percentage of pixels whose predicted disparity value exceeds a certain threshold among valid pixels ($d > 0$). Usually, thresholds are 1 pixel, 3 pixels, and 5 pixels. This metric allows us to directly determine the accuracy of our predicted disparity maps. I reported the percentage of pixels with errors larger than 3 disparities in both non-occluded (3-noc) and all regions (3-all).
- **End-Point-Error** (EPE): average disparity error in both non-occluded (EPE-noc) and all the pixels (EPE-all). is another commonly used metric.
- **Average Time** ($t_{ave}$) that is easy to count and intuitive to evaluate efficiency. It is the time taken by AnyNet from input to output, from beginning to end for pair of images. To reduce the impact of random errors, I tested 10 pairs of stereo images each time, experimented 5 times under the same conditions, and averaged the time it took for each pair of images.

For the estimated pose, the evaluation of accuracy is the most concern. In this process, we will inevitably encounter two accuracy indicators ATE and RPE (Zhang and Scaramuzza, 2018).

- **Absolute Pose Error** (APE): The absolute trajectory error is the direct difference between the estimated pose and the real pose, which can reflect the accuracy of the algorithm and the global consistency of the trajectory very intuitively. Note that the estimated pose and ground truth are usually not in the same coordinate system, so we need to align the two first. The APE for frame $i$ is defined as follows:

$$\begin{aligned} E_i &= P_{ref_i}^{-1} T_i P_{est_i} \\ APE_i &= ||trans(E_i)|| \end{aligned} \tag{5.1}$$

which $P_{ref_i}$ is the ground truth pose at frame $i$, $P_{est_i}$ is the estimated camera pose at frame $i$, and $T_i$ is the transformation matrix in between. Generally take its translation part for statistics, such as maximum, minimum, mean, and so on.

- **Relative pose error** (RPE): RPE mainly describes the accuracy of the pose difference between two frames with a fixed time difference $\delta$, which is defined as follows:

$$\begin{aligned} E_i &= (P_{ref_i}^{-1} P_{ref_{i+\delta}})(P_{est_i}^{-1} P_{i+\delta}) \\ RPE_i &= ||trans(E_i)|| \end{aligned} \tag{5.2}$$

  Similarly, after aligning with the timestamps, both the real pose and the estimated pose calculate the pose changes at the same time interval and then do difference to obtain the relative pose error, this criterion is suitable for estimating the drift of the system.

For the evaluation of the reconstruction module, since there is no dense ground truth in the KITTI dataset, I choose indirect to evaluate the effect of reconstructing point clouds. Therefore, I chose to use Cloud-to-Cloud (C2C) distance to quantify the distance between point clouds obtained by different variables. In my experiments, I used variants 3 and 4 as my reference point cloud respectively. Then use the target point cloud to register, compare, and calculate the C2C distance. The purpose of this is to get the completeness of the target point cloud and the distance between the two point clouds.

- **Cloud-to-Cloud** (C2C): C2C is used to determine the separation between two clouds or between a mesh and a point cloud. The fundamental C2C distance calculation algorithm determines the nearest neighbor distance between the comparison cloud dataset and the reference cloud. When determining the distance between two points, the nearest points in the reference cloud are searched for and their Euclidean distance is determined. This method is known as the nearest neighbor distance principle (Ahmad Fuad et al., 2018).

# 6 Results

The main content of this Chapter is my experimental results and analysis. In Section 6.1 I compare three disparity estimation methods used in my experiments. They are SGBM, AnyNet, and AnyNet-cva. In Section 6.2, I evaluate the performance of my proposed slam system on the 05 sequences in the KITTI dataset. Trajectory accuracy (i.e. estimated pose accuracy) is evaluated with APE and RPE, respectively. In Section 6.3 I first analyze the running time required by different SLAM modules in my SLAM system and get their impact on the system. The reconstruction results of the 5 variants were then compared.

## 6.1 Disparity Esimation

As shown in Figure 6.1, the disparity map estimated by AnyNet is generally accurate. For example, in row (c), the overall outline of the car and tree can be better discerned. Even regions with no ground truth can more accurately estimate disparity. The estimated disparity error is significantly larger for areas with few textures and high repetitions, such as the ground and the car body, in terms of local details. In the original image on the far left, for example, the white wall has few and repeated textures, resulting in an estimated disparity map that is not distinguishable.

At the same time, for the disparity estimation method used in my experiment, I did a comparative experiment. Input the same stereo image pair, compare the output disparity map, and evaluate it using different metrics. As shown in Figure 6.2, the estimated disparity map and the corresponding error map of the SGBM algorithm, AnyNet, and AnyNet-cva are respectively. The disparity map produced by the SGBM algorithm, as shown in the figure, has a relatively high degree of boundary discrimination, the smoothness of the adjacent areas is relatively good, and there will be no major mutation. However, there will be some empty values because some pixels cannot be successfully matched when stereo matching is performed. A portion of the field of view of the reference image and the target image cannot be viewed concurrently. As a result, some disparity values, such as the lower left corner, cannot be estimated. The empty value of an area is also responsible for the SGBM algorithm's relatively large overall pixel error. So the usual processing is to perform interpolation to complete these empty values.

The disparity map obtained from the SGBM method needs to be post-processed, and here I only

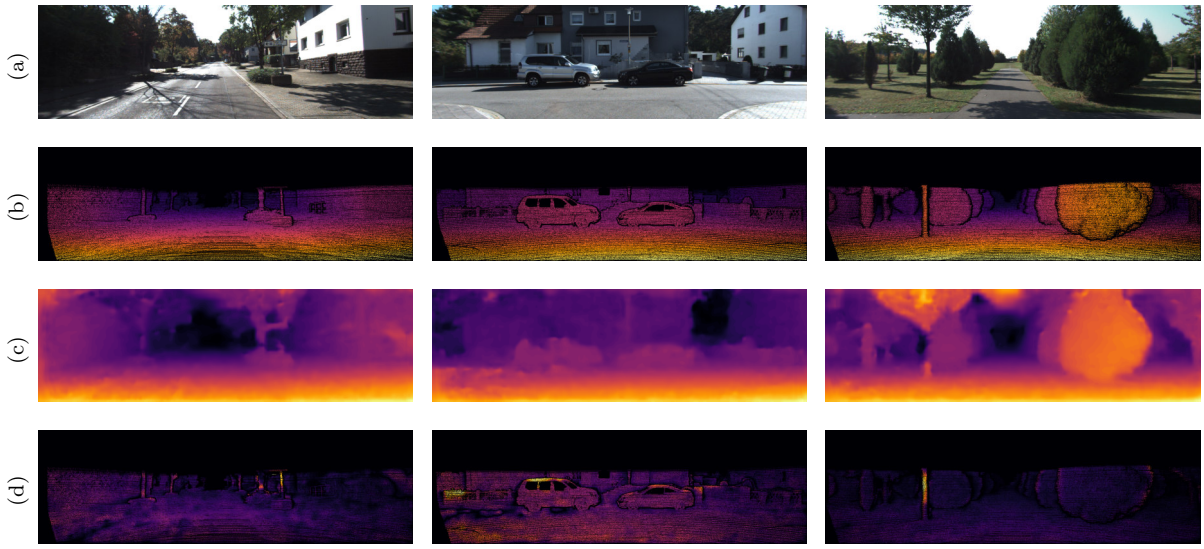
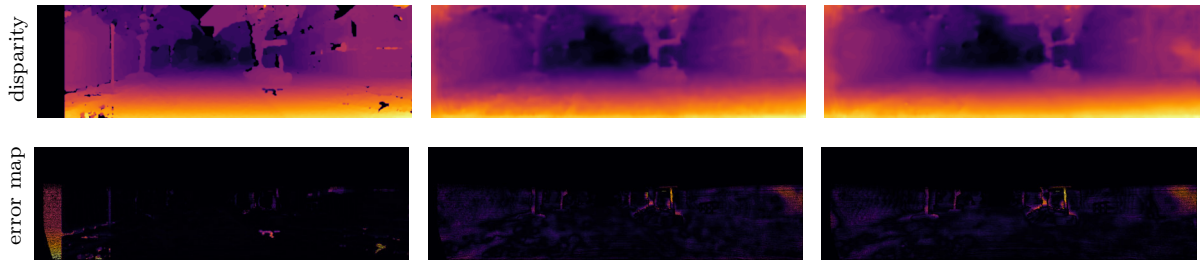



