ACKNOWLEDGEMENTS

Many people were kind enough to help me in the process of completing my paper, including my teachers, classmates and family members.

First of all, I am grateful for the help of Prof. Yong Yue and Dr. Xiaohui Zhu. Prof. Yong Yue gave me a lot of important advice on research direction and topic selection. During the project and dissertation process, he answered many of my questions and also carefully gave me suggestions for revising my paper. Dr. Xiaohui Zhu provided me with some specific guidance during the completion of the project, and also proposed revisions to my paper in the oral presentation.

Secondly, I would like to thank Xi'an Jiaotong-Liverpool University for facilitating access to literature, etc.

Finally I would like to thank my classmates for their help in the data collection and other aspects of the experiment.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

1.1 Background

The 3D reconstruction of the real world is widely used in the fields of virtual reality (VR) (Bruno et al., 2010), augmented reality (AR) (Yang et al., 2020), cultural relic restoration (Song et al., 2018; Hu et al., 2020; Ferdani et al., 2020), environment monitoring (Haile and Rientjes, 2005), and urban planing (Heo et al., 2013). And due to advances in sensing technology and reconstruction algorithms, modeling based on scanned object information has become the focus of 3D reconstruction in recent years. Compared to conventional approaches, these methods can automatically process data through sensors and algorithms, extract features and model without the need to spend too much attention on manual data collection and inspection. Therefore, these methods approaches are more suitable for large-scale scenarios and more sophisticated modeling than conventional approaches.

As with urban planing, terrain reconstruction, there are some challenges on large scale reconstruction on the lake environment. Because large-scale reconstruction means that the collected data needs to contain large and dense amount of object feature information. 3D point cloud is one of the data types that represent the characteristics of the object. Compared with other data types, 3D point cloud has shown obvious advantages in large-scale reconstruction recently (Xu and Stilla, 2021). The point cloud is a dense collection of points with rich information of objective, such as 3D coordinates and RGB colors. Unlike other low-dimensional data, such as 2D projections, 3D spatial information in point cloud can better represent the spatial characteristics of the object as well as simplify and streamline modeling in terms of geometric meshing. At the same time, the feature of point cloud containing dense and high-dimensional target information can represent the object to the maximum extent. Therefore, this project used dense 3d point clouds to reconstruct Jinji Lake environment scene.

The basic and most important property of a point cloud is the spatial information of the target, so how to spatially detect the distance to the target object has become the focus of point cloud acquisition research. In general, point cloud acquisition can be achieved in two ways: the ranging-based and the imaging-based (Xu and Stilla, 2021). The illustration of these two methods to generate point clouds for the same scene is shown in Figure 1.1. Ranging-based methods obtain the depth of the object by computing the time between

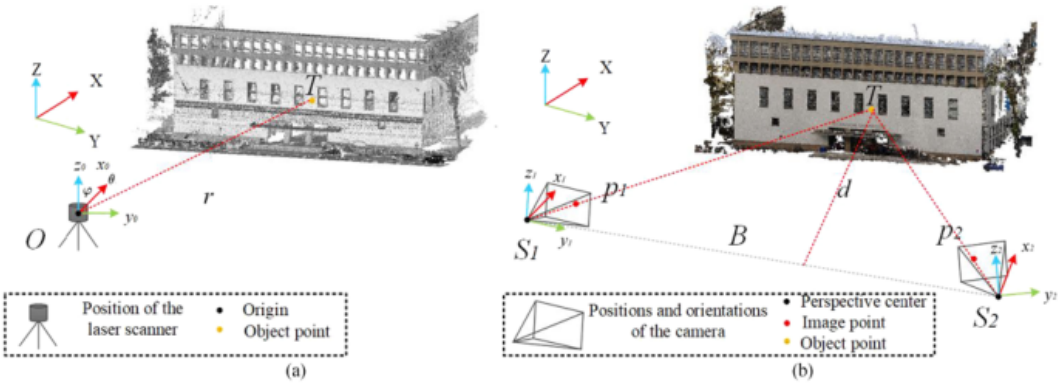

Figure 1.1: Two methods to generate point clouds: (a)Ranging-method method. (b) Imaging-method (Xu and Stilla, 2021).

the transmission and reception of the reflected signal from the sensor, also known as time of fight (TOF), and are commonly realized by light detection and ranging (LiDAR), and RGB-D cameras. RGB-D cameras, such as Microsoft Kinect (Izadi et al., 2011) and Intel RealSense, which fuse the image acquired by the camera with the depth obtained by sensors such as infrared, can acquire dense point clouds with colours in real time from scanned objects (Izadi et al., 2011). This method has led to a wide range of applications for indoor and outdoor objects such as tables, chairs and sculptures. And Python open source library Open3D (Zhou et al., 2018) supported by Intel also provides a complete pipeline from RGB-D images to dense point clouds to point cloud processing and modelling. However, the range of RGB-D cameras is between 0.25m and 10m, so the limitation of the range depth makes this approach unsuitable for large scale airborne data acquisition of Jinji Lake. In contrast, LiDAR, which sends multiple laser beams to achieve ranging through TOF, is more appropriate for large-scale airborne data acquisition. Imaging-based reconstruction method takes multi-angle photographs of the target object map and then produces dense point cloud through extracted photographic features, feature matching, calculation of camera parameters, and estimates positions, as shown in Figure 1.2. In recent years, with the rise and maturity of drone technology, the use of drones equipped with high-resolution monocular cameras to achieve point cloud generation has been widely used in large-scale reconstruction. Compared with the imaging-based method, LiDAR can directly generate dense and accurate point clouds in real time, without the need for subsequent back-end generation. However, this method requires the use of expensive sensors. In addition, the point cloud generated using only LiDAR only includes the target spatial information but not the color information. In contrast, the imaging-based approach only needs to use a cheap monocular camera. At the same time, the depth map generated at the back end can be combined with the original image to generate point cloud with RGB information.

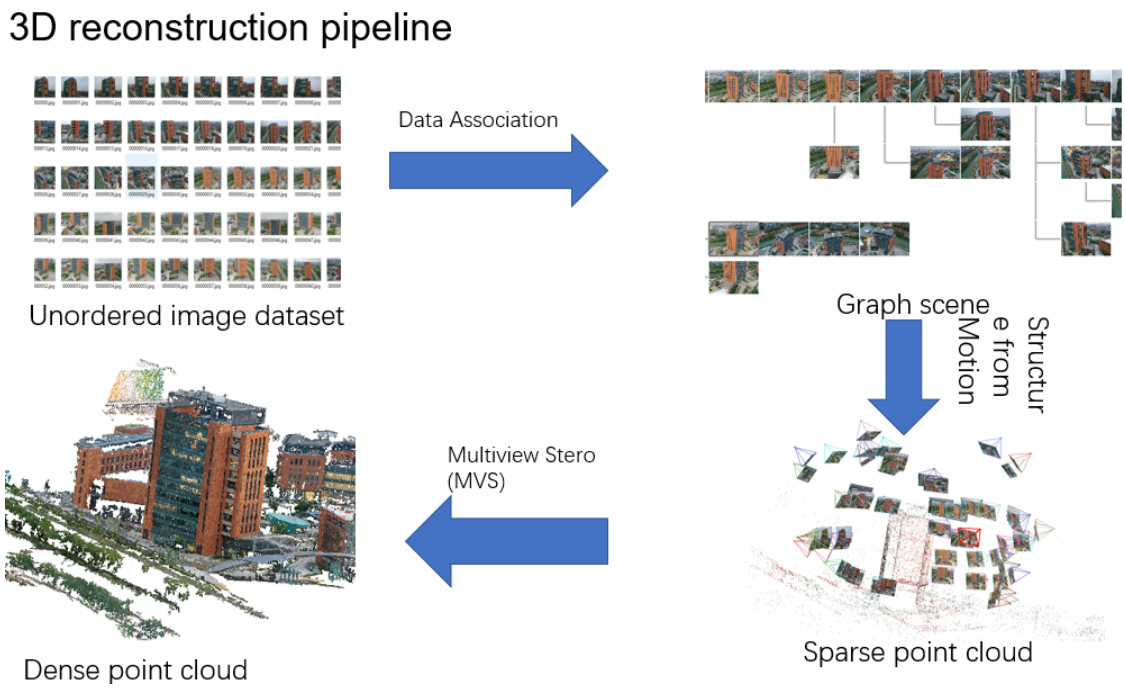

Figure 1.2: The illustration of imaging-based reconstruction.

1.2 Problem Statement and Aim

Considering the cost advantages of using imaging-based reconstruction methods and the ability to generate point clouds comparable in scale to LiDAR by algorithms for large-scale modeling, this project carried out dense 3D reconstruction based on part of the Jinji Lake scene images taken by a unmanned aerial vehicle (UAV) to design and implement a fast, exquisite large-scale 3D model.

Now, imaging-based 3D reconstruction has been widely used in fields ranging from toys to the terrain of mountains. In addition, some researchers have proposed many algorithms to optimize this method to improve the processing speed and progress. However, in the face of a large number of photos collected in large-scale reconstruction, the multi-view stereo (MVS) dense reconstruction method still requires a lot of time to process, and the untextured area of the object will cause the problem of missing point clouds. This problem is described in detail in Chapter 2 Related Work. Different from the optimization of some traditional reconstruction pipeline's algorithms, there have been breakthroughs in recent years in point cloud reconstruction using deep learning. Due to the efficient feature extraction of convolutional neural networks and the fast matrix computing speed of GPU, end-to-end depth estimation models based on deep learning have been a solution to break the traditional reconstruction bottleneck. Therefore, the man aim of this project is to how to use a leaning based approach to reconstruct a dense point cloud of part of Jinji Lake environment scene. In order to achieve this aim, the following objectives are proposed in this paper:

- To generate dense point cloud using collected aerial photographs of part of the Jinji Lake environment scene;
- To demonstrate that deep learning 3D reconstruction is also applicable to large-scale aerial photography, and that it can be modeled faster and with better quality than traditional methods;

The structure of this paper is shown below. Firstly, Chapter 1 Introduction introduces the background and motivation for the dissertation. And then, Chapter 2 summarizes some preliminaries and reviews recent research literatures related to traditional and learning-based reconstruction methods of dense point clouds. The design of the reconstruction of the Lake Jinji Lake environment scene is described in Chapter 3. The next chapter will show the specific implementation and experiment results. Finally, the discussion and the conclusion will be written in chapter 5.

CHAPTER 2. RELATED WORK

This chapter presents some basic algorithms for 3D reconstruction, as well as an overview of recent point cloud reconstruction methods. Therefore, this section is divided into five sections. The first four sections provide an introduction and literature view of camera calibration, structure from motion (SfM), traditional and learning-based reconstruction approaches, respectively. The final section summarizes the chapter and provides a risk analysis of the project based on the above review.

