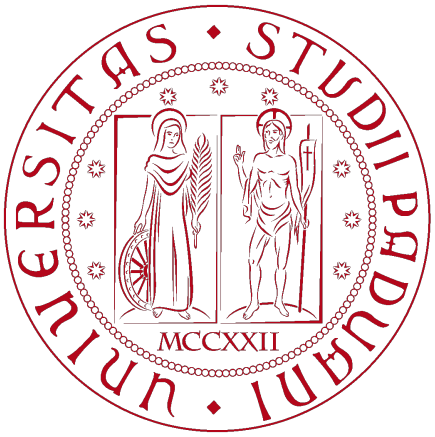


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Department of Information Engineering

Master's degree in Computer Engineering

Featuremetric Refined Structure From Motion with a Hand-held Camera and Point Cloud Registration in Urban Scenarios

Supervisor: Dr. Alberto Pretto

Author: Kiavash Ghamsari

Academic Year **2022/23**

Graduation Date: **13/07/2023**

Abstract

Structure from Motion (SfM), the task of recovering 3D scene structure and camera poses from 2D images or video frames, is a prominent topic in 3D Computer Vision. SfM has applications in various areas such as 3D modeling, augmented reality, robotics, and autonomous systems. Recent research has made significant improvements in the accuracy and the challenges associated with SfM. This thesis reviews and compares state-of-the-art approaches with a special focus on "Pixel-Perfect Structure-from-Motion with Featuremetric Refinement" paper. In our experiment, several videos from the city of Padova were captured using a bike-mounted camera and processed through the SfM algorithm. The generated 3D reconstructions are refined and re-evaluated after applying the aforementioned method. Next, an algorithm is developed to register the generated local point clouds with a global, georeferenced point cloud of the whole city acquired by an airplane equipped with a high-resolution LiDAR. Qualitative and quantitative experiments demonstrate promising results in generating accurate 3D reconstruction and consistent alignments between the reconstructed local point clouds and the global point cloud.

Contents

Chapter 1

Introduction

Structure from Motion is the task of calculating the 3D structure of a scene and the pose of cameras from a set of 2D images or video frames. It is a fundamental tool in many 3D Computer Vision applications such as 3D modeling, augmented reality, Robotics and Autonomous Systems. Several algorithms have been presented in this area. However, in general, most of them follow the same procedure. The incremental Structure from Motion algorithm, [9], consists of the following major steps. For every new image: - Feature detection and matching, i.e. identifying distinctive features and finding the correspondings in the previous images - Camera pose estimation using epipolar geometry - 3D point extraction by triangulation - and finally Bundle adjustment, i.e. optimizing the camera poses and 3D point positions by minimizing the reprojection error. (See Chapter 3)

Since the algorithm is incremental and iterative, without re-observing previously seen landmarks the errors increase for each new input image. Moreover, the reconstruction process may fail and it is no longer possible to record new images, with results that converge to a completely wrong 3D scene. Therefore, the accuracy and robustness of each step of the algorithm are crucial.

Some of the key challenges in feature matching, which is also a core task in many computer vision tasks, include occlusions, repetitive patterns, low-texture regions, and changes in lighting conditions. Noise and outliers, like moving objects, also can negatively impact the accuracy of the reconstructed 3D scene and camera poses. Moreover, SfM algorithms are computationally intensive, making them challenging to deploy in real-time or online applications where speed is crucial.

Recent papers have shown great improvement in all these challenges. In this thesis, some of the best approaches are reviewed and compared. Among these papers, "Pixel-Perfect Structure-from-Motion with Featuremetric Refinement", [6], has shown a prominent performance. Their method can be added as an extra refinement step to the SfM pipeline. In summary, this paper generates a feature map per image using Convolutional Neural Networks. Then, the position of the existing keypoints are adjusted by defining a flow from one point to another through a gradient descent update and a loss function based on the differences in their feature map values. Furthermore, in bundle adjustment's reprojection error, this paper considers the difference between the feature vectors instead of the typical euclidean distance between the original 2D data points and the reprojected 3D points (See chapter 3).