Figure 6.1: Inference results of AnyNet on the KITTI2012 dataset. Due to the limitation of hardware, SP-Net is not used for further disparity optimization. The output is the disparity map of stage 3. Row (a) is the reference image, (b) is the ground truth, and (c) is the predicted disparity map. The brighter the color, the larger the disparity value, and the closer the depth. For row (d), the error map (only consider pixels with valid disparity value), the brighter the color, the greater the error.

complete the empty value. Firstly, calculate the integral map of the disparity map and save the number of all accumulated pixel points at each integral value in the corresponding integral map (pixels at voids are not counted, because a pixel value of 0 at a void has no effect on the integral value and would instead smooth the image). Multi-level mean filtering is then used. First, a larger initial window is used to do the mean filtering, assigning values to the voids in large areas. Then the next filtering, the window size will be reduced to half of the original, using the original integration diagram filtering again, to the smaller holes assigned value (cover the original value) And so on, until the window size becomes $3 * 3$, then stop filtering, to get the final results. The use of multi-level filtering takes into account the fact that for initially larger areas of voids, more neighborhood values need to be referenced, which cannot be filled if smaller filter windows are used, while if all larger windows are used, the image will be severely smoothed. The filter window is therefore constantly adjusted according to the size of the voids. By first assigning values to all the voids with a large window and then using a gradual change to a smaller window filter to overwrite the original values, this ensures that the voids are filled in and that the image is not over-smoothed.

The disparity and error maps show no discernible difference between AnyNet and AnyNet-cva. However, both outperform the SGBM algorithm in terms of performance. Not only can dispar-




(a) SGBM (b) Anynet (c) AnyNet-cva

Figure 6.2: The disparity estimation results and corresponding error maps of the SGBM algorithm, AnyNet, and AnyNet-cva methods. The description of the colors is shown in Figure 6.1

ity estimates be obtained for non-common view regions, but the appearance of hole values is also avoided, because the learning method assigns a disparity value to each pixel, even if it is not very accurate.

From the quantitative results in Table 6.1, it can be seen intuitively that the modified AnyNet proposed in my work is better than SGBM in terms of prediction accuracy and prediction efficiency. The traditional stereo matching algorithm can only obtain the disparity of the successfully matched pixels. Therefore, a part of the disparity map obtained in this way has no disparity value. One reason is that the reference image and the target image have a part of the field of view that is not co-viewed, and the other reason is that the pixels that are not successfully matched will show empty values. Although the hole value interpolation process has been implemented, there are still a small number of hole values that have not been successfully completed. At the same time, because multi-level mean filtering is also used in the interpolation process, this will also smooth the disparity map. Other reasons include parameter settings of the SGBM algorithm. Nevertheless, it is still possible to obtain the preliminary conclusion that the estimation accuracy of AnyNet is better. In terms of efficiency, the average estimated time to obtain a disparity map is 0.09 seconds. It is also significantly faster than the SGBM algorithm.

When AnyNet uses the cva module, from the perspective of disparity accuracy, the impact is not large, and it is slightly improved. From the perspective of model optimization, better performance results indicate that the gradient has reached a better place. Before the 250th epoch, the L1 loss value fluctuated repeatedly, and the pixel error of the validation result is almost smooth, indicating that the gradient of the objective function is close to 0 at this time, or it is jumping left and right near a local minimum. After the 250th epoch, a Laplacian loss is used, which takes into account not only the difference between the estimated disparity value and the ground truth but also the log uncertainty. That is to say, an optimization item has been added to the objective function. Then the gradient may increase at once so that it jumps out of this local minimum. But from a global

| Method | 3-noc (%) | 3-all (%) | EPE-noc | EPE-all | Run-time (s) |
|---|---|---|---|---|---|
| SGBM | 14.13 | 15.43 | 4.21 | 4.94 | 0.25 |
| AnyNet | 9.58 | 9.92 | 1.65 | 1.69 | **0.09** |
| AnyNet-cva | **9.42** | **9.84** | **1.43** | **1.48** | 0.19 |

Table 6.1: Quantitative results of different stereo matching methods under the KITTI 2012 dataset. Among them, the threshold of pixel error is 3 pixels, EPE is the average pixel error, and the statistical area includes the non-occlusion area and all areas. The smaller the value of both, the higher the prediction accuracy.

point of view, the local minimum may be a sub-global minimum, so even if the gradient drops again, the change is not great. For the average time, due to the three-dimensional convolution of the cost volume in the CVA module, the extra computation introduced is relatively large, because the three-dimensional convolution is computationally expensive.

To sum up, choosing the lightweight AnyNet as the implementer of the disparity estimation module is feasible in terms of efficiency. Even if it only runs on the CPU, its efficiency is still competitive. However, the accuracy of disparity estimation is not so prominent. Fortunately, combined with the SLAM system in a loosely coupled manner, the learning network can be flexibly adjusted or replaced. n future work, it is possible to try to adjust the network structure of AnyNet. Most fundamentally, the resolution can be increased when constructing the cost volume in stage 1. Or try to construct a cost volume that better reflects the similarity between the left and right feature vectors. The construction of the cost volume usually plays a significant role in a stereo matching network.

## 6.2 Pose Estimation

The main purpose of visual odometry is to estimate the pose of the camera through inter-frame matching and frame-to-local map matching. A reasonable keyframe selection strategy, not only ensures that tracking is not easily lost but also improves the accuracy of pose estimation. In the experiments, the pose estimation module used by the four different variants is the same, so there are no comparative results for this module. But different metrics were used to evaluate the estimates. Quantitative and qualitative analysis results are obtained by saving the camera pose of each frame.

As shown in Figure 6.3, the dashed line is the ground truth trajectory and the solid line is the experimental result on KITTI dataset 05. Overall, the accuracy is very good, and it is basically in line with the ground truth. Especially at the beginning, when the error accumulation is not very large, the accuracy is very good. As the cumulative error increases later, it can be seen that there

will be a certain deviation. Although the cumulative error increases, the overall trajectory does not show a large deviation.