2.1 Camera parameters and calibrations

In general, 3D reconstruction is an important part of 3D computer vision. As with other 3D computer vision methods, the first step in 3D reconstruction is camera calibration, i.e. obtaining camera parameters. And camera parameter is a necessary condition for both traditional and learning-based reconstruction methods. The image is essentially the projection of a real object in the world frame coordinate system, rotated and translated in the camera's image plane (2D coordinates of the image). The rotation of coordinates can be referred to as Row, Pitch and Yaw, representing the rotation of the x-axis, y-axis and z-axis respectively (Janota et al., 2015). These three rotations: Row, Pitch, Yaw can be represented by the following equations. α, β and γ represent the angle of rotation of the three coordinates respectively, and $\hat{X}$ represents the coordinate after rotating.

$$R_x(\alpha) = R_3^2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \Rightarrow \hat{X} = R_3^2 X \tag{2.1}$$

$$R_y(\beta) = R_2^1 = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \Rightarrow \hat{X} = R_2^1 X \tag{2.2}$$

$$R_0(\gamma) = R_1^0 = \begin{bmatrix} \cos\gamma & -\sin y & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \hat{X} = R^0 X \tag{2.3}$$

An illustration of these three rotations is shown in Figure 2.1.

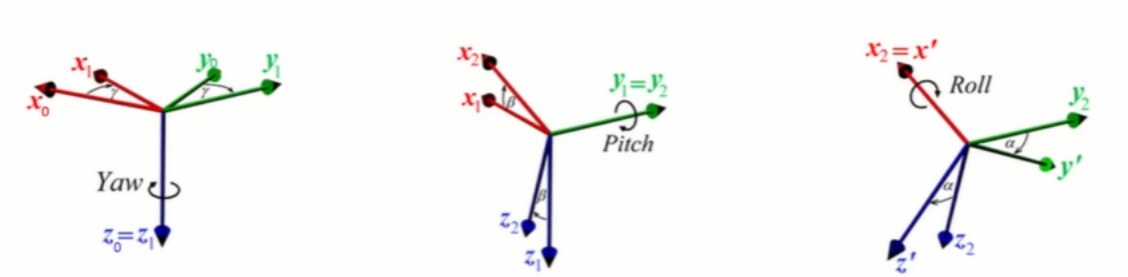

Figure 2.1: The illustration of coordinate rotation (Janota et al., 2015).

Similar to the transformation of a matrix which can be decomposed into the multiplication of multiple matrices, all coordinate rotations can be decomposed into Yaw, Pitch and Roll, so the camera's rotation matrix R can be decomposed into the following equation.

$$R_3^0 = R_1^0 R_2^1 R_3^2 = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{2.4}$$

Coordinates in 3 dimensions are not only rotated but also translated, which means the distance from the origin of the spatial coordinates to the camera centre. Thus the combination of rotation and movement of coordinates can be expressed $\hat{X}$=RX + T. And then extrinsic parameters of the camera, which combine coordinate rotation and movement, can be represented the matrix:

$$\left[\begin{array}{c|c} \mathrm{R} & \mathbf{t} \end{array}\right] \tag{2.5}$$

The points X: k[x,y,z,1] in the world frame coordinate can be unified under the camera coordinate system after multiplying by the 3*4 external reference matrix, as shown in Figure 2.2. The Z-axis of this coordinate crosses the centre of the camera and the image plane. f is the distance from the centre of the camera to the image plane, i.e. focal length. The projection of the object on the camera can be found according to the law of similar triangles. Suppose the coordinates of one is [X,Y,Z] and the coordinate of its projection is [x,y], then Z/f=X/x, so x=(f X)/Z. Similarly, we can find y, y=(f Y)/Z. Therefore, the projection transformation

can be expressed as follows:

$$(\mathrm{X},\mathrm{Y},\mathrm{Z})^{\top} \mapsto (f\mathrm{X}/\mathrm{z}, f\mathrm{Y}/\mathrm{z})^{\top}, i.e. R^3 \mapsto R^2 \tag{2.6}$$

This transformation whic is also called camera intrinsic parameters can be represented by the following matrix.

$$\begin{bmatrix} f & & 0 \\ & f & 0 \\ & & 1 \quad 0 \end{bmatrix} \tag{2.7}$$

Frequently, the centre of the camera and the image original point are not simultaneously on the z-axis, while the focal lengths of the camera on the x- and y-axes are not exactly the same, so the intrinsic parameter K of the camera is often expressed as:

$$\begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \tag{2.8}$$

The product of the intrinsic and extrinsic camera parameters is therefore referred to as the camera parameter p:

$$\mathrm{P} = \mathrm{K} \begin{bmatrix} R & t \end{bmatrix} \tag{2.9}$$

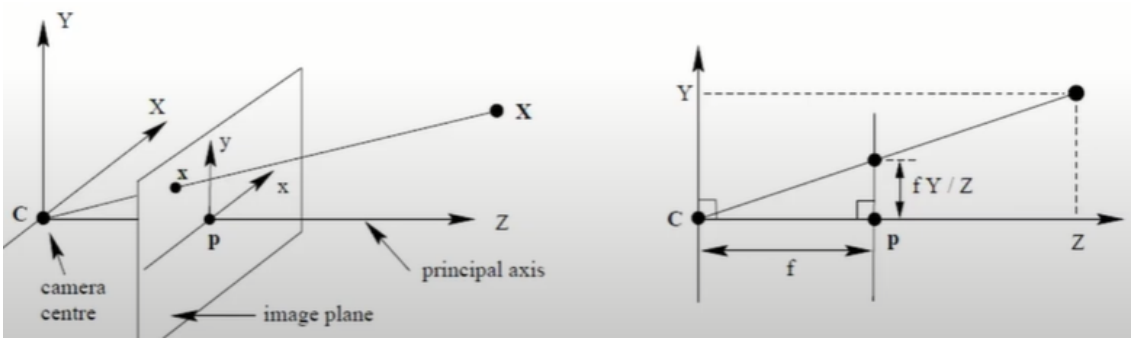

Figure 2.2: The illustration of camera model.

How to obtain a solution for the camera parameter P is the so-called camera calibration problem. Zhang (2000) proposed a camera calibration has been widely used in 3D vision. This algorithm deploys the board, which is shown in Figure 2.3, on the target and sets the world coordinate system on the grid . Since the values of the upper grid points of the board are known, the camera parameters can be computed. This method has the advantages of anti-distortion and robustness, but it is difficult to deploy a calibrated board over large-scale and long-range environments. Therefore, it is easier to implement in SfM to inverse solve

the base matrix based on two matched images to obtain the camera parameters. This method is presented in the next section.

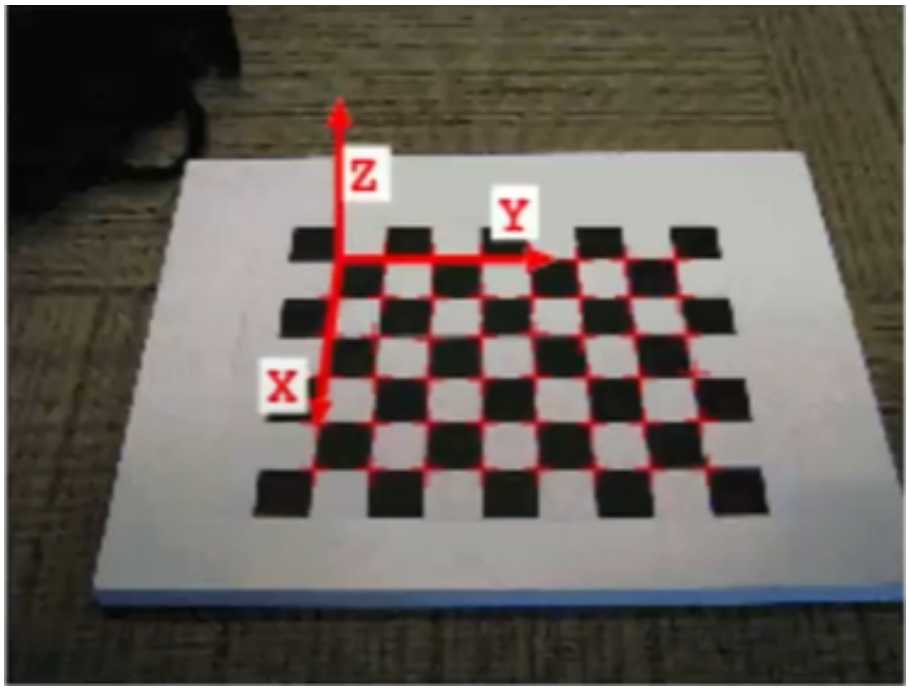

Figure 2.3: The camera calibration board.

2.2 SfM

The core task of SfM is to restore the structure of the object from unordered images taken by the camera's motion. In general, the steps of SfM, which is shown in Figure 2.4, can be divided into feature extraction and matching, data association, image matching and geometric constraints, structure reduction and estimation of camera parameters.

Before feature matching, the most key step is feature extraction for each image. In contrast to corner detection (Harris et al., 1988) and principle component analysis (Yang and Yang, 2002), the image feature detection algorithm proposed by Lowe (1999) has good robustness in image rotation and zooming. Hence, this algorithm is called scale-invariant feature transform (SIFT). Given the advantages of this algorithm, it is widely used in SfM as a way to extract image features. The key idea of SIFT extracting features is to search for image positions on all scale spaces and identify potential points of interest with scale and rotation invariance by means of a Gaussian differential function (DOG). After calculating the SIFT descriptors for each image, the matching of eigenvalues can be achieved by calculating the Euclidean shortest distance between the descriptors of the two images. In addition, the use of K-Nearest Neighbor (KNN) algorithm (Setyawan et al., 2015) and GPU accelerating (Wu, 2011) improved the speed of features matching.

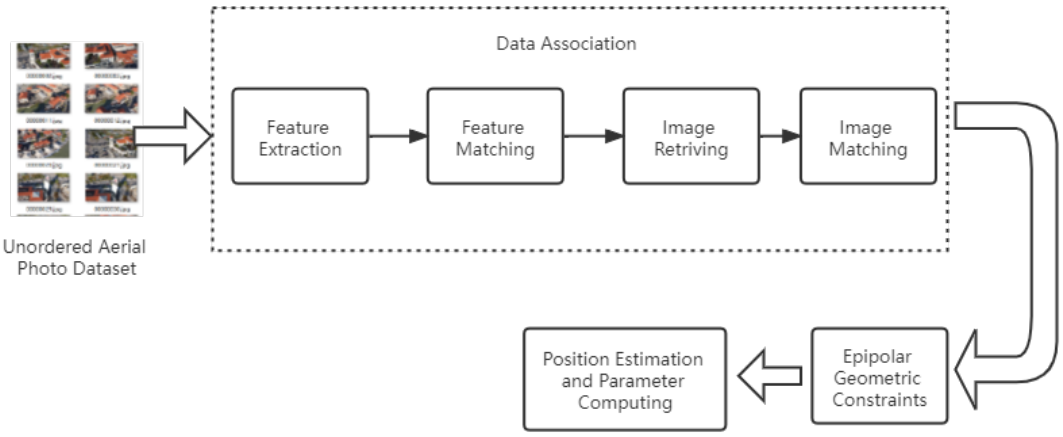

Figure 2.4: The principle of Structure from Motion.