In this thesis, we acquired several datasets from a hand-held camera moving in an urban environment. Our camera is fixed on the head of a vehicle, and several videos are captured from the streets of the city center of padova. The frames of these videos are passed through Structure from Motion pipeline, and then, the results are refined by [6] paper mentioned above, obtaining very promising 3D reconstructions. Next, we will develop an algorithm that localizes the reconstructed point clouds inside a global map of the city. What if the task of localization becomes perfectly offline without any usage of GPS or other online technologies? A large-scale dataset of more than 3 million 3D points has already been acquired via an airplane equipped with a LiDAR over the city of Padova, covering an area of 1600m by 1000m. After some proper preprocessing, our reconstructed point clouds are registered within the big city point cloud, i.e. the position of our captured point cloud is found in the city. A detailed description of the proposed algorithm can be found in chapter 5.

This thesis is structured as follows:

- In Chapter 2 , the mathematical and geometrical background required for our work are presented, including camera geometry, epipolar geometry
- In Chapter 3, the algorithm of Structure from Motion is described in detail, and then, a selection of papers with the best contributions to this field are reviewed; With particular attention to "Pixel-Perfect Structure-from-Motion with Featuremetric Refinement", [6]
- In Chapter 4, we will use SfM in a real world scenario. We will develop an algorithm for the task of localization using the generated point clouds
- In Chapter 5, our dataset, metrics, experiment results, and their discussion are provided

- Lastly, in Chapter 6, the prons and cons of our method are discussed, and a few ideas for future improvements are suggested

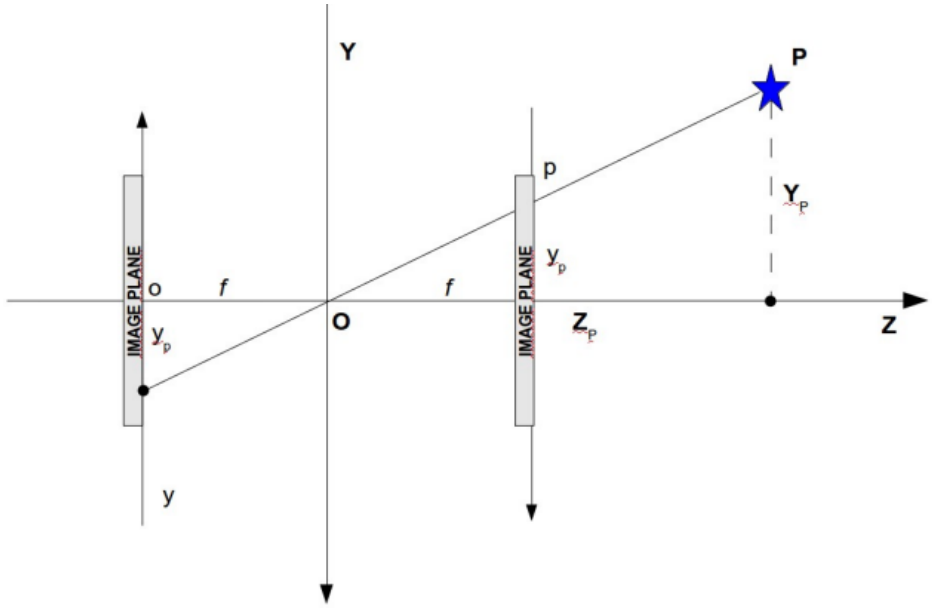
Chapter 2

Background

2.1 Pinhole Camera

A pinhole camera consists of a small hole and a plane behind. The light from the real-world object passes through the hole and forms an inverted and laterally inverted image on the plane which is at a distance F from the hole. For the sake of simplicity, it is assumed that there is a virtual screen in front of the hole at the same distance F. The same image from the real screen is excepted which is upright if the image falls on this virtual screen. The distance between the camera's center and the image plane is known as the focal length. The intersection of the optical axis of the lens and the image plane is the principal point. These parameters, which characterize the internal geometry of the camera, are known as intrinsic parameters.