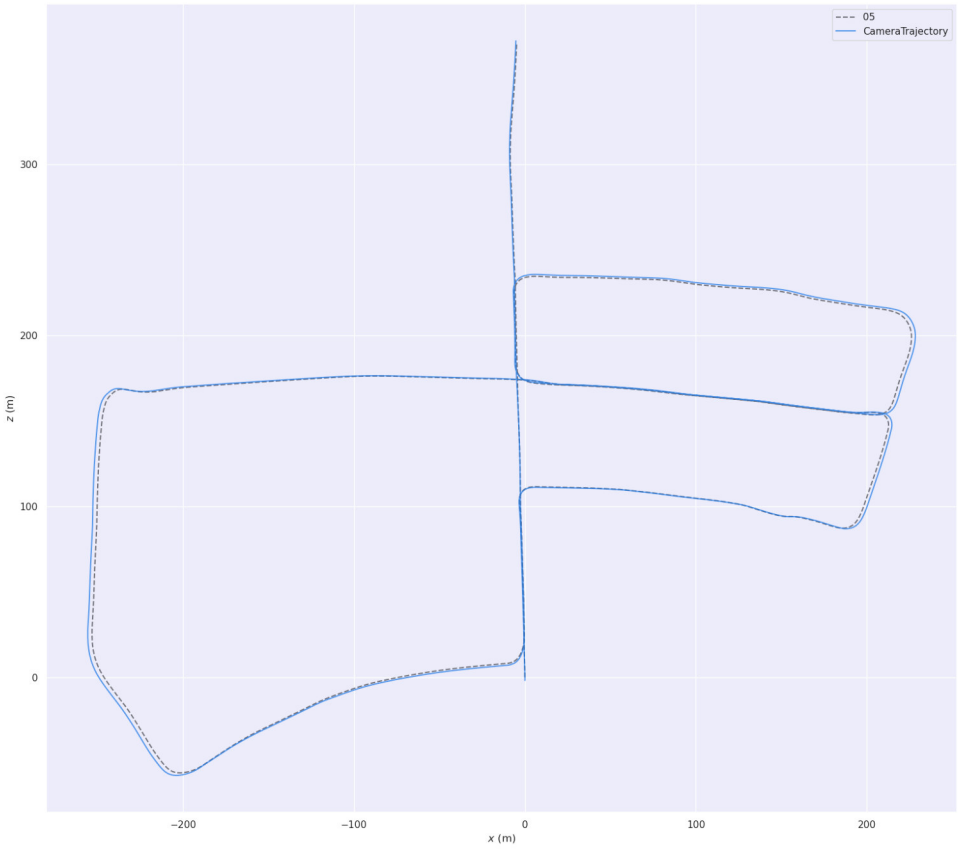



Figure 6.3: Ground truth (dashed line) and experimental results (solid line) on KITTI 05 dataset. The origin of the coordinate system is the starting point

The absolute pose error (APE) directly computes the difference between the true value of the camera pose and the estimated value of the pose estimation module. It can intuitively reflect the accuracy and global consistency of the trajectory. From Figure 6.4, the upper and lower figures are the distribution of APE in the trajectory map and the change of APE as the trajectory progresses. As can be seen from the above trajectory diagram, the average error of the trajectory is basically between 1-2m, and a few scenes such as turning places have a significant error, which reaches 2.6 meters. A reasonable explanation is that when turning, there is a less common view between adjacent frames, and fewer feature points, which increases the probability of tracking loss or incorrectly matched feature points. This leads to an increase in the pose estimation error. There are also possible reasons for the increased error caused by special scenes such as few scene textures or dynamic objects. By retrospectively inputting the original stereo image, it is found that, for

example, in the vicinity of sequence 2000, dynamic objects of vehicles and pedestrians have always appeared. When there are dynamic objects in the environment, it will bring wrong observation data to the system, for example, it will cause wrong feature point matching, which will cause the error of camera pose estimation to increase, and reduce the accuracy and robustness of the system. And a feasible method is to detect these dynamic objects and treat them as outliers and remove them.

RPE is essentially similar to the calculation method of APE, the difference is that RPE calculates the difference between two APEs. RPE only uses the change of pose and does not care about the estimated absolute pose, thus eliminating the influence of the estimated absolute position on the final result. As can be seen from the upper graph of Figure 6.5, the RPE in the full trajectory is relatively stable, with only a few places having prominent values. Corresponding to the figure below, you can view the change couple of RPE in more detail. It can be seen from this that the estimated overall change is relatively stable relative to the error change, but has a few large fluctuations. The reason may be the influence of the special scene mentioned above.

Both the specific and accurate values are shown in Table 6.2. Generally speaking, whether it is discussed from RPE or APE data, the estimated pose accuracy is relatively high. The average APE is only about 1.4 meters, and the average RPE is only about 0.016. This is a performance with high absolute accuracy and small fluctuations for a large outdoor scene with a trajectory range close to $400m * 400m$. However, only from the result of pose accuracy, compare with the SOTA results, neither the pure visual SLAM work nor the VIO (visual-inertial odometry) work is not yet competitive. The main reason can always be attributed to the shortcomings of the camera and the limitations of the stereo matching algorithm. For example, the impact of illumination changes the gray value of the image, which in turn affects the extraction and matching of feature points, and finally affects the estimation of the pose. An important work of ORB-SLAM is to use ORB feature points that are insensitive to rotation, scale, and illumination. This is enough to show that the work around the feature points influences the follow-up work. Then it is natural to be connected to the improvement of computer vision and deep learning in this area. The convolutional neural network extracts feature points to serve the front end of SLAM, which is theoretically feasible and has a lot of room for research and improvement. Many researchers have already started work in this area.

## 6.3 Reconstruction

If you do not consider the time cost and only pursue precision, there are many outstanding jobs. In other words, under the current situation of rapid development of hardware, excellent and impressive results have been achieved. Therefore, one of the original intentions of my work is to achieve