As shown in Figure 2.4, after all of the images have been matched, the matched image pairs are available for Fundamental Matrix (F) computing, and decomposition of the camera parameters. This approach is based on epipolar geometric constraints, which is shown in Figure 2.5. For two matched images, a point x on a image called reference image with the corresponding camera centres of the two images can form a plane, called an epipolar plane, π. The intersection of π and the image plane is called the epipolar line, l. Figure 2.5 depicts an epipolar constraint relationship whereby a point x on a reference image can correspond to an epipolar line on a matching photograph, as follows:

$$\mathrm{l}' = \left[\mathrm{e}'\right]_{\times} \mathrm{H}_{\pi}\mathrm{x} = \mathrm{F}\mathrm{x} \tag{2.10}$$

H represents the relationship from x to the corresponding matching point x'. The correspondence between x and the corresponding epipolar line is represented by the fundamental matrix F. Since the corresponding points of the reference image are present on the epipolar lines, the matched epipolar line -l' can be computed based on the feature points x and x' on the two matched images. Therefore, F of the matched photo pairs can be derived by knowing x and x', which relationship is shown in Figure 2.6. And F is determined by the camera parameters and its inverse solution formula is shown below:

$$\mathrm{F} = \left[\mathbf{e}'\right]_{\times} \mathrm{K}'\mathrm{R}\mathrm{K}^{-1} = \mathrm{K}'^{-\mathrm{T}}[\mathbf{t}]_{\times}\mathrm{R}\mathrm{K}^{-1} = \mathrm{K}'^{-\mathrm{T}}\mathrm{R}\left[\mathrm{R}^{\top}\mathbf{t}\right]_{\times}\mathrm{K}^{-1} \tag{2.11}$$

Based on F decomposition formula, the 8-point algorithm (Hartley, 1997) can be used to find the camera parameters for each image.

After the above SfM steps, the two matched images can be used to generate the sparse point cloud of feature points based on epipolar geometric triangulation, while the matching relationship and camera parameters are provided to the next step of dense point cloud reconstruction.

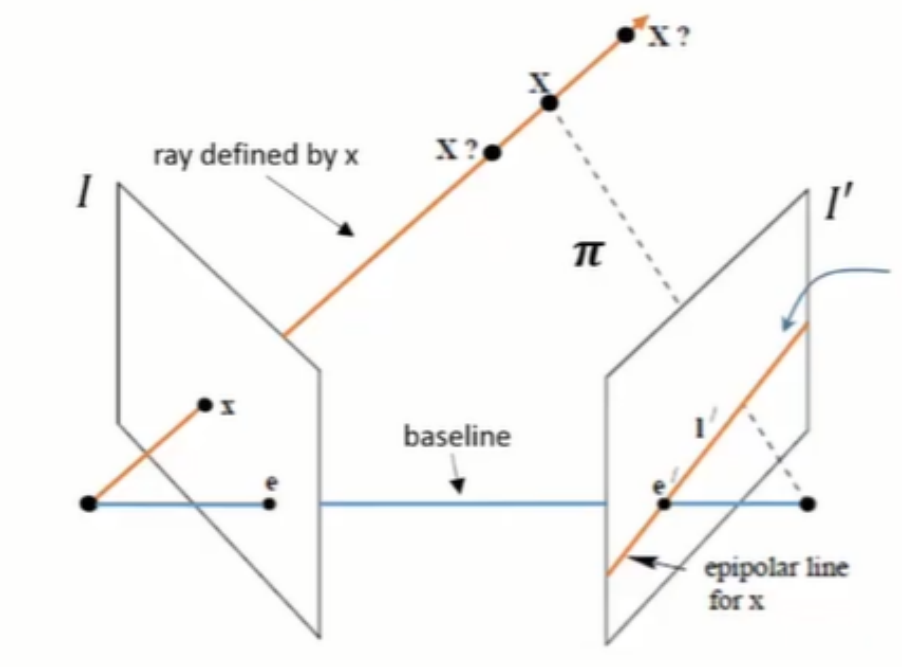

Figure 2.5: The illustration of epipolar geometric constraints.

Inspired by these SfM steps, researchers have studied and optimised an increasing number of SfM algorithms for large-scale reconstruction of thousands of images (Agarwal et al., 2011). Nister and Stewenius (2006) proposed an algorithm for tree retrieval of all SIFT feature points, which reduces the time complexity of feature matching. The three most widely used large-scale SfM strategies include incremental (Wu, 2013; Snavely et al., 2006), hierarchical (Gherardi et al., 2010) and global (Crandall et al., 2011). Incremental SfM performs better in terms of reconstruction accuracy and robustness than the other two strategies (Schonberger and Frahm, 2016), and is therefore more widely used. However, as the number of cameras increases, incremental SfM processes photos more and more slowly, due to its time complexity of O(n^4) (Wu, 2013). Therefore, Wu (2013) proposed a new incremntal SfM, optimized in four steps: feature matching, bundle adjustment methods, adding sub-steps, re-triangulation. Furthermore, Schonberger and Frahm (2016) proposed a geometric verification strategy, triangulation and outlier filtering to further improve the robustness and reconstruction quality of incremental SfM. In addition, Agarwal et al. (2011) performed SfM reconstruction on over 10w photographs taken in Rome, which proved the usability of the SfM strategy for large-scale reconstruction. Therefore, based on the significant advantages of this method in terms of reconstruction speed, robustness and quality, my project used incremental SfM to obtain the image pairing, camera parameters and sparse point clouds of the Jinji Lake environment scene as the first step of dense reconstruction.

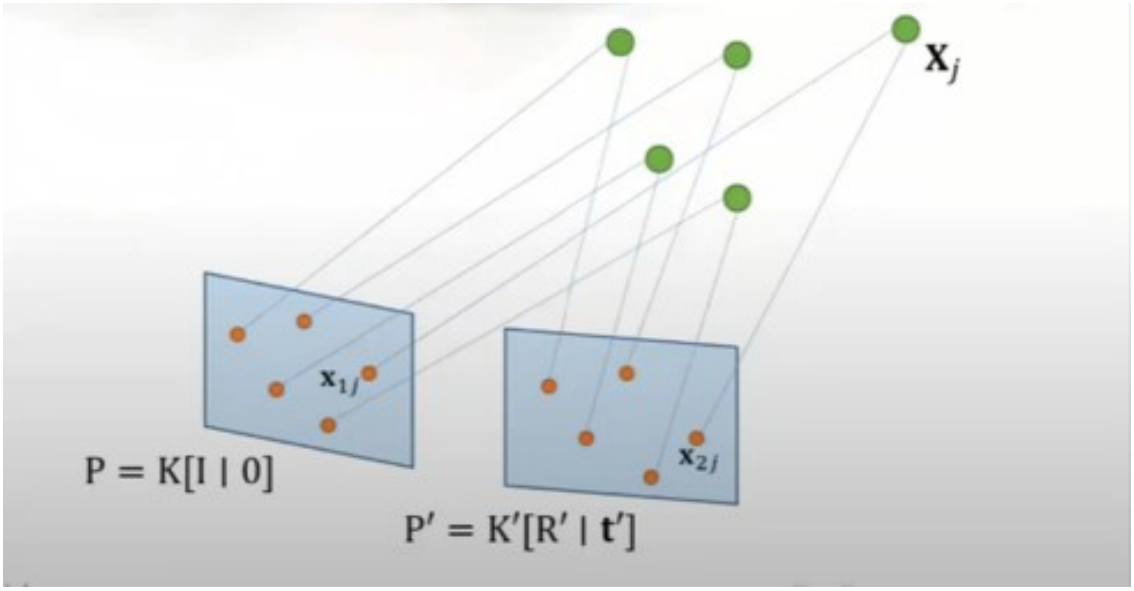

Figure 2.6: SfM triangulation.

2.3 Traditional dense point cloud reconstruction

The dense point cloud reconstruction is based on several images from different perspectives, hence the method for this is called multi-view stereo (MVS). Different from SfM, where the inputs only need images, MVS also requires the input of camera parameters and image matching relationships from SfM. This is because the underlying implementation of MVS still relies on the epipolar geometric constraints. Since product of the point existing on the line and the line is 0, the corresponding pixel point in the matched image and F have the following relationship:

$$x_2 F x_1 = 0 \tag{2.12}$$

Based on this relationship, a image can be used as a reference image. And the epipolar line l', i.e. the line corresponding to the pixel point x', can be found on the matching photograph at a pixel point, which is shown in Figure 2.7.

The matched image will look for the closest point on l' to the reference image until the two camera centres are parallel to the rays connecting the pixel points, which means that the point is out of camera's view range or obscured. However, this method will fail if the image's texture is not obvious or there is a large number of identical pixels on the epipolar line. Therefore,it is more common in MVS to match multiple pixels in a matrix, which is called a patch(Bleyer et al., 2011). This method shows better robustness by selecting other points around the point to form a patch for matching. And zero-mean normalized cross correlation (NCC) implements quantification of the cost between two patches. The process of patch match is shown in Figure 2.8. Furthermore, PMVS algorithm proposed by Furukawa et al. (2010*a*) improves the quality of the reconstruction by optimizing the selection of patches and feature extraction. And then Furukawa

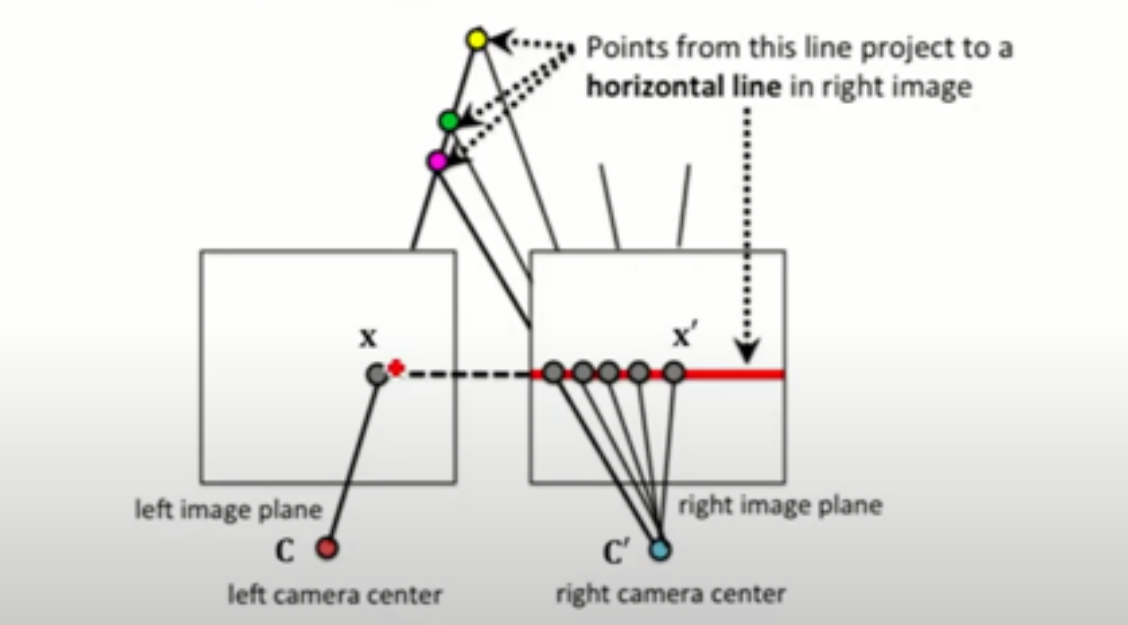

Figure 2.7: Two-view Stereo.

et al. (2010*b*) proposed CMVS algorithm, which is based on his PMVS, to improve the speed of dense reconstruction on large sets of unordered images. CMVS clusters the images based on a point cloud of sparse feature points from the SfM input, ensuring that each cluster contains at least one feature point in the patch.