Let f be the focal length, i.e. the distance between the image plane and the hole. And, principal point (c_x, c_y) are the coordinates of the optical center in the image plane. The 2D coordinates of a 3D point on the image plane can be calculated by similar triangles' equation:



$$x_p = \frac{X_P f}{Z_P} \quad , \quad y_p = \frac{Y_P f}{Z_P} \tag{2.1}$$

Extrinsic parameters describe how the camera is positioned and oriented in relation to the real world frame. The translation t and rotation R matrices, and scaling of the camera are some of these parameters.

a 3D point is projected onto the image plane, by transforming the point from world coordinate (X_w, Y_w, Z_w) system to the camera coordinate system using the extrinsic parameters (Rotation R and Translation t matrices). After having the coordinates of the 3D point from the center of the camera, i.e. (X_p, Y_p, Z_p), using the intrinsic parameters of the camera, the point is projected onto the image plane. This transformation from world frame to the image plane is encapsulated in the projection matrix.

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{X_P f}{Z_P} \\ \frac{Y_P f}{Z_P} \end{bmatrix}}_{\text{similar triangle equation}} \Rightarrow \begin{bmatrix} X_P f \\ Y_P f \\ Z_P \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix} \tag{2.2}$$

Let's consider Principal Point (u_c, v_c) in pixels, and (w, h) are the pixel width and

height in meters, respectively:

$$\alpha_u = \frac{f}{w} \tag{2.3}$$

$$\alpha_v = \frac{f}{h} \tag{2.4}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u_c + \frac{X_P \alpha_u}{Z_p} \\ v_c + \frac{Y_P \alpha_v}{Z_P} \end{bmatrix} \Rightarrow \begin{bmatrix} X_P \alpha_u + Z_P u_c \\ Y_P \alpha_v + Z_P v_c \\ Z_P \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_c & 0 \\ 0 & \alpha_v & u_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix} \tag{2.5}$$

Let K be a 3×3 matrix that contains the intrinsic parameters:

$$K = \begin{bmatrix} \alpha_u & 0 & u_c \\ 0 & \alpha_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

Finally, the point P in real world coordinates system is transformed to the image plane, u, by:

$$u = K[R|t]P \tag{2.7}$$

As can be seen from the equations, the depth (Z) of the real world points have vanished during the projection. In the next sections, we will see how to recover the depth.

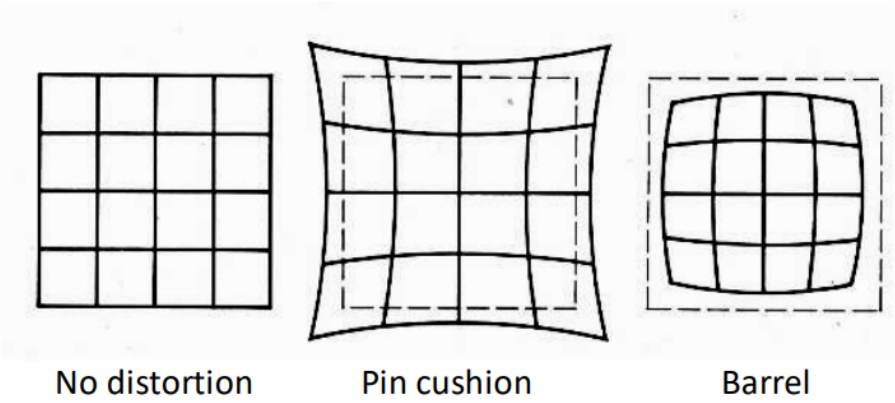
2.2 Camera Distortion and Calibration

Distortion refers to the deviation between the ideal pinhole camera model and the actual camera used to capture images. The pinhole camera model assumes that light rays pass through a single point, or the pinhole, before forming an image on a flat image plane. However, real-world cameras have imperfect lenses, that don't let rays reflect straightly and cause distortions to the image.