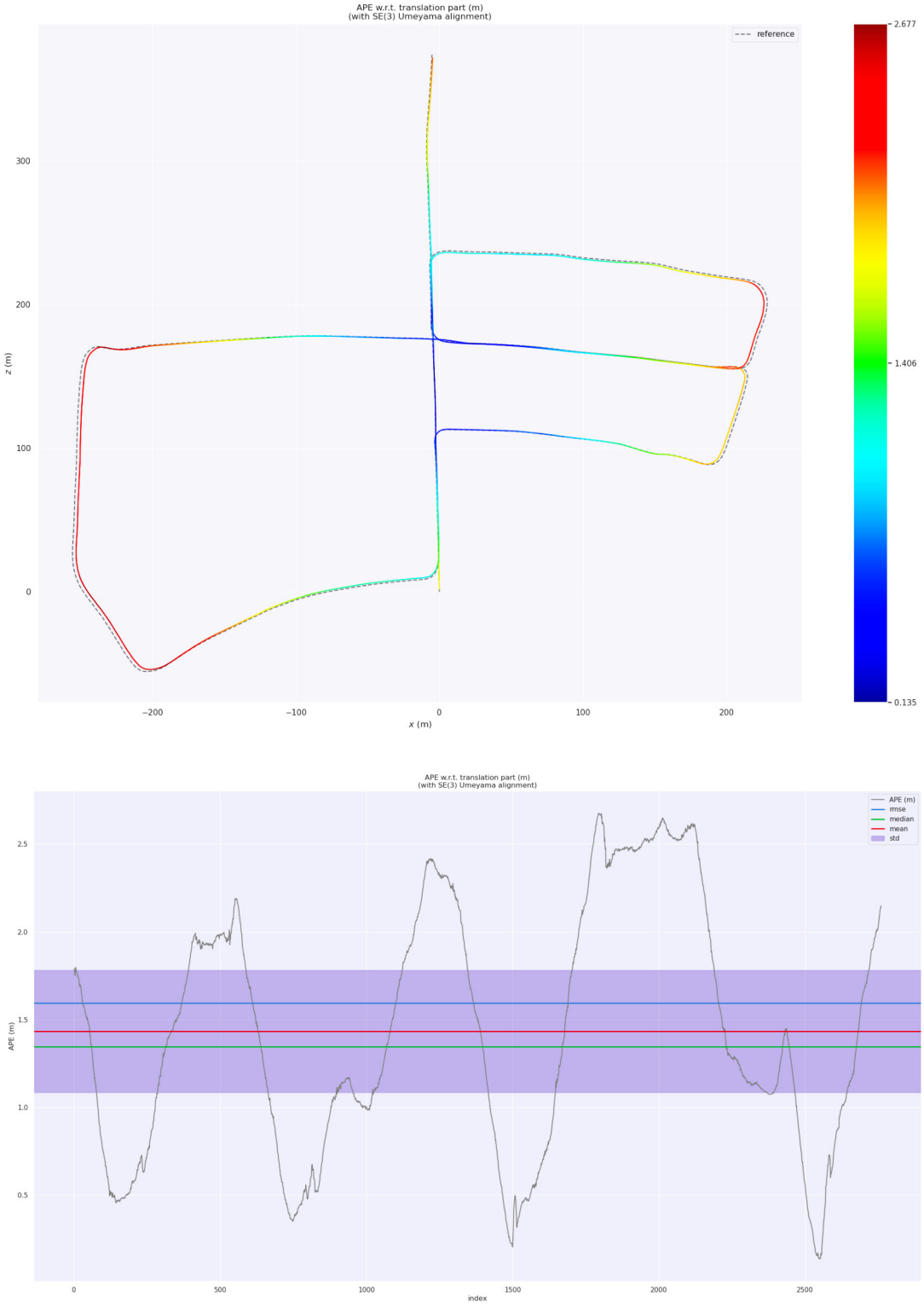



Figure 6.4: Absolute pose error (APE) representation in the trajectory graph and the corresponding APE change as the trajectory progresses. The origin of the coordinate system is the starting point. The index represents the number of frames.

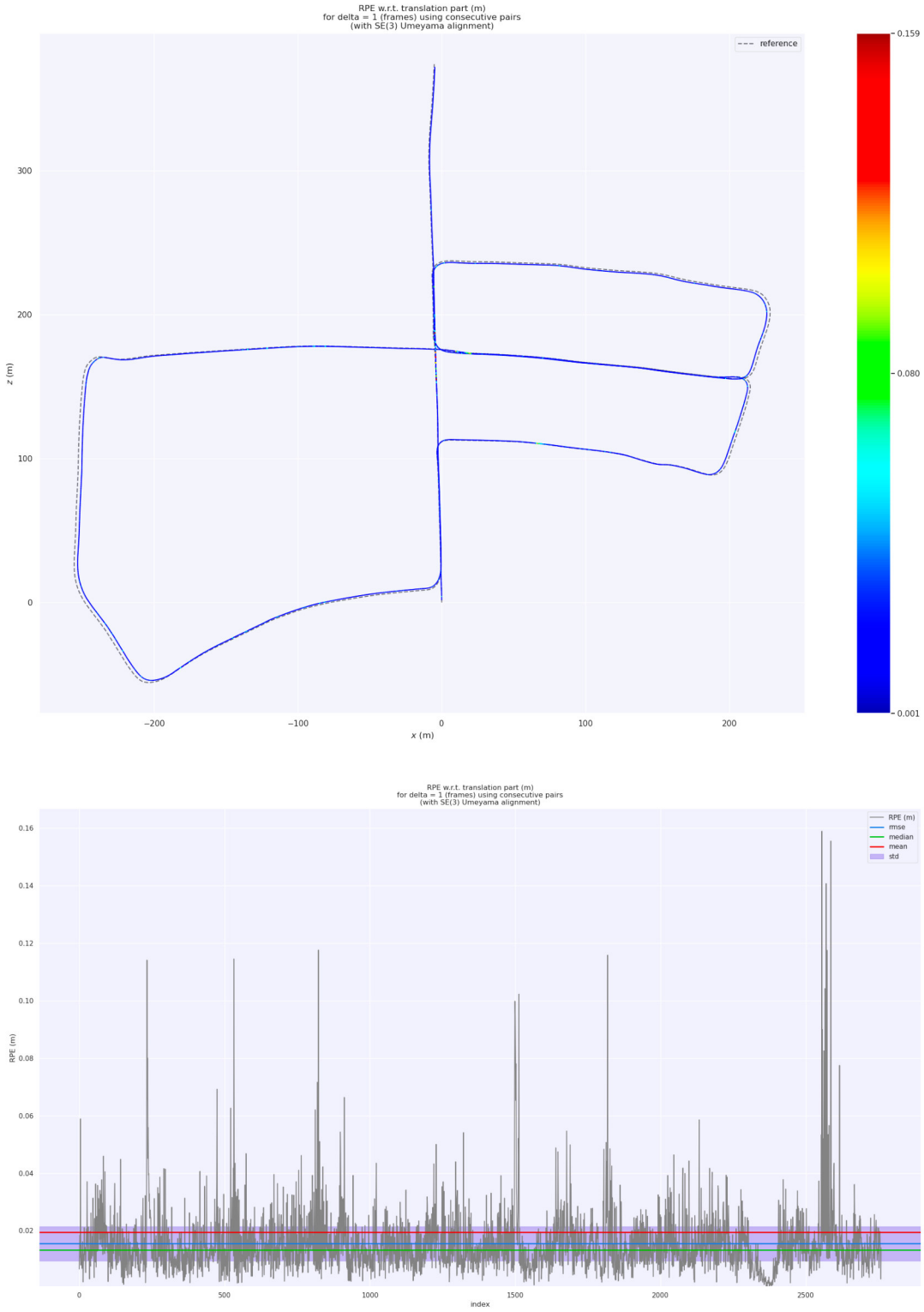



Figure 6.5: Relative pose error (RPE with interval 1 frame) representation in the trajectory graph and the corresponding change as the trajectory progresses. The origin of the coordinate system is the starting point. The index represents the number of frames.

| | MAX | MIN | MEAN | STD | MEDIAN | RMSE |
|---|---|---|---|---|---|---|
| APE | 2.677 | 0.135 | **1.434** | **0.698** | 1.346 | 1.595 |
| RPE | 0.159 | 0.001 | **0.016** | **0.012** | 0.013 | 0.020 |

Table 6.2: Maximum, Minimum, Mean and Standard Deviation, Median, Mean Squared Error of APE and RPE

dense reconstruction only on ordinary hardware without GPU support. Then efficiency is one of my main concerns.

First, from Figure 6.6, you can see an overall picture of one running time of the 5 variants in my experiment. I count the overall time of each module, and then divide it by the number of modules runs to get an average running time of different modules, which can be understood as the time consumption of locating each part of the dense mapping process at the same time. The exact time and different time standards are shown in Table 6.3. The most intuitive point is that the total duration of all variants does not exceed one second.