MVS can be divided into 3 types, depending on the form of output presentation: voxel based methods (Kar et al., 2017), depth map based methods (Tola et al., 2012) and feature point based methods (Furukawa et al., 2010*b*). (Yao et al., 2018). The depth map-based method allows for simple fusion of point clouds as well as other output transformations because of the advantages of scalability and portability, citep (Yao et al., 2018). Therefore, based on recent MVS benchmarks (Knapitsch et al., 2017), almost all of the best MVS algorithms for reconstruction are almost always based on depth map methods (Galliani et al., 2015*a*).

2.4 Learned reconstruction

Although traditional MVS has made great strides in terms of speed and quality of reconstruction, and is also widely used in areas of research and application. However, with huge amount of images, traditional methods need take over tens of hours to reconstruct. And texture, brightness and other factors also affect the quality of the reconstruction. Thus there is still much to be done to improve dense point cloud reconstruction. Recently, with the excellent performance of deep learning in areas of computer vision, such as classification and target detection, many researchers have tried to design deep learning algorithms to optimize 3D point cloud reconstruction. As mentioned in the previous section, the quality of the traditional MVS match is influenced by some factors such as the material of the photo, so the researcher focused on how to optimize

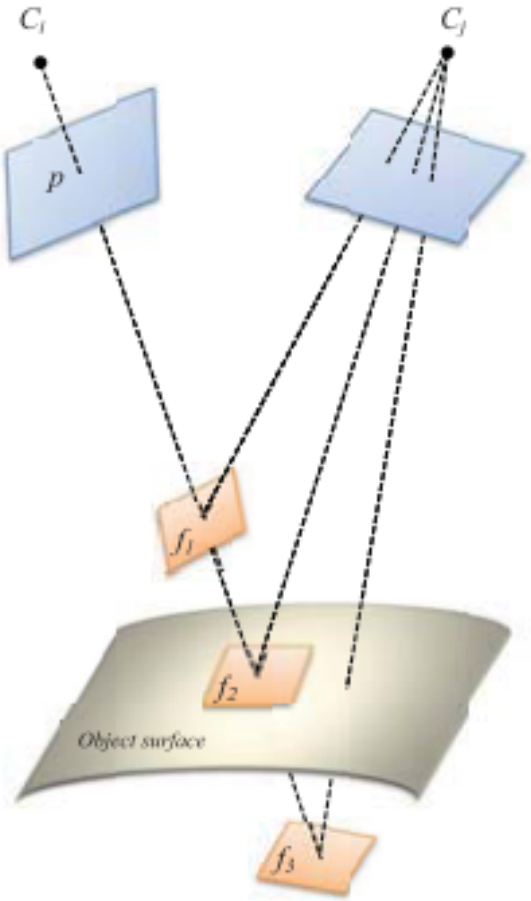

Figure 2.8: The illustration of patch-match stereo.

the match. Matchnet algorithm proposed by Han et al. (2015) first used deep learning to match images' pixel patch. Furthermore, Zbontar et al. (2016) quantified the matching cost by convolutional neural network (CNN) to obtain features of the dataset.

The end-to-end deep learning approach above, called learned stereo, has significant advantages over traditional methods for stereo matching. However, these methods do not address how to deploy on MVS. Multi-patch similarity Hartmann et al. (2017) can be used as a deep learning based cost metric for replacing cost measurement functions such as NCC in MVS, and has the advantage of being faster and better at matching. Meanwhile, after extracting the image features and wrap them into 3D space, SurfaceNet proposed by Ji et al. (2017) used a 3D convolutional social neural network to regularize the cost. These learned MVS algorithms, although they perform well on reconstructions, are only suitable for small-scale reconstructions due to their volumetric-based representation. The MVSNet proposed by Yao et al. (2018) algorithm allows learned MVS to be used for large-scale 3D reconstruction. The innovation of this algorithm is to encode the camera parameters for a large number of features, thus constructing a cost volume that can be regularised by 3d convolution. The structure of MVSNet is shown in Figure 2.9. The method is hundreds

of times faster than traditional MVS methods such as OpenMVS (Cernea, n.d.) and Colmap (Schonberger and Frahm, 2016) in DTU benchmarks (Jensen et al., 2014), and the quality of the reconstruction is higher in areas of weak material on the scene. And then, many improvements to the MVSNet algorithm, such as R-MVSNet(Yao et al., 2019) using Reccurent regularisation instead of 3D convolution, have greatly optimised the high GPU memory consumption of MVSNet. However, since the 3D convolution approach is better for feature extraction and estimating cost volume, the focus of many researchers has still been on 3D CNN regularization. And MVSNet trained on the DTU dataset still works well on other datasets.

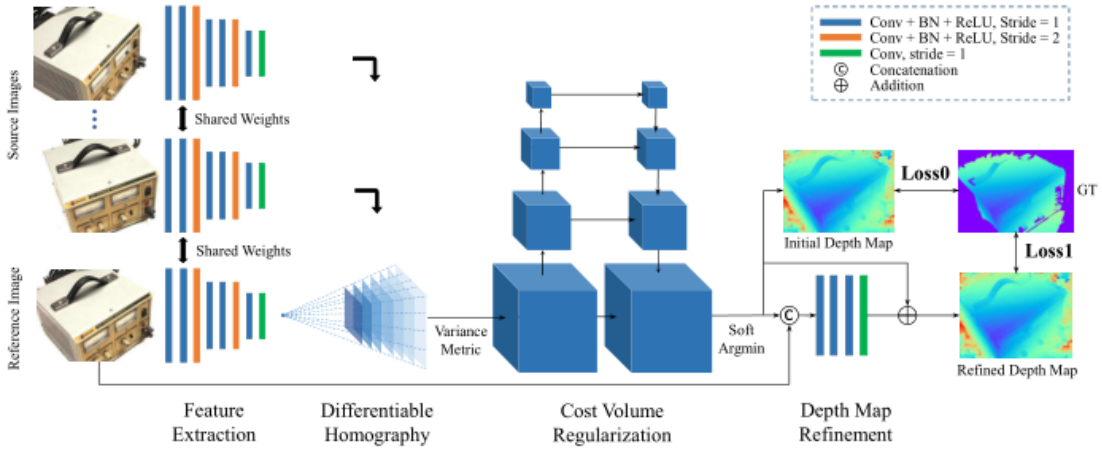

Figure 2.9: The structure of MVSNet proposed by Yao et al. (2018).

2.5 Summary

As traditional methods have problems in terms of reconstruction speed and quality of reconstruction in areas where features are not obvious, this paper adopts Learned MVS for 3D reconstruction. since MVSNet has good reconstruction results on other types of datasets without additional training, this demonstrates the broad applicability of the learning-based reconstruction approach to different targets. Therefore, this paper reconstructed Jinji Lake environment scene based on deep learning.

CHAPTER 3. METHODOLGY

This chapter introduces the main methodolgy of point cloud dense reconstruction of Jinji Lake environment scene. Given the good performance of learning-based MVS in many reconstruction objects, this paper designed a complete deep learning based 3D point cloud reconstruction pipeline to reconstruct the part view of Jinji Lake environment scene. And the framework of this pipeline is shown in Figure 3.1. As shown in the figure, the method still uses the incremental SfM algorithm to achieve the solution of the camera intrinsic parameter K and the extrinsic parameters [R|T]. And then, in the dense point cloud reconstruction part, the obtained images, camera parameters and camera pairs are fed into a trained neural network model to obtain the depth map. Therefore, this chapter is divided into two sections: data processing and point cloud dense reconstruction.

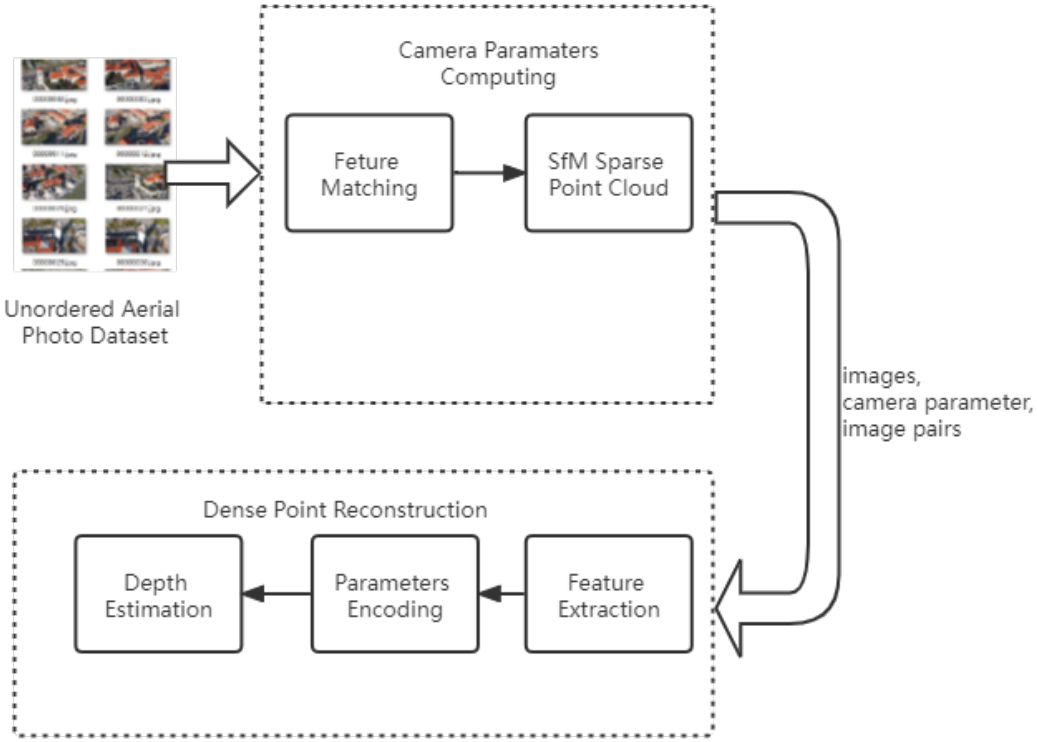

Figure 3.1: The framework of Jinji Lake enviroment scene reconstruction algorithm.

3.1 Data processing

The work of this section is to prepare data for dense point cloud modeling, that is, to create a test data set for the neural network model that contains three parts: pictures, camera parameters, and camera pairs. Whether it is traditional or learning-based MVS, the acquisition methods of these three data sets are based on the SfM algorithm. Therefore, this paper used optimized incremental SfM for data processing and parameter extraction.

As shown in Figure 2.4, the first step of the SfM algorithm is to find SIFT feature points for each picture, which steps are as follows:

- Step1: Scale-space extremum detection: search for image positions on all scales. Construct Gaussian difference (DoG) pyramids to identify potential points of interest that are invariant to scale and rotation by means of Gaussian differential functions;
- Step2: Key point location: at each candidate location, a fine model is fitted to determine the location and scale;
- Step3: Direction determination: Based on the local gradient direction of the image, assign directions to each key point location;
- Step4: Key point descriptions: Measures the local gradient of an image at a selected scale in the immediate vicinity of each key point.