There are two kinds of camera distortions:

1. Radial distortion is caused by the curvature of the camera lens. This type of distortion causes straight lines to appear curved in the image, especially near the edges of the image.

2. Tangential distortion occurs when the lens is not aligned perfectly parallel to the image plane. This causes the image to appear skewed, with some parts appearing closer or farther from the camera than they should.



Homography: Let there be two images from the same planar scene, e.g. a rectangular identity card, but from different viewpoints. The second view can be obtained from the first view by multiplying the first image by a 3x3 matrix called Homography matrix.

Camera calibration is the process of estimating the intrinsic parameters like focal length, principal point, and distortion coefficients, as well as the extrinsic parameters like camera position and orientation for each image. The accuracy of this step has a significant impact on 3D reconstruction. As it is mentioned in previous sections, the depth of a pixel has a direct relation with focal length. The depth can be in meters, while focal length is usually around millimeters. Therefore, a small error in focal length can cause high misplacement of a 3D point's position. A checkerboard is used to take pictures of for calibration since it is a planar object and it has distinctive features, i.e. checkerboard corners, that are easy to detect. A set of pictures are taken from different points of view. By using homography, an initial guess for intrinsic parameters is calculated. Then, the final calibration parameters are refined by optimizing the reporjection equations for checkerboard corners.

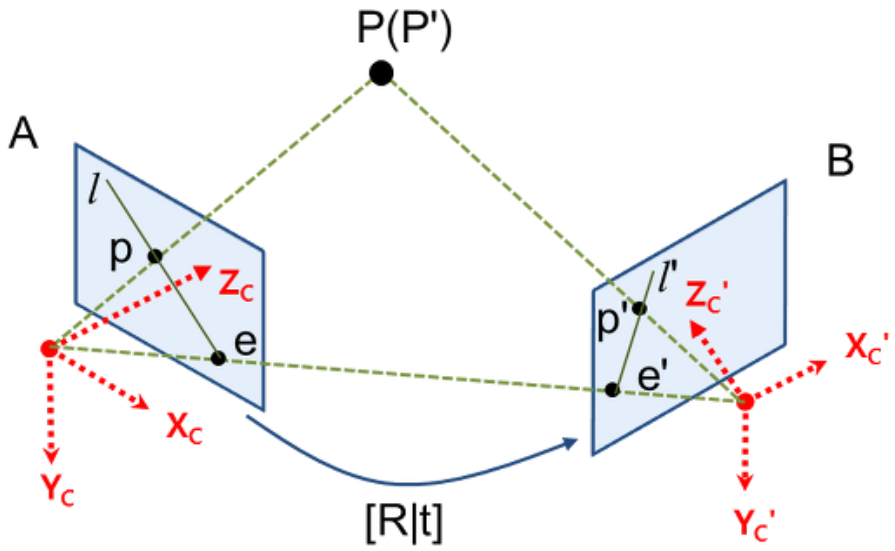


Figure 2.1: Stereo Cameras

2.3 Stereo Cameras

Epipolar geometry refers to the geometry of two images of a 3D scene taken by two cameras. The two cameras are presumed to have a defined relative pose, which is the position and orientation of one camera in relation to the other, and a known internal calibration (i.e., focal length, principle point).

Epipolar lines: Epipolar line is the intersection of the image planes with a plane that passes through the two camera centers and a 3D point in the scene. Any point in one image must lie on the epipolar line for its equivalent point in the other image. In figure 2.1, l and l' are the epipolar lines.

Epipoles: The epipole is the location where the picture planes of the two cameras are intersected by the baseline, which connects the centers of the two cameras. To put it another way, each camera perceives the epipole of the other camera as a projection of the other camera's center.

Essential Matrix: Let $\tilde{p}_l$ and $\tilde{p}_r$ be the vectors from the center of cameras to the points p and p' from figure 2.1, meaning that the intrinsic parameters are known.