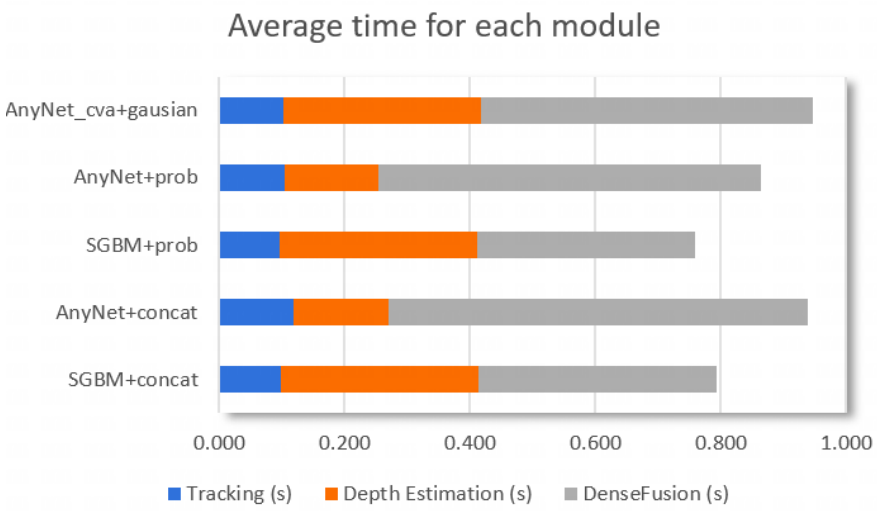



Figure 6.6: The average time that individual modules of different variants of the system are running. The specific time and other statistics are shown in detail in Table 6.3.

Then, it can be seen that the tracking duration in all variants is almost the same, and the time is very short around 0.1 seconds. I think this is a reasonable situation, because the work I have done is mainly in the depth estimation module and the dense reconstruction module, and the adjustment of the camera pose estimation module does not affect the efficiency of estimating pose. In the depth estimation part, the characteristic is that the duration of the variant using AnyNet is almost the same, and the others are about twice as long. This is consistent with the results I obtained

in my evaluation of depth estimation in Table 6.1. This again shows that the depth estimation module I applied has high efficiency. Finally, in the reconstruction module (DenseFusion in the figure), it is interesting that although the same reconstruction method is used, its efficiency varies depending on how the depth is acquired. For example, they all use the method of a concatenation of point clouds, but using the AnyNet variant, the reconstruction time is longer. The same is true for variants that use probabilistic filtering. So the experimental results tell me that the addition of AnyNet will make the reconstruction more time-consuming. This is actually understandable because it can estimate more pixels with disparity, and SGBM not only has hole values but also the non-covisible parts of the left and right views that cannot be estimated. Therefore, more pixels with estimated disparity will naturally introduce more points, and more calculations are required whether it is direct mapping or fusion. Considering that the same depth estimation method is used, the fusion update method of probability filtering can be seen to be more efficient than the direct connection method.

Regarding the last variant 5 which is with the CVA module, compared with AnyNet + prob, the most direct is that the depth estimation module spends more time, but the reason here is that the estimation uncertainty of the CVA module is introduced, which leads to more calculations. There is no obvious difference in the fusion part because, although the CVA module is introduced, it has no effect on the estimated disparity map, so the fusion object, that is, the point cloud is also close to the same, so it can be roughly considered that the fusion cost time is close.

Finally, from the experimental results of the running time of each module. To sum up, the addition of Anynet provides a more accurate and denser point cloud and also ensures higher efficiency. The fusion method based on probability filtering is more efficient than simply connecting point clouds.

| | Tracking (s) | | | Depth Estimation (s) | | | DenseFusion (s) | | | Mean Totall (s) |
|---|---|---|---|---|---|---|---|---|---|---|
| | Mean | Max | Min | Mean | Max | Min | Mean | Max | Min | |
| sgbm+con | 0.098 | 0.185 | 0.051 | 0.314 | 0.437 | 0.267 | 0.380 | 3.055 | 0.064 | 0.792 |
| any+con | 0.119 | 0.230 | 0.053 | 0.151 | 0.363 | 0.129 | 0.669 | 2.164 | 0.115 | 0.938 |
| sgbm+pro | 0.097 | 0.177 | 0.053 | 0.315 | 0.468 | 0.261 | 0.348 | 3.368 | 0.075 | 0.759 |
| any+pro | 0.104 | 0.223 | 0.057 | 0.151 | 0.345 | 0.124 | 0.608 | 4.229 | 0.118 | 0.863 |
| cva+gau | 0.102 | 0.226 | 0.062 | 0.316 | 0.424 | 0.204 | 0.530 | 1.606 | 0.202 | 0.947 |

Table 6.3: The time that individual modules of different variants of the system are running. The metrics of average time, maximum time, and minimum time are used respectively

Efficiency is one of the important criteria for evaluating SLAM. At the same time, in my work,

the effect of dense reconstruction is also an important indicator for me to consider. Considering hardware limitations and computational stress, in my experiments, only a part of the dataset was taken as input. The 5 variables were used to reconstruct and compare the results of the experiments.

Figure 6.7 shows the dense reconstruction results for these 5 variants. From left to right: SGBM+concat, AnyNet+concat, SGBM+prob, AnyNet+prob, AnyNet-cva+gausian. It can be seen intuitively that the reconstruction effect from left to right is gradually improving, at least in terms of visual effects. The most primitive method is using the SGBM algorithm to obtain disparity, stitching the point clouds corresponding to the filtered keyframes. Although this method is very simple, its reconstruction effect is not satisfactory. Problems such as noise, ghosting, point cloud divergence, and wrong disparity assignment are all there. There are many reasons for these problems. For example, the most direct one is that the accuracy and integrity of the disparity map estimated by the SGBM algorithm are not good, and there are holes and so on. At the same time, the camera pose estimated in the visual odometry stage is not absolutely accurate, and the effect of simply splicing point clouds is that there will be a large number of redundant point clouds and ghosting.

In this case, there are several ideas to improve the effect of reconstruction. The first is to start with pose estimation to improve the accuracy of camera pose estimation, but this is not the focus of my work; the second is to improve the accuracy and completeness of disparity estimation; the third is to improve the accuracy of point clouds. Fusion; then for the second point, I used the improved AnyNet for disparity estimation and the stitching method for reconstruction. As shown in Figure 6.7 on the left 2, the noise is significantly reduced, and the divergence is also better. But redundancy and ghosting and misallocation of disparity still exist.

Then, follow the third idea mentioned above and use a more reasonable point cloud fusion method. Here I use a method based on probability filtering. As shown in left 3 in Figure 6.7, it is the result obtained by me using the SGBM algorithm + probability filtering, and it can be clearly seen that the improvement has been significantly improved. Noise, redundancies, overlaps, and misallocations are all significantly reduced. This shows that the strategy of point cloud fusion may be more important for the reconstruction model. Of course, this is the premise that there will be no huge gap in the accuracy of disparity estimation.

Combine the second and third ideas. Using the improved AnyNet and probability filtering at the same time, that is, the result obtained by the new method I proposed is shown in the right 2 of Figure 6.7. It can be seen that the noise is significantly reduced, which is that the use of AnyNet makes the results of disparity estimation more robust, so at the same time, the situation of misallocation is also improved. The redundant situation is solved by a reasonable point cloud