The core of the feature point is to scale the image into four layers, and each layer performs five-layer Gaussian convolution, the formula is as follows:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-m/2)^2+(y-n/2)^2}{2\sigma^2}} \tag{3.1}$$

Based on the properties of the Gaussian function, the use of Gaussian convolution enables the extraction of the properties of the region. The images after each layer of convolution are differenced to generate a new layer with 4 differential images, this is called DoG. DoG is performed for each layer to form a DOG pyramid, which is shown in Figure 3.2. After searching out feature points for each image, feature matching can be performed between the two images. The feature points find the two points closest to the Euclidean distance between the photo to be matched and if the ratio of the two points is less than 0.8, the match is successful (Lowe, 1999). Typically, there are some matching errors after matching, so additional matching checks are needed to increase the robustness of the results. Random Sample Consensus (RANSAC) (Shi et al., 2013) is an efficient detection method for feature point matching. The steps of this algorithm in SIFT are shown below:

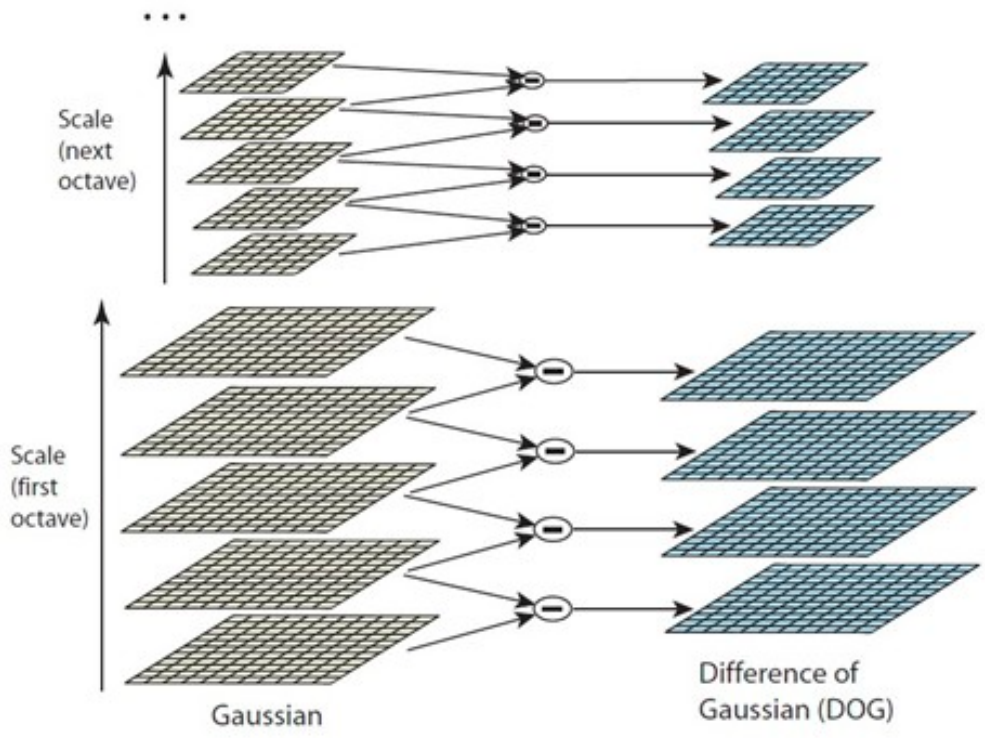

Figure 3.2: Generation of DoG pyramids.

- Step1: To select 4 pairs randomly from the matched pairs;
- Step2: To calculate the transformation matrix M based on these 4 matching point pairs;
- Step3: To use this calculated matrix to test all other outliers, dividing all outliers into two parts by a threshold value;
- Step4: Randomly select four points that satisfy the matrix, and return to step 2 until the transformation matrix no longer changes.

The form of the SIFT feature matching after RANSAC filtering is shown in Figure 3.3.

As mentioned in Chapter 2, Wu (2011) implemented the GPU to compute SIFT effectively increases the speed. However, even with the improved speed of feature matching, simply matching each image has a time complexity of O(N .K^2), with N representing the number of images and K representing the number of features to be matched per image. Suppose the unordered dataset contains 10,000 images, i.e. N=1000, and each image has 1000 feature points, i.e. K=1000. Then the number of comparisons required for feature matching is 1,000,000,000. Such a large time complexity is unacceptable no matter how fast each match is, not to mention that having more feature points on a high resolution photo will make the number of matches exponentially explode. Therefore, it is necessary to use a tree data structure for image feature retrieval, and then build a bag of words reduces the time complexity to O(K .b .L) (Nister and Stewenius, 2006).

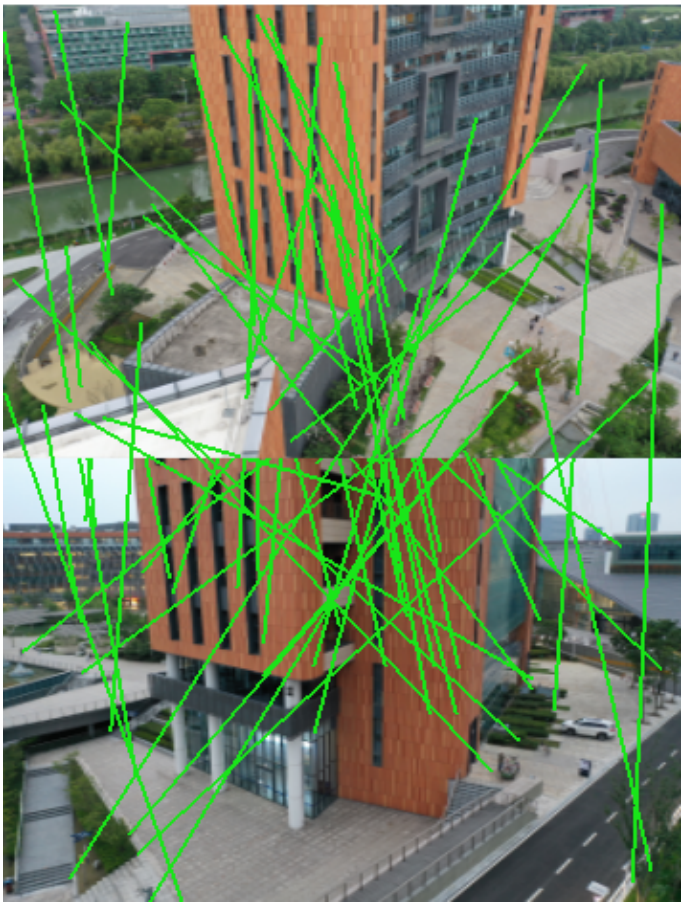

Figure 3.3: SIFT feature matching of one building's images at XJTLU.

This image retrieval algorithm, as shown in Figure 2.5, is divided into the following four steps:

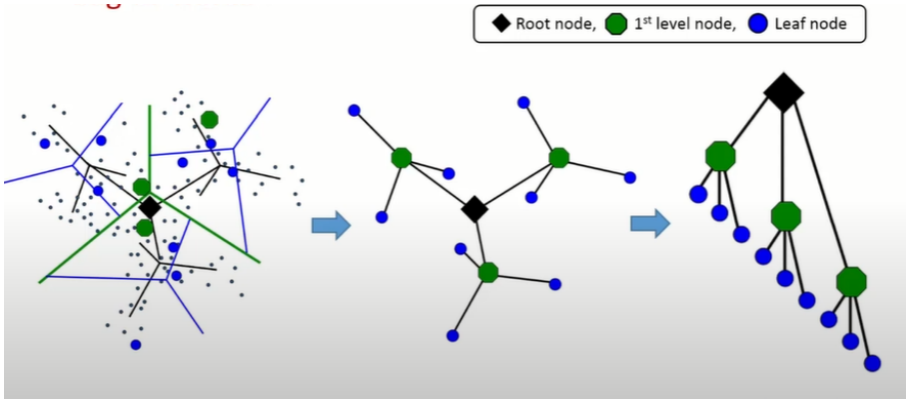

Figure 3.4: SIFT feature point K-means trees, where K is 3 (Nister and Stewenius, 2006).

- Step1: Compute SIFT feature points and 128-dimensional descriptors for all images at once, cluster all feature points using the K-means algorithm, and generate a hierarchical index tree;
- Step2: For the single feature point of each image, select the image with the nearest feature point in the constructed K-means index tree for matching;

- Step3: For each image feature match, set the priority value for the matched photo, which is 0 initially. For each image, the priority is counted by one for each match;
- Step4: When all feature points of a photo have been matched, choose the other photos with the highest priority value as matched image.

Incremental SfM selects a number of images to be used as the reference, depending on the thread chosen. Afterwards, incremental image matching is performed according to the feature retrieval method described above, and camera parameters are calculated and feature sparse point clouds are generated, as shown in Figure 3.5.

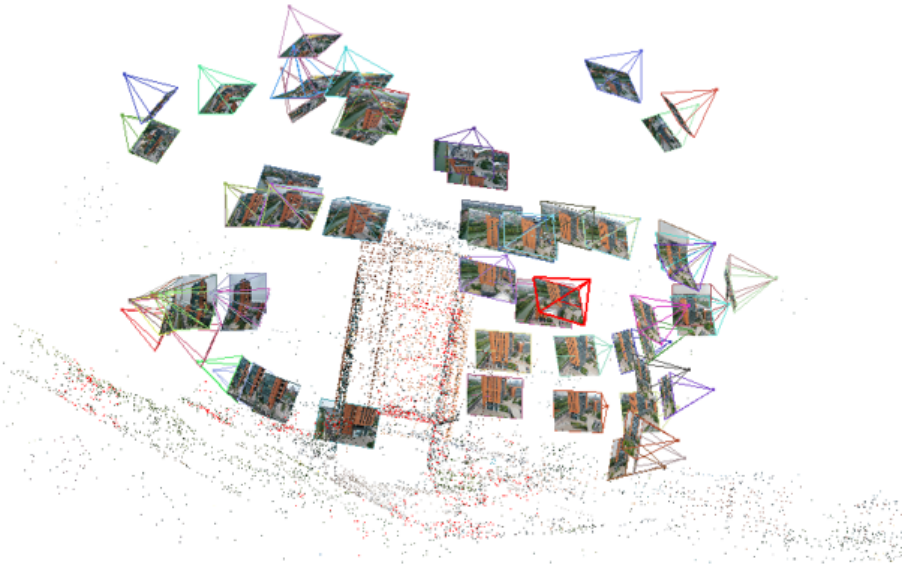

Figure 3.5: The incremental SfM result from the building's images at XJTLU.

3.2 Dense point cloud reconstruction

This Paper designed an end-to-end learning-based MVS neural network model. The working process of the model, which draws on the traditional MVS workflow, includes 4 parts: feature extraction, cost volume regularization, depth estimation and construction of Loss function. And the workflow is shown in Figure 3.6. As mentioned in the previous section, the output is flexible based on the depth map reconstruction, which is allowed for a very simple conversion to point clouds. Therefore, the point cloud reconstruction problem can be converted into a depth estimation problem, which facilitates the construction of the cost function in the neural network model.

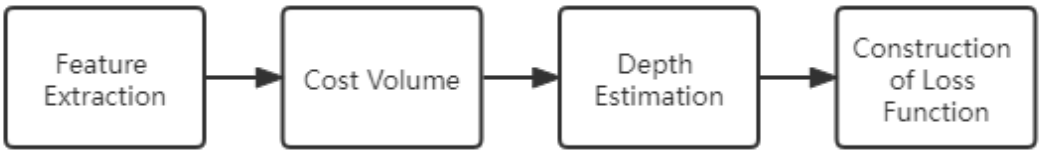

Figure 3.6: The workflow of the learning-based MVS network.