Recalling translation (t) and rotation (R) matrices, $(t \times R\tilde{p}_r)$ is the cross product vector which is orthogonal to the epipolar plane. Hence,

$$\tilde{p}_l \cdot (t \times R\tilde{p}_r) = 0 \tag{2.8}$$

Essential matrix is a 3x3 matrix that relates two cameras' poses of two viewpoints. It encodes, the relative position and orientation of the two cameras in relation to one another:

$$E = R[t]_\times \tag{2.9}$$

Where $[t]_\times$ is

$$[t]_\times = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} \tag{2.10}$$

By replacing the essential matrix in equation 2.8:

$$\tilde{p}_l^\top \times E \times \tilde{p}_r = 0 \tag{2.11}$$

Having the linear equation above for a set of corresponding points between two images, the essential matrix can be calculated. The most commonly used algorithm to find the essential matrix is known as "Eight-point algorithm".

In 3D vision problems, once the essential matrix has been calculated, the relative position and orientation of the two cameras, i.e. the rotation matrix R and translation vector t, are extracted.

Fundamental Matrix: A 3x3 matrix that contains not only relative camera poses but also camera intrinsic parameters. By having Essential matrix, K and K' as intrinsic camera matrices, the fundamental matrix is defined as follows:

$$F = K^{-T} \times E \times K'^{-1} \tag{2.12}$$

The same as essential matrix, for points p and p' on image planes:

$$p \times F \times p' = 0 \tag{2.13}$$

To calculate the fundamental matrix, the linear equation above can be expanded as follows:

$$p = \begin{bmatrix} x & y & 1 \end{bmatrix}, F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}, p' = \begin{bmatrix} x' & y' & 1 \end{bmatrix}$$

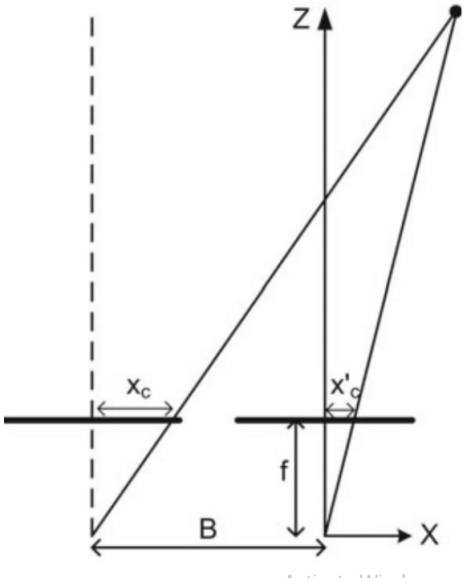


Figure 2.2: Rectilinear Rig

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0$$

By rewriting the equation above for multiple correspondences, the homogeneous linear system can be solved using SVD method.

2.4 Two View Geometry

Two-view geometry is the task of finding the relation between two images captured from the same scene by two cameras with different viewpoints, and detecting the 3D position of points in the scene. We start by detecting the depth of a pixel with basic geometry calculus, and then, generalize it to all possible camera settings.

Rectilinear Rig Starting with two cameras with the same intrinsic parameters on the same baseline, i.e. the planes of two cameras are aligned in one line as shown in figure 2.2, By using similar triangles equations, the depth of a keypoint can be calculated:

$$x'_c = f\frac{X}{Z} \quad , \quad x_c = f\frac{X+B}{Z} \tag{2.14}$$

$$d = x_c - x'_c = \frac{fB}{Z} \tag{2.15}$$

$$Z = \frac{fB}{d} \tag{2.16}$$

Let d, from the equation above, called the disparity which is the difference in the coordinates of the pixels in both images pointing to the same 3D point. Therefore, the only challenge is to find the disparity of each pixel.