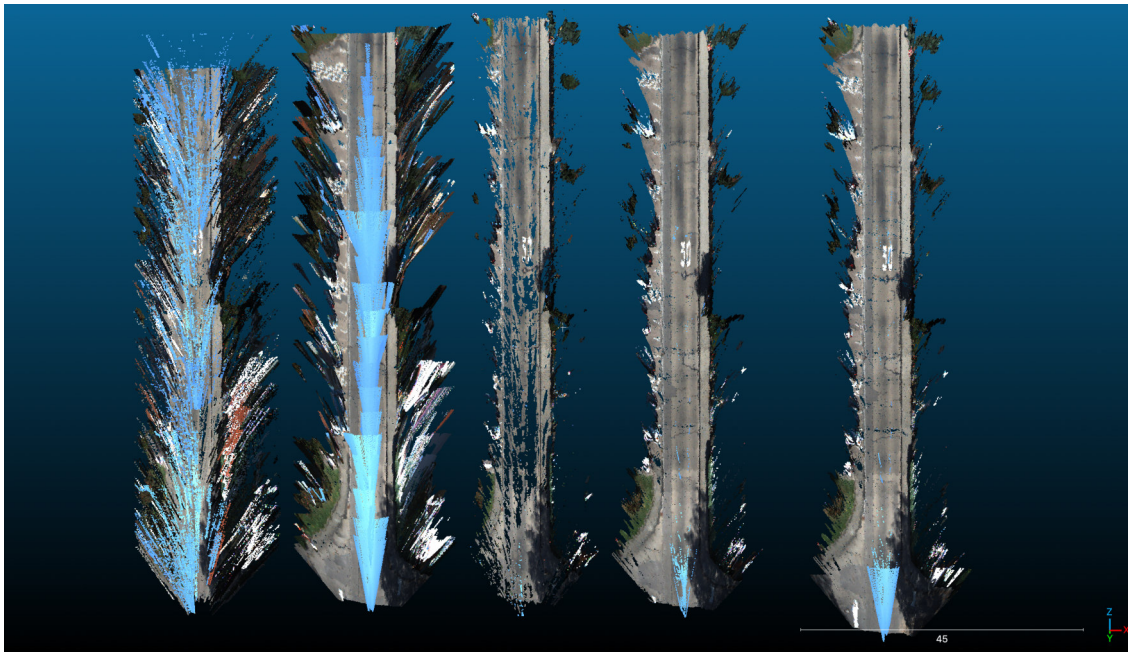


Figure 6.7: Dense reconstruction results for 5 variants. Take the first 100 stereo images from the KITTI dataset 05 sequence as input. From left to right: SGBM+concat, AnyNet+concat, SGBM+prob, AnyNet+prob, AnyNet-cva+gausian. The number in the lower right corner is the global zoom factor under the current resolution, and the larger the number, the more the scale will be reduced under the current resolution. On the contrary, it is bigger

fusion and update strategy. In conclusion, my proposed work shows excellent results in terms of the reconstruction effect.

Finally, I also propose using the CVA module to estimate the uncertainty, which will help the subsequent reconstruction. It can be seen that the result is close to the fourth result, which is also quite good, but the wrong assignment such as the sky is slightly more and the reconstructed point cloud is more complete. This can also be seen in Figure 6.9. Mainly in the beginning and end parts, the reconstructed point cloud is more complete. This shows that by combining uncertainty, the Gaussian filter fusion is a simple but effective approach.

It can be seen intuitively from Figure 6.7 that the reconstruction results without using the point cloud fusion strategy have problems of point cloud redundancy and overlap. This shows that a reasonable point cloud fusion strategy plays a vital role in building a globally consistent model. Therefore, the following comparisons mainly revolve around the latter three variables. First, I use the point cloud obtained by the SGBM+prob method as a reference, and then use the point cloud obtained by AnyNet+prob to align the point cloud, and then calculate the C2C distance between

the two point clouds. The results are shown in Figure 6.8. The upper picture is the result of SGBM, and the lower picture is the result of AnyNet. For better contrast, I panned out the SGBM results and set the color to white. The color distribution in the results of AnyNet is shown in the index on the right. The closer the color is to the upper end, the farther the absolute distance between the two point clouds is, and vice versa. However, since there is no absolutely accurate dense ground truth, it does not represent here, the redder the color, the greater the error, which can only indirectly illustrate the integrity of the reconstruction. For example, in the results of AnyNet, for the overlapping parts, the basic color is blue, indicating that the two point clouds are close, which means that the two methods have better reconstruction effects on this area, but obviously, the results of AnyNet are denser. However, for the non-overlapping parts, the color is more green or yellow, which does not mean that the reconstruction of the AnyNet method is more accurate, but it can only indicate that it is reconstructed more completely. To sum up here, under the premise of using the fusion method of probability filtering, using the AnyNet method, the reconstruction effect is more dense and complete.

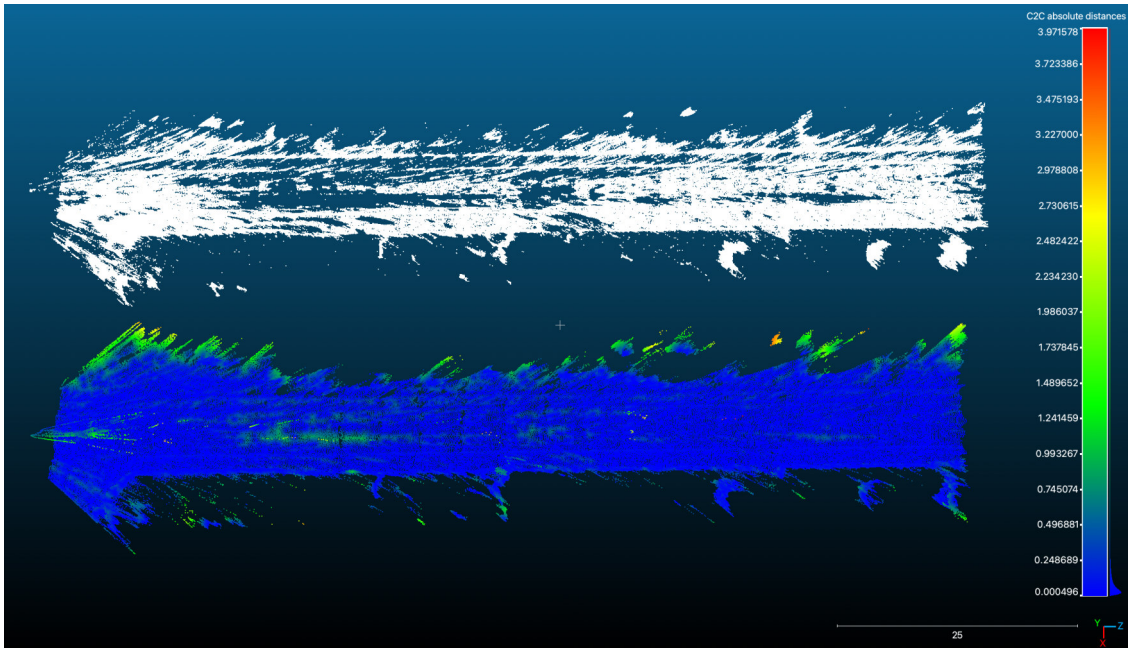



Figure 6.8: Dense reconstruction results of AnyNet+prob (bottom) and SGBM+prob (top). Using the results of SGBM+prob as a reference (white) to compare the reconstruction results of AnyNet+prob. The definition of color is modeled by the C2C (cloud to cloud) distance, as indicated on the right, the color closer to the upper end indicates the farther the absolute distance between the two point clouds, and vice versa.

Similarly, the results of AnyNet+prob and AnyNet-cva+gausian are compared, and the results are shown in Figure 6.9. The difference between these two methods is mainly in the fusion method.

The main difference between the fusion method of probabilistic filtering and the Gaussian filtering method is shown in Section 4.4. From the results of dense reconstruction, there is not much difference between the two as a whole. The point cloud obtained by the Gaussian filtering method is more complete, but it fails to remove the point cloud with misassigned disparity. However, Gaussian filtering needs uncertainty for fusion. this raises the threshold for its use, although the method of fusion itself is relatively simple. Therefore, in conclusion, there is little difference between using probability filtering and Gaussian filtering fusion to obtain reconstruction results. It mainly depends on which indicator we focus on. If we focus on integrity, then using the latter is a good choice. If accuracy is a concern, then using the former is more robust.