3.2.1 Image feature extraction

As with other image classification algorithms, here we also use a 2D CNNs for feature extraction. In addition, different from other convolution methods, in order to implement MVS, multiple images must be input simultaneously for parallel convolution. Thus the feature extraction neural network is a CNN with N images as input and N feature values as output, and its structure is shown in the Figure 3.7. In order to design a deeper CNN to better extract image features, this paper divides the feature extraction network into a convolutional tower with three parts, where the third and sixth layer have the stride of 2 to achieve the layering of the convolutional tower. Each layer of the CNN follows a Batch-normalization (BN) and uses ReLU as the activation function to output.

3.2.2 Cost volume regularization

After the feature extraction network, each image is transformed into a extracted feature map with 32 channels reduced in size by a quarter. In traditional MVS, the depth is calculated by constructing similar triangles based on epipolar geometry after the homography variation. The first step in the cost volume layer is therefore to unify all the feature maps into different fronto-parallel planes of the reference image after the planer homograhy transformation. This step is done by picking a random image as the reference image and then having each image multiplied by a 3*3 homograpy matrix. This matrix is represented as follows, N1 represents the principal axis of the reference image.

$$\mathbf{H}_i(d) = \mathbf{K}_i \cdot \mathbf{R}_i \cdot \left(\mathbf{I} - \frac{(\mathbf{t}_1 - \mathbf{t}_i) \cdot \mathbf{n}_1^T}{d} \right) \cdot \mathbf{R}_1^T \cdot \mathbf{K}_1^T \tag{3.2}$$

The importance of this step is that the conversion of feature maps from 2D to 3D is achieved by introducing depth and encoding camera parameters.

And then, after the planer homography transformation has been completed, the MVS approach is to implement a depth solution based on the disparity. Therefore in this step, the cost volume network needs to estimated the depth. The core of this step is the conversion of the multi-layer feature map into a gap value, called cost volume (C). Due to the variable number of input images to MVS, ranging from tens to hundreds

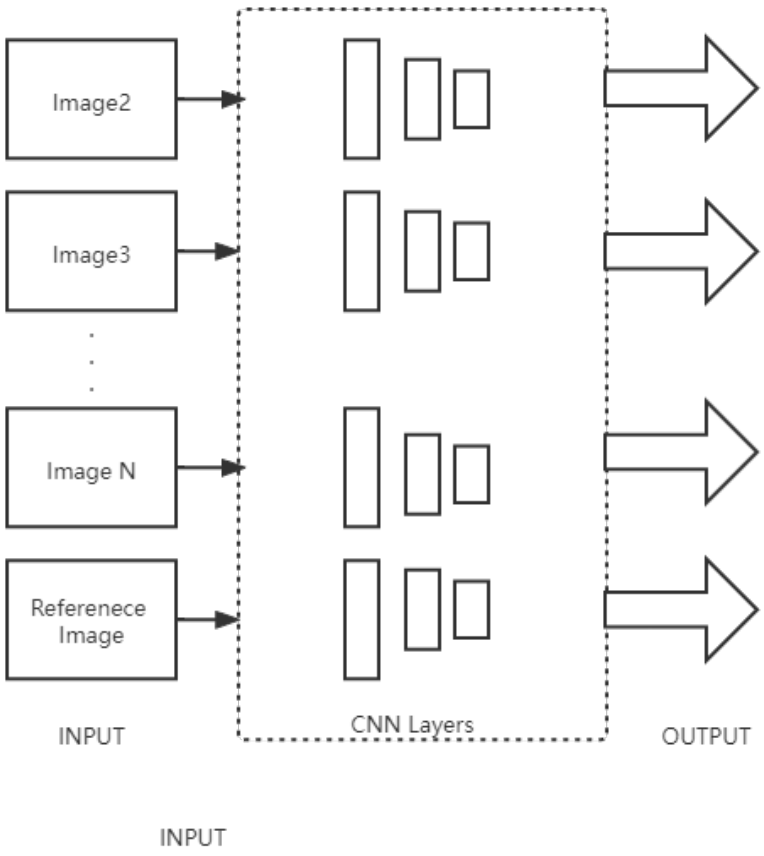

Figure 3.7: The framework of image feature extracting network.

of thousands, aggregation cannot be achieved by simple subtraction or Euclidean distance. Therefore, a variance-based approach is used here to measure:

$$\mathbf{C} = \frac{\sum_{i=1}^{N} \left(\mathbf{V}_i - \overline{\mathbf{V}}_i\right)^2}{N} \tag{3.3}$$

Training of C in neural networks is achieved by cost volume regularization. A 3D UNet was performed on C using a 3-dimensional convolution of a four-scale network. The 3D convolution of C not only allows the model to learn depth features, but also enhances the robustness of the model. The final output is converted to probability using the softmax function, which reduces the effect of light reflections and texture on the images. This process is shown in Figure 3.8.

3.2.3 Depth estimation and model training

In this step, a depth map is estimated based on cost volume C, and a loss function is calculated by combining the ground truth depth of the images in the dataset to complete the final model training. The equation for

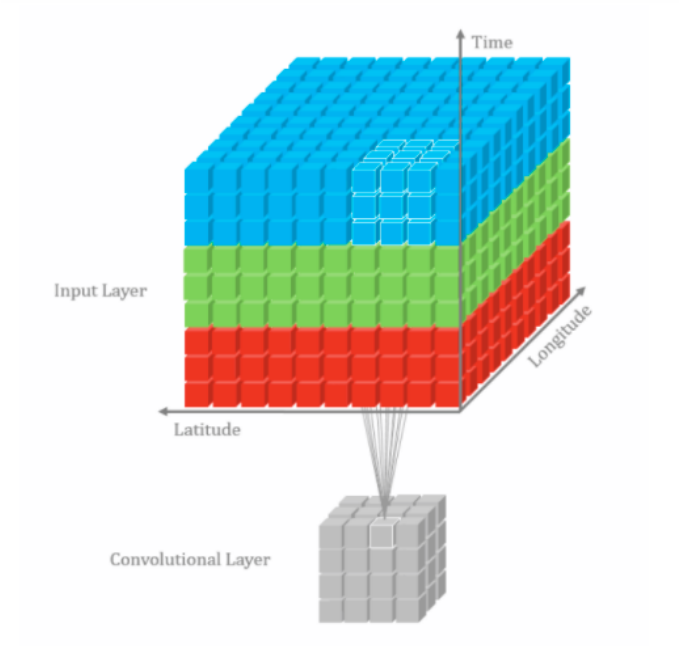

Figure 3.8: The process of 3D convolution.

the initial depth map is shown below.

$$\mathbf{D} = \sum_{d=d_{\min}}^{d_{\max}} d \times \mathbf{P}(d) \tag{3.4}$$

D is obtained by multiplying the maximum and minimum values of depth based on the epipolar geometric constraint by the corresponding summation of probabilities P(C).

Since very deep neural networks cause the gradient to disappear, which in this model is what makes the deep data too smooth (fuzzy). Therefore, in addition to obtaining the loss function from the initial depth value and the ground truth depth map from the dataset, an additional loss function is needed for regularization. This method obtains the refiend depth map by aggregating the original dataset images and the initial depth map into one image with 4 channels, and then performing 2D convolution to obtain the refiend depth map. The refiend depth map and the ground truth depth map form the second regularized loss function Loss2. The Loss function of the learning-based MVS model loss function designed in this paper is as follows:

$$Loss = \sum_{p \in \mathbf{p}_{valid}} \underbrace{\left\| d(p) - \hat{d}_i(p) \right\|_1}_{Loss0} + \lambda \cdot \underbrace{\left\| d(p) - \hat{d}_r(p) \right\|_1}_{Loss1} \tag{3.5}$$

CHAPTER 4. IMPLEMENTATION AND RESULTS

This chapter specifically introduces the use and experimental results of learning-based MVS for the 3D reconstruction of the Jinji Lake environment scene. And the implementation environment is shown in Table 4.1.

Table 4.1: Implementation Environment

Hardware Environment	Nvidia GTX 1080 Ti with 11G RAM; Intel Core i5-7400 CPU
Incremental SfM Environment	C++; Colmap; Visual Studio 2019; CMVS+PMVS
Dense Point Cloud Reconstruction Environment	Python 3.7; Pytorch 1.10; Open3D

4.1 Dataset collecting and processing

The datasets used in this paper include both the neural network model training dataset and Jinji Lake environment scene captured by UAV. The model training set is the DTU Datset (Jensen et al., 2014), which includes four file directories of images, corresponding depth maps, image pairs and camera parameters. The datatsets of Jinji Lake contains three catalogs consisting of photos, image pairs and camera parameters, captured by DJI MINI2 and processed by incremental SfM.

The features contained in the photo dataset are particularly important in order to obtain a high quality reconstruction model. Typically, a drone spiralling around a building and taking photographs will result in a large number of high coverage multi-view images, as shown in Figure 3.5. However, for a large scale aerial photograph of a lake like Jinji Lake, this approach would be time consuming and difficult to achieve full coverage. Therefore, the drone was kept at a constant altitude of 50m and was used to take full-scale aerial photographs of the target area in a roundabout way, which is shown in Figure 4.1.

The consumption of the learned MVS on the GPU is mainly caused by 3D convolution. Let the length, width, channel and depth of the cost volume of the convolution be W, H, C, D, and the size of the convolution object

Figure 4.1: UAV route over Jinji Lake environment.

is W*H*C*D. Taking into account the memory size of the graphics card, the size of the input picture W, H are reshaped into 1600, 1200.

4.2 Model training

This paper used the deep learning library Pytorch to build the MVS model, and used the GTX 1080 Ti GPU for calculations. Due to the limitation of GPU memory, the batch of model training is 1, that is, only one set of images is trained, and the depth number is set to 128. After 14 training epochs, the average Loss function value is less than 5. The process of model training is shown in Figure 4.2.

```
Epoch 9/16, Iter 2146/27097, train loss = 9.384, time = 0.459
Epoch 9/16, Iter 2147/27097, train loss = 3.752, time = 0.822
Epoch 9/16, Iter 2148/27097, train loss = 7.355, time = 0.465
Epoch 9/16, Iter 2149/27097, train loss = 0.899, time = 0.471
Epoch 9/16, Iter 2150/27097, train loss = 9.668, time = 0.470
Epoch 9/16, Iter 2151/27097, train loss = 38.239, time = 0.467
Epoch 9/16, Iter 2152/27097, train loss = 66.256, time = 0.467
Epoch 9/16, Iter 2153/27097, train loss = 9.868, time = 0.470
Epoch 9/16, Iter 2154/27097, train loss = 4.286, time = 0.467
Epoch 9/16, Iter 2155/27097, train loss = 0.560, time = 0.470
Epoch 9/16, Iter 2156/27097, train loss = 49.554, time = 0.464
```

Figure 4.2: The process of model training.