For other camera settings , e.g. non-parallel image planes and epipolar lines, different camera parameters, the fundamental matrix is the key option that can be used to find the depth of points. A 3D point can be defined as the intersection of the rays that start from the centers of cameras and cross the corresponding 2D keypoints in each image plane. In practice, first, the fundamental and the essential matrices are calculated by finding the accurate feature matches between the two images. Then, after having the relative pose of cameras, for all pixels, the corresponding pixel from the other image is found along the epipolar lines. The epipolar lines help to limit the search space for the matching process from all pixels over the images to only the pixels on epipolar line and its surroundings. By having camera poses and all matches, 3D points could be obtained by triangulation.

Chapter 3

Structure from Motion

3.1 Structure from Motion Algorithm

Structure from Motion is the task of calculating the 3D structure of a scene and the pose of cameras from images captured in multiple views or video frames. In this thesis, Incremental Structure from Motion, [12], will be explained as it is the base algorithm for most papers in these criteria.

The algorithm can be viewed as a pipeline that contains the following separate steps:

1. **Feature detection and matching:** The first step in SFM is feature detection and matching, where the algorithm identifies common features or points in the 2D images or video frames. These features can be edges, corners, blobs, or other distinctive patterns that can be detected consistently across the images. The algorithm, then, matches these features between different input frames. This step is one of the most challenging parts in SfM. Occlusion, illumination changes, perspective transformation, and moving objects

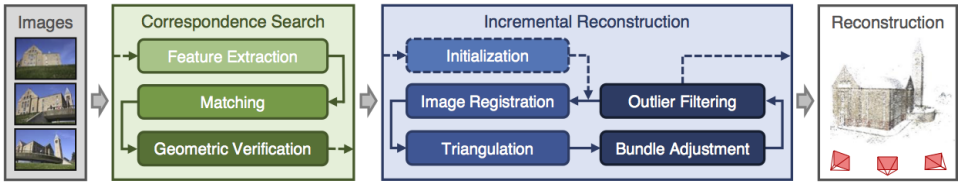


Figure 3.1: An overview of Structure from Motion Revisited, [12] pipeline

are some of them. However, since this is a core task in many Computer Vision tasks, there has been tons of research in feature detection and matching, including deep based approaches. SIFT ([7]), SuperPoint ([2]), D2-Net ([3]), R2D2 ([8]) are often used among SfM related papers which will be described briefly in the next chapter.

2. **Geometric Verification:** Raw matches are not enough for the scene reconstruction. Matches in similar repeated patterns, multiple the same objects in one image, etc, are some issues that may cause wrong poses and 3D points. So, outliers must be removed from the list of matches. By using the existing matches and epipolar geometry, camera matrices(i.e. H, F, and E matrices) are calculated. Then, the rest of the matches are verified by checking the geometric consistency of the reconstructed 3D points and camera poses. If they do not result in the same poses, they are filtered out. Having more matches is good. But, not all of them can be used.

3. **Reconstruction Initialization:** If the pipeline is considered as an iterative or sequential process, a starting point is essential. Choosing a good initial pair of images is so important. At the end of each iteration, the camera parameters and 3D structure are optimized. Without a good initial estimate, the optimization process usually doesn't converge and is stuck in local minima. A good initial pair usually is chosen from the images with the highest number of matches.

4. **Image Registration and Triangulation:** After initialization, the algorithm calculates the camera pose for the new image by methods in section 2.3 and uses triangulation to estimate the 3D position of the matched features in every new image. Triangulation is the process of finding the intersection point of two or more vectors that originate from the center of the camera and pass through the 2D features. The intersection point represents the 3D position of the feature in the scene.

5. **Bundle Adjustment:** As the previous processes involve some errors. There are inconsistencies and misalignments between the estimated camera parameters and the triangulated 3D points, and since the algorithm is incremental, the errors are accumulated for every new image. Bundle adjustment refines those parameters by minimizing the non-linear reprojection equations. For each generated 3D point, its 3D coordinates are transformed into the corresponding 2D image coordinates using the estimated camera projection matrix. Then, the difference between the observed 2D projection and the reprojected 2D coordinates is considered as the error. Levenberg-Marquardt algorithm is usually used to minimize the error. The optimization problem

can be high-dimensional, and it may take a long time to converge to a good solution. Nonetheless, bundle adjustment can significantly improve the accuracy of the reconstructed 3D scene and camera poses.