In my dense reconstruction work, many factors can affect its results, such as the accuracy of

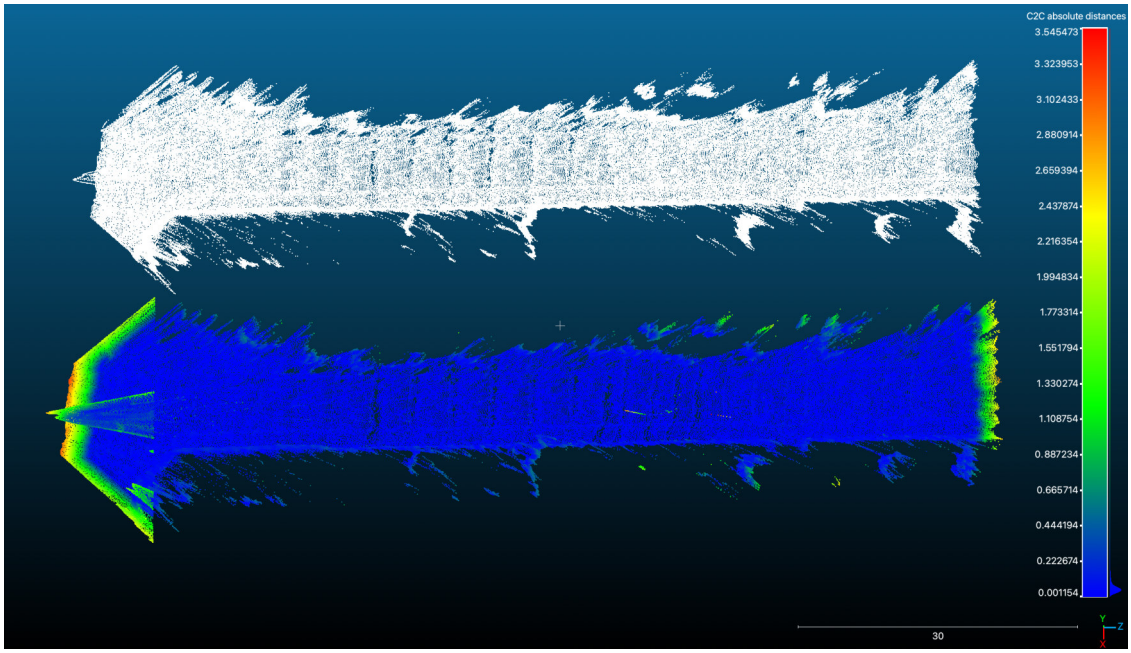



Figure 6.9: Dense reconstruction results for AnyNet+prob (top) and AnyNet-cva+gausian (bottom). Using the results of AnyNet+prob as a reference (white), compare the reconstruction results of AnyNet-cva+gausian. For the definition of colors, refer to the color definitions in Figure 6.8

camera pose, the accuracy of depth value estimation, the processing of outliers, and the online update method of redundant point clouds. Every part undoubtedly has a non-negligible impact on the final reconstruction result. The point cloud fusion strategy based on the covisible keyframes I proposed is simple and effective. When the depth value estimated by the SGBM algorithm has a lot of noise, it can also remove these outliers and obtain a smooth reconstruction model. Combined with quantized uncertainty values, Gaussian fusion also exhibits the same characteristics, simple but effective, and more complete than the former. But they all have some application restrictions, such as the former, the confidence and weight of each new point are empirically given constants.

As far as I know, there are better ways to quantify the confidence of disparity estimation, such as one of the variants of CVA. Similarly, in the Gaussian fusion method, I also assume that the depth obeys the Gaussian distribution and has nothing to do with the direction. When filtering the points, the selected uncertainty threshold is also based on experience. I think that based on this, using a variant of CVA to obtain more accurate confidence is a new work worth trying. Or a more detailed quantitative analysis of the influence of the uncertainty value on the reconstruction results.

# 7 Conclusion

In my work, I propose a new deep learning-based method for stereo SLAM. I designed a framework for loosely coupling the SLAM system and the end-to-end stereo matching network, used the network for depth estimation, and used the point cloud fusion strategy based on the probability filtering between covisible keyframes to realize the real-time dense reconstruction of outdoor scenes using only CPU.

I designed reasonable experimental evaluation for the main three functional modules: pose estimation, depth estimation, and dense reconstruction. From the experiment results of the pose estimation part, the efficiency of AnyNet in disparity estimation is competitive. Even in relying only on the CPU, the efficiency of disparity estimation is around 10hz. Compared with this, its precision performance is not so outstanding. The most obvious shortcoming is that for areas with less texture, there are more wrong disparity value assignments. I also compared the variant of AnyNet: the combination of AnyNet + CVA-Net. The uncertainty of the disparity estimation process is quantified using CVA, a separate module. Despite the increased computational cost, uncertainty provides more reliable information for subsequent work such as dense mapping. In future work, regarding this part, I think there are other options when constructing the cost body. In AnyNet, the L1 distance is used to calculate the distance between the left and right feature vectors, or it can be interpreted as a similarity. This method is simple and effective, but it lost a lot of dimensional information. A group-wise cost volume combines the advantages of a concatenation cost volume and a correlation cost volume. This will be attempted in my future research work.

The results of the pose estimation experiments show that my SLAM method using ORB-SLAM2 as the basis maintains the accuracy and robustness of its pose estimation. In the outdoor 400*400 trajectory, the average absolute pose error is only 1.4 meters. Likewise, the relative pose error is only 0.016 meters. However, in some special scenarios, obvious error fluctuations are shown. For example, in turning scenes and scenes with dynamic objects. Inspired, one of the important research points is the extraction and matching of feature points. Therefore, a more robust feature point extraction method, such as extracting feature points or matching with a deep learning network. This is also a focus of my future research work.

Finally, the conclusion drawn from the experimental results of dense reconstruction is that the

point cloud fusion strategy based on covisible keyframes and probability filtering is very obvious and effective, which can solve the problems of point cloud redundancy and noise removal, and can obtain dense reconstruction with global consistency. The main idea behind it is that points that common-view keyframes can observe will be fused and updated, which will make the points more stable or reliable. On the contrary, those outliers will either not be observed, or they will be judged unstable and eliminated. At the same time, I also used the uncertainty obtained from CVA-Net and the Gaussian filter I proposed, based on the former process, to achieve this effect as well. But there is still some work on the dense reconstruction part that I will do in my future research work. First: In probability filtering, I set the initial confidence and weight of points as constant according to experience. This is not the usual case. Using a variant of CVA-Net to estimate confidence is a viable strategy. More reasonable and accurate confidence results in a more accurate fusion. Second: When using Gaussian filtering, I limit the use of restrictions, and the modeling of depth and uncertainty is relatively ill-considered. More reasonable and comprehensive modeling of depth and uncertainty may get a better result.

# 8 Acknowledgements

After six months of hard work, I finally finished the work of my master's thesis. From the beginning of confirming the direction of the dissertation to the advancement and realization of each step, it has been accompanied by a lot of challenges and achievements. Looking back on my three years of study at the University of Leibniz Hannover, I am full of rewards despite the ups and downs. And these problem-solving methodologies and the refinement of my will will be my wealth for the rest of my life.