4.3 Model testing and point cloud reconstruction

The images acquired by the UAV were used to obtain camera parameters and photo matching pairs using Incremental SfM, which generated the characteristic sparse point cloud shown in Figure 4.3. After Incre-

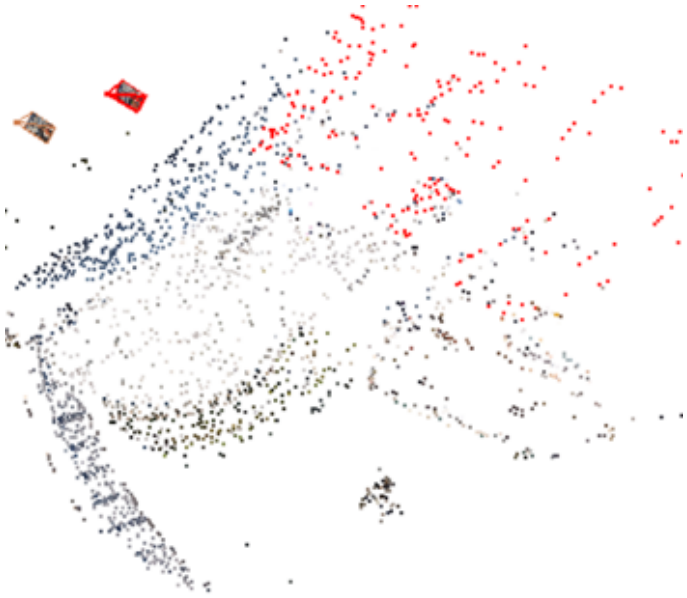

Figure 4.3: Sparse point cloud generating by SfM.

mental SfM, the resulting dataset is fed into the MVS network for dense modelling and the format of the dataset is shown in Figure 4.4. The MVS test network was loaded with the model after 14 rounds of training in the training network, and the dataset was processed by the size reshape. And then the depth estimation was achieved by the neural network while being transformed into a depth map of the same size. As mentioned before, the reconstruction base based on the depth map has the advantage of scalability, so the depth map can quickly realize the conversion to the dense point cloud. (Galliani et al., 2015*b*).

```
.
├── images
│   ├── 00000000.jpg
│   ├── 00000001.jpg
│   └── ...
├── cams
│   ├── 00000000_cam.txt
│   ├── 00000001_cam.txt
│   └── ...
└── pair.txt
```

Figure 4.4: Data set file structure.

4.4 Experimental results

The dense point cloud reconstruction of the part of Jinji Lake environment scene using the learning-based MVS is shown in Figure 4.5. It is significant that the quality of the point clouds generated with this reconstruction is better than the traditional MVS method, which is shown in Figure 5.6, for low-texture areas such as walls and light-reflective areas such as water surfaces. Meanwhile, the learning-based approach is tens of times faster than the traditional for both the DTU dataset reconstruction and the reconstruction of the Jinji Lake images, as shown in Figure 4.7.

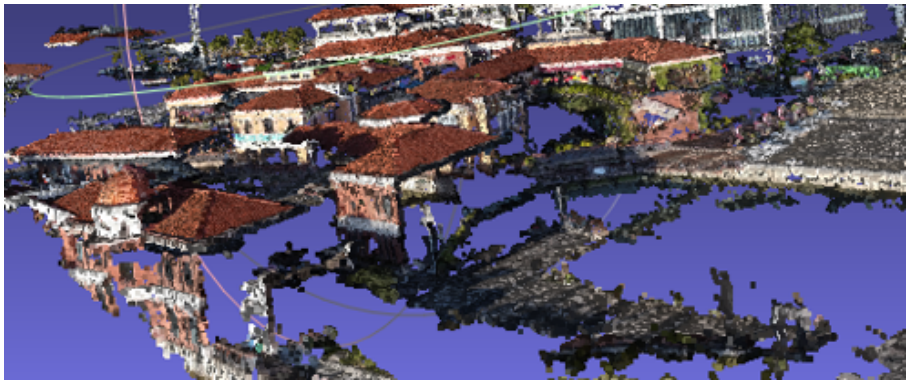

Figure 4.5: Dense Reconstruction generated by learning-based MVS.

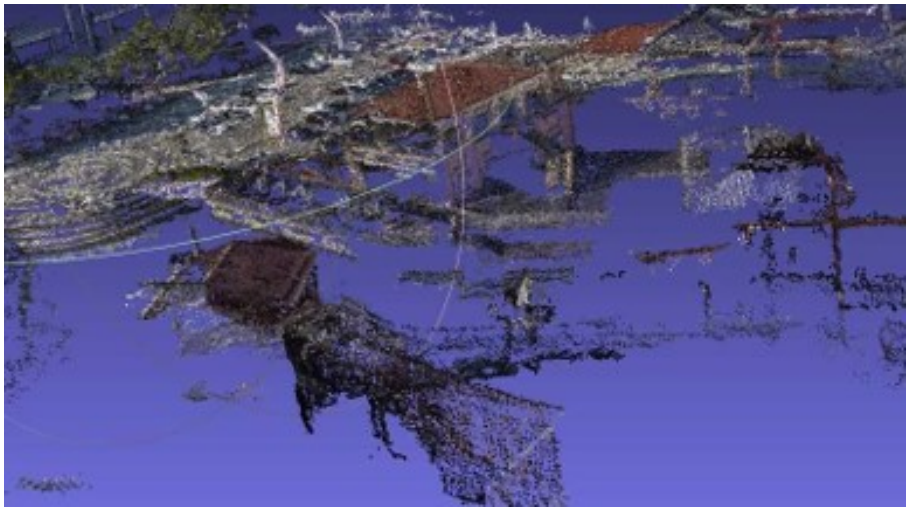

Figure 4.6: Dense Reconstruction generated by traditional method.

At last, the experiment has also implemented meshing and texture rendering using the Python open source library Open3D (**?**) and the open source software Meshlab (Cignoni et al., 2008). Poisson surface reconstruction, which is based on point cloud normal vectors for triangulation grouping modelling, has good performance and robustness in point cloud topological connectivity (Kazhdan et al., 2006). This paper therefore used Open3D to encapsulate the Poisson triangulation approach to point cloud meshing. And the 3D reconstruction model of Jinji Lake environment is shown in Figure 4.8.

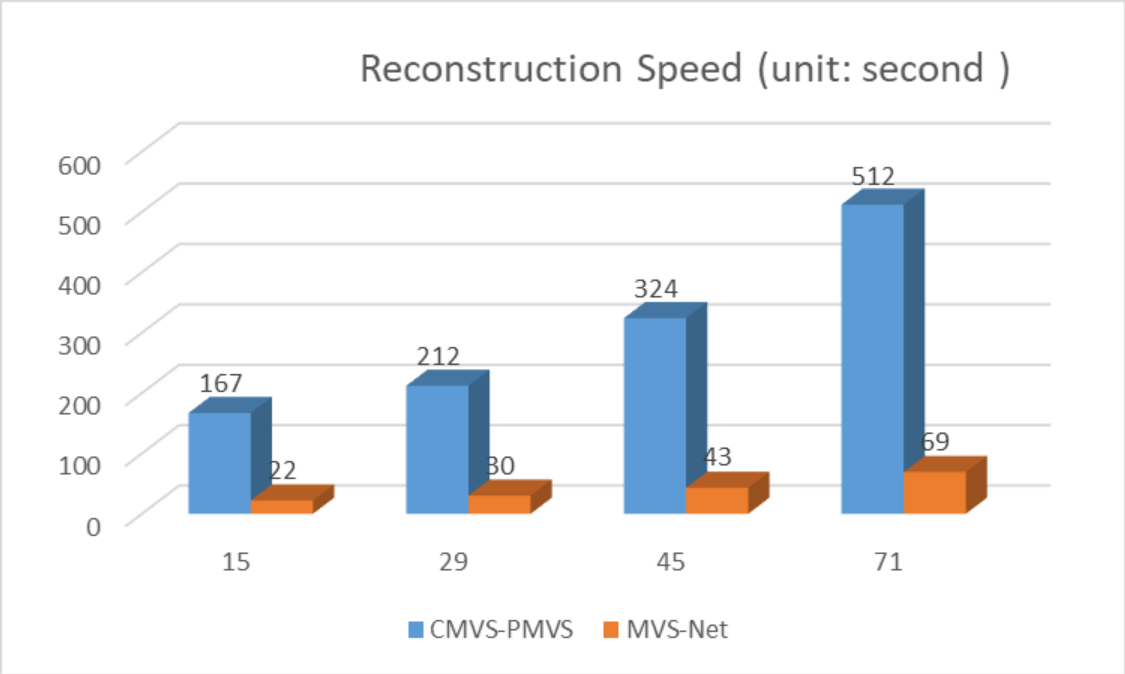

Figure 4.7: Comparison of the speed of reconstruction between the two methods.

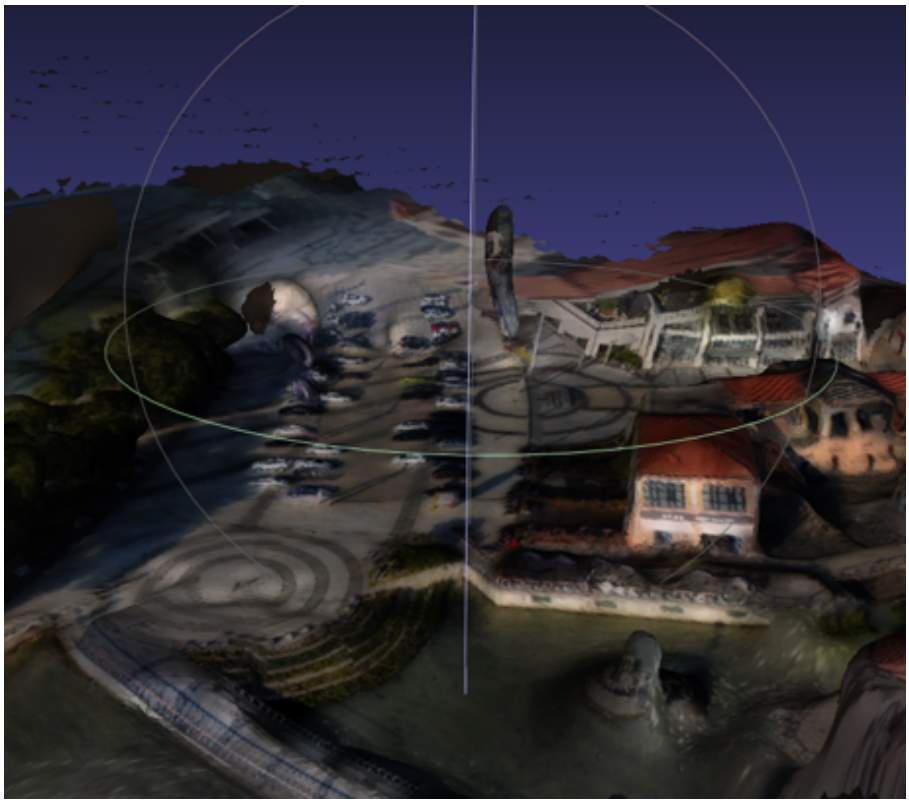

Figure 4.8: 3d model of part of Jinji Lake environment scene.

CHAPTER 5. CONCLUSION AND DISCUSSION

This chapter provides a summary of the work done throughout the paper. It also discusses methodological and experimental shortcomings, and looks forward to the future work.