The completion of this master's thesis definitely depends on more than just my personal efforts. I would like to express my sincere gratitude to the following people and institution. First and foremost is my supervisor, Dr.-Ing. Max Mehltretter, is a very knowledgeable and diverse scholar. From my student project to my master's thesis, he has always been patient and friendly in giving me effective academic advice and assistance. I have learned a lot from him.

Then I would like to thank Standard Robots (Shenzhen) Co., Ltd. for giving me the opportunity to do my internship. I would like to thank my colleagues M.S. Fengchi Lv and M.S. Bing Duan. Some of the advice they gave me was also crucial.

Next, I would like to thank my parents, without their selfless, unrequited financial support and inspiring moral support, I would not have been able to complete my master's degree. Finally, I am grateful for my tenacity and steadfastness.

# Bibliography

Ahmad Fuad, N., Yusoff, A., Ismail, Z. and Majid, Z., 2018. Comparing the performance of point cloud registration methods for landslide monitoring using mobile laser scanning data. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci* 42, pp. 11–21.

Bangunharcana, A., Cho, J. W., Lee, S., Kweon, I. S., Kim, K.-S. and Kim, S., 2021. Correlate-and-excite: Real-time stereo matching via guided cost volume excitation. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 3542–3548.

Bloesch, M., Czarnowski, J., Clark, R., Leutenegger, S. and Davison, A. J., 2018. Codeslam—learning a compact, optimisable representation for dense visual slam. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2560–2568.

Chang, J.-R. and Chen, Y.-S., 2018. Pyramid stereo matching network. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5410–5418.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database. In: *2009 IEEE conference on computer vision and pattern recognition*, Ieee, pp. 248–255.

Ding, Y., Lin, L., Wang, L., Zhang, M. and Li, D., 2020. Digging into the multi-scale structure for a more refined depth map and 3d reconstruction. *Neural Computing and Applications* 32(15), pp. 11217–11228.

Eigen, D. and Fergus, R., 2015. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: *Proceedings of the IEEE international conference on computer vision*, pp. 2650–2658.

Eigen, D., Puhrsch, C. and Fergus, R., 2014. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*.

Engel, J., Schöps, T. and Cremers, D., 2014. Lsd-slam: Large-scale direct monocular slam. In: *European conference on computer vision*, Springer, pp. 834–849.

Engel, J., Sturm, J. and Cremers, D., 2013. Semi-dense visual odometry for a monocular camera. In: *Proceedings of the IEEE international conference on computer vision*, pp. 1449–1456.

Gálvez-López, D. and Tardos, J. D., 2012. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics* 28(5), pp. 1188–1197.

Geiger, A., Lenz, P. and Urtasun, R., 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In: *2012 IEEE conference on computer vision and pattern recognition*, IEEE, pp. 3354–3361.

Hirschmuller, H., 2007. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence* 30(2), pp. 328–341.

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A. et al., 2011. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In: *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 559–568.

Ji, X., Ye, X., Xu, H. and Li, H., 2018. Dense reconstruction from monocular slam with fusion of sparse map-points and cnn-inferred depth. In: *2018 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, pp. 1–6.

Keller, M., Lefloch, D., Lambers, M., Izadi, S., Weyrich, T. and Kolb, A., 2013. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In: *2013 International Conference on 3D Vision-3DV 2013*, IEEE, pp. 1–8.

Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A. and Bry, A., 2017. End-to-end learning of geometry and context for deep stereo regression. In: *Proceedings of the IEEE international conference on computer vision*, pp. 66–75.

Kingma, D. P. and Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lepetit, V., Moreno-Noguer, F. and Fua, P., 2009. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision* 81(2), pp. 155–166.

Li, B., Shen, C., Dai, Y., Van Den Hengel, A. and He, M., 2015. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1119–1127.

Liu, F., Shen, C., Lin, G. and Reid, I., 2015. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence* 38(10), pp. 2024–2039.

Liu, S., De Mello, S., Gu, J., Zhong, G., Yang, M.-H. and Kautz, J., 2017. Learning affinity via spatial propagation networks. *Advances in Neural Information Processing Systems*.

Mehltretter, M., 2022. Joint estimation of depth and its uncertainty from stereo images using bayesian deep learning. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 2, pp. 69–78.

Mehltretter, M. and Heipke, C., 2019. Cnn-based cost volume analysis as confidence measure for dense matching. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 0–0.

Mehltretter, M. and Heipke, C., 2021. Aleatoric uncertainty estimation for dense stereo matching via cnn-based cost volume analysis. *ISPRS Journal of Photogrammetry and Remote Sensing* 171, pp. 63–75.

Menze, M. and Geiger, A., 2015. Object scene flow for autonomous vehicles. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3061–3070.

Mur-Artal, R. and Tardós, J. D., 2017. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics* 33(5), pp. 1255–1262.

Mur-Artal, R., Montiel, J. M. M. and Tardos, J. D., 2015. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics* 31(5), pp. 1147–1163.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S. and Fitzgibbon, A., 2011a. Kinectfusion: Real-time dense surface mapping and tracking. In: *2011 10th IEEE international symposium on mixed and augmented reality*, IEEE, pp. 127–136.

Newcombe, R. A., Lovegrove, S. J. and Davison, A. J., 2011b. Dtam: Dense tracking and mapping in real-time. In: *2011 international conference on computer vision*, IEEE, pp. 2320–2327.

Pire, T., Fischer, T., Castro, G., De Cristóforis, P., Civera, J. and Berlles, J. J., 2017. S-ptam: Stereo parallel tracking and mapping. *Robotics and Autonomous Systems* 93, pp. 27–42.

Shaked, A. and Wolf, L., 2017. Improved stereo matching with constant highway networks and reflective confidence learning. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4641–4650.

Shamsafar, F., Woerz, S., Rahim, R. and Zell, A., 2022. Mobilestereonet: Towards lightweight deep networks for stereo matching. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2417–2426.

Tateno, K., Tombari, F., Laina, I. and Navab, N., 2017. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6243–6252.

Triggs, B., McLauchlan, P. F., Hartley, R. I. and Fitzgibbon, A. W., 1999. Bundle adjustment—a modern synthesis. In: *International workshop on vision algorithms*, Springer, pp. 298–372.

Wang, J., Liu, H., Cong, L., Xiahou, Z. and Wang, L., 2018. Cnn-monofusion: online monocular dense reconstruction using learned depth from single view. In: *2018 IEEE international symposium on mixed and augmented reality adjunct (ISMAR-Adjunct)*, IEEE, pp. 57–62.

Wang, Y., Lai, Z., Huang, G., Wang, B. H., Van Der Maaten, L., Campbell, M. and Weinberger, K. Q., 2019. Anytime stereo image depth estimation on mobile devices. In: *2019 international conference on robotics and automation (ICRA)*, IEEE, pp. 5893–5900.

Whelan, T., Kaess, M., Johannsson, H., Fallon, M., Leonard, J. J. and McDonald, J., 2015. Real-time large-scale dense rgb-d slam with volumetric fusion. *The International Journal of Robotics Research* 34(4-5), pp. 598–626.

Xu, G., Cheng, J., Guo, P. and Yang, X., 2022. Attention concatenation volume for accurate and efficient stereo matching. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12981–12990.

Ye, X., Ji, X., Sun, B., Chen, S., Wang, Z. and Li, H., 2020. Drm-slam: Towards dense reconstruction of monocular slam with scene depth fusion. *Neurocomputing* 396, pp. 76–91.

Zbontar, J. and LeCun, Y., 2015. Computing the stereo matching cost with a convolutional neural network. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1592–1599.

Zhang, Z. and Scaramuzza, D., 2018. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 7244–7251.