5.1 Conclusion

This paper completed a 3D reconstruction of part of the landscape around Jinji Lake. The core task is to achieve a dense reconstruction of aerial multi-view images based on deep learning. The first step in the reconstruction is to solve for the camera's intrinsic and extrinsic parameters using the incremental SfM algorithm. After this, the MVS network model trained by DTU dataset performed a large-scale dense point cloud reconstruction based on the images. Based on the comparison between the reconstruction results and the traditional MVS's, the paper draws two conclusions: 1. The learning-based MVS outperforms traditional methods in terms of speed and quality of large-scale dense point cloud reconstruction; 2. The learning-based approach is not only suitable for objects such as buildings, tables and chairs provided by the dataset, but also works well for modelling large-scale aerial images.

5.2 Discussion

The content of the thesis completes the large-scale reconstruction and proves that the method also works well for large-scale data from UAV aerial photography. However, due to the limitation of computing power and algorithms, this project still has some problems and limitations. The limited video memory of the GPU resulted in the need to reduce the size of the image, which affected the effectiveness of the feature extraction. Also in this paper, the cost volume was regularised directly using 3D convolution, without considering how to optimise the memory footprint. There are also a number of problems with the 3D model of Jinji Lake landscape in the experimental section. This is due to the lack of performance of the drones and route design issues, which resulted in poor and incomplete coverage of the overlapping parts of the photographs obtained during aerial photography. The generated dense point cloud therefore contains approximately 200,000 3D point data, which is far less than even the millions of points in the dense reconstruction of Fig-

ure 3.5. Furthermore, the problem of the point cloud containing less information is even more pronounced after Poisson surface meshing.

Based on these issues, there are still many areas that can be optimised for the future work. Firstly, the data collection route can be optimised by referring to the literature and combining it with the UAV parameters. The next attempt is to use better GPU arithmetic to reconstruct the high-resolution original images, while optimizing the 3D convolutional layer. Also, this paper only considers the generation of dense point clouds, but not the manipulation of point clouds, such as filtering, segmentation, etc., which should be implemented in future work. Furthermore, if the reconstruction of part of the lake is extended to the whole lake or a larger area, it is clearly not possible to collect data by drone in one time. Therefore, in the future this project can solve this problem by implementing the following functions:

- 1. To combine EXIF data containing GPS information from image itself with 3D reconstruction to increase the progress of the reconstruction. And to make the flexible conversion of XY coordinates and latitude and longitude in the world coordinate system to enable 3D reconstruction models to be combined with other models or maps, etc.;
- 2. To perform point cloud reconstruction in batches of regions. Then to merge multiple point clouds using a point cloud alignment algorithm.

REFERENCES CITED

Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S. M. and Szeliski, R. (2011), 'Building rome in a day', *Communications of the ACM* **54**(10), 105–112.

Bleyer, M., Rhemann, C. and Rother, C. (2011), Patchmatch stereo-stereo matching with slanted support windows., *in* 'Bmvc', Vol. 11, pp. 1–11.

Bruno, F., Bruno, S., De Sensi, G., Luchi, M.-L., Mancuso, S. and Muzzupappa, M. (2010), 'From 3d reconstruction to virtual reality: A complete methodology for digital archaeological exhibition', *Journal of Cultural Heritage* **11**(1), 42–49.

Cernea, D. (n.d.), 'Openmvs: multi-view stereo reconstruction library. 2020', *URL: https://cdcseacave. github. io/openMVS* .

Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F. and Ranzuglia, G. (2008), MeshLab: an Open-Source Mesh Processing Tool, *in* V. Scarano, R. D. Chiara and U. Erra, eds, 'Eurographics Italian Chapter Conference', The Eurographics Association.

Crandall, D., Owens, A., Snavely, N. and Huttenlocher, D. (2011), Discrete-continuous optimization for large-scale structure from motion, *in* 'CVPR 2011', IEEE, pp. 3001–3008.

Ferdani, D., Fanini, B., Piccioli, M. C., Carboni, F. and Vigliarolo, P. (2020), '3d reconstruction and validation of historical background for immersive vr applications and games: The case study of the forum of augustus in rome', *Journal of Cultural Heritage* **43**, 129–143.

Furukawa, Y., Curless, B., Seitz, S. M. and Szeliski, R. (2010*a*), Towards internet-scale multi-view stereo, *in* '2010 IEEE computer society conference on computer vision and pattern recognition', IEEE, pp. 1434–1441.

Furukawa, Y., Curless, B., Seitz, S. M. and Szeliski, R. (2010*b*), Towards internet-scale multi-view stereo, *in* '2010 IEEE computer society conference on computer vision and pattern recognition', IEEE, pp. 1434–1441.

Galliani, S., Lasinger, K. and Schindler, K. (2015*a*), Massively parallel multiview stereopsis by surface normal diffusion, *in* '2015 IEEE International Conference on Computer Vision (ICCV)', pp. 873–881.

Galliani, S., Lasinger, K. and Schindler, K. (2015*b*), Massively parallel multiview stereopsis by surface normal diffusion, *in* 'Proceedings of the IEEE International Conference on Computer Vision', pp. 873–881.

Gherardi, R., Farenzena, M. and Fusiello, A. (2010), Improving the efficiency of hierarchical structure-and-motion, *in* '2010 IEEE computer society conference on computer vision and pattern recognition', IEEE, pp. 1594–1600.

Haile, A. T. and Rientjes, T. (2005), 'Effects of lidar dem resolution in flood modelling: a model sensitivity study for the city of tegucigalpa, honduras', *Isprs wg iii/3, iii/4* **3**, 12–14.

Han, X., Leung, T., Jia, Y., Sukthankar, R. and Berg, A. C. (2015), Matchnet: Unifying feature and metric learning for patch-based matching, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 3279–3286.

Harris, C., Stephens, M. et al. (1988), A combined corner and edge detector, *in* 'Alvey vision conference', Vol. 15, Citeseer, pp. 10–5244.

Hartley, R. I. (1997), 'In defense of the eight-point algorithm', *IEEE Transactions on pattern analysis and machine intelligence* **19**(6), 580–593.

Hartmann, W., Galliani, S., Havlena, M., Van Gool, L. and Schindler, K. (2017), Learned multi-patch similarity, *in* 'Proceedings of the IEEE International Conference on Computer Vision', pp. 1586–1594.

Heo, J., Jeong, S., Park, H.-K., Jung, J., Han, S., Hong, S. and Sohn, H.-G. (2013), 'Productive high-complexity 3d city modeling with point clouds collected from terrestrial lidar', *Computers, Environment and Urban Systems* **41**, 26–38.

Hu, S., Li, Z., Wang, S., Ai, M. and Hu, Q. (2020), 'A texture selection approach for cultural artifact 3d reconstruction considering both geometry and radiation quality', *Remote Sensing* **12**(16), 2521.

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.

Janota, A., Šimák, V., Nemec, D. and Hrbček, J. (2015), 'Improving the precision and speed of euler angles computation from low-cost rotation sensor data', *Sensors* **15**(3), 7016–7039.

Jensen, R., Dahl, A., Vogiatzis, G., Tola, E. and Aanæs, H. (2014), Large scale multi-view stereopsis evaluation, *in* '2014 IEEE Conference on Computer Vision and Pattern Recognition', IEEE, pp. 406–413.

Ji, M., Gall, J., Zheng, H., Liu, Y. and Fang, L. (2017), Surfacenet: An end-to-end 3d neural network for multiview stereopsis, *in* 'Proceedings of the IEEE International Conference on Computer Vision', pp. 2307–2315.

Kar, A., Häne, C. and Malik, J. (2017), 'Learning a multi-view stereo machine', *arXiv preprint arXiv:1708.05375* .

Kazhdan, M., Bolitho, M. and Hoppe, H. (2006), Poisson surface reconstruction, *in* 'Proceedings of the fourth Eurographics symposium on Geometry processing', Vol. 7.

Knapitsch, A., Park, J., Zhou, Q.-Y. and Koltun, V. (2017), 'Tanks and temples: Benchmarking large-scale scene reconstruction', *ACM Transactions on Graphics (ToG)* **36**(4), 1–13.

Lowe, D. G. (1999), Object recognition from local scale-invariant features, *in* 'Proceedings of the seventh IEEE international conference on computer vision', Vol. 2, Ieee, pp. 1150–1157.

Nister, D. and Stewenius, H. (2006), Scalable recognition with a vocabulary tree, *in* '2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)', Vol. 2, Ieee, pp. 2161–2168.

Schonberger, J. L. and Frahm, J.-M. (2016), Structure-from-motion revisited, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 4104–4113.

Setyawan, I., Timotius, I. K. and Kalvin, M. (2015), Automatic batik motifs classification using various combinations of sift features moments and k-nearest neighbor, *in* '2015 7th International Conference on Information Technology and Electrical Engineering (ICITEE)', IEEE, pp. 269–274.

Shi, G., Xu, X. and Dai, Y. (2013), Sift feature point matching based on improved ransac algorithm, *in* '2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics', Vol. 1, IEEE, pp. 474–477.

Snavely, N., Seitz, S. M. and Szeliski, R. (2006), Photo tourism: exploring photo collections in 3d, *in* 'ACM siggraph 2006 papers', pp. 835–846.

Song, L., Li, X., Yang, Y.-g., Zhu, X., Guo, Q. and Liu, H. (2018), 'Structured-light based 3d reconstruction system for cultural relic packaging', *Sensors* **18**(9), 2981.

Tola, E., Strecha, C. and Fua, P. (2012), 'Efficient large-scale multi-view stereo for ultra high-resolution image sets', *Machine Vision and Applications* **23**(5), 903–920.

Wu, C. (2011), 'Siftgpu: A gpu implementation of scale invariant feature transform (sift)(2007)', *URL http://cs. unc. edu/˜ ccwu/siftgpu* .

Wu, C. (2013), Towards linear-time incremental structure from motion, *in* '2013 International Conference on 3D Vision-3DV 2013', IEEE, pp. 127–134.

Xu, Y. and Stilla, U. (2021), 'Towards building and civil infrastructure reconstruction from point clouds: A review on data and key techniques', *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* .

Yang, J. and Yang, J.-y. (2002), 'From image vector to matrix: A straightforward image projection technique—impca vs. pca', *Pattern Recognition* **35**(9), 1997–1999.

Yang, X., Zhou, L., Jiang, H., Tang, Z., Wang, Y., Bao, H. and Zhang, G. (2020), 'Mobile3drecon: real-time monocular 3d reconstruction on a mobile phone', *IEEE Transactions on Visualization and Computer Graphics* **26**(12), 3446–3456.

Yao, Y., Luo, Z., Li, S., Fang, T. and Quan, L. (2018), Mvsnet: Depth inference for unstructured multi-view stereo, *in* 'Proceedings of the European Conference on Computer Vision (ECCV)', pp. 767–783.

Yao, Y., Luo, Z., Li, S., Shen, T., Fang, T. and Quan, L. (2019), Recurrent mvsnet for high-resolution multi-view stereo depth inference, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 5525–5534.

Zbontar, J., LeCun, Y. et al. (2016), 'Stereo matching by training a convolutional neural network to compare image patches.', *J. Mach. Learn. Res.* **17**(1), 2287–2318.

Zhang, Z. (2000), 'A flexible new technique for camera calibration', *IEEE Transactions on pattern analysis and machine intelligence* **22**(11), 1330–1334.

Zhou, Q.-Y., Park, J. and Koltun, V. (2018), 'Open3D: A modern library for 3D data processing', *arXiv:1801.09847* .