3.2 Related Works

Structure of Motion is a core task for many other projects. The accuracy and speed of the algorithm is crucial. For instance, even a minor rotation error of 1 degree in the camera can lead to significant misalignment of point coordinates in meters, particularly in open scenes. Also, during our experiment, it is observed that the pipeline fails in the initialization step for a considerable number of attempts because of the lack of correct matches. Therefore, over the past few decades, there have been lots of efforts to improve each part of these algorithms. Here is a review of the recent best papers in this area:

3.2.1 Researches in Feature Matching

The first step in SfM pipeline is feature matching which is one of the core tasks in many Computer Vision applications as they are used to identify the corresponding matches among the input images. Some of the challenges are deformation in different viewpoints, occlusion, illumination changes, moving objects, etc. Here is a summary of improvements specifically related to SfM algorithm:

Multi-View Optimization of Local Feature Geometry ([4]) refines the existing keypoints. It tracks each keypoint in all input images and creates a tentative matches graph with keypoints as nodes and matches as edges. Each keypoint is known by its surrounding pixels information. So, a h*w patch is selected around the feature and a "d" dimensional descriptor is calculated by a neural network with Siamese architecture. Then, for each pixel, a dot product similarity is calculated with every pixel in the other patch (h*w*h*w dimension). After normalization, the pixel that has the most similarity value with keypoint is supposed to be selected, and the translation vector ($T_{u->v}$) moves the keypoint to the selected pixel. In the next stage, since each keypoint could be seen in multiple images, a single incorrect match or displacement can cause a cascade of errors in the results. Therefore, for each track of matches, a non-linear equation for minimizing the dot similarity between the patch and all patches for the same keypoint in other images is optimized. The idea of this paper is also used in [6] which will be described in detail in the next chapter.

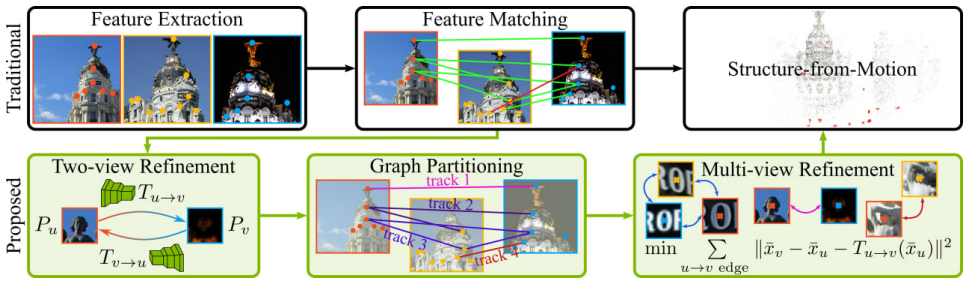


Figure 3.2: Paper: Multi-View Optimization of Local Feature Geometry, [4]

Deep Two-View Structure-from-Motion Revisited ([17]) uses optical flow to predict dense matches between two frames. They use DICLFlow, explained in [18] to generate dense matching points between two consecutive frames. Before calculating the essential matrix, noisy matches like moving objects are filtered out. They assumed that optical flow is more accurate in rich textured areas. So, they used SIFT method to detect less textured areas in order to mask out the optical flow there. Then, the essential matrix is calculated using the classic five-point algorithm with RANSAC, and the dense depth map is computed by performing triangulation. Before this process, the matching is performed for the second time by constraining the search space to epipolar lines computed from the relative camera poses. They filtered out noisy matches twice which means dense matches by optical flow are too noisy. Higher number of matches is good for obtaining more points in sparse reconstruction. However, they tend to be noisy. They also, introduced a Scale-Invariant Depth Module to deal with the up-to-scale relative pose and mismatch between the scale of the camera poses and the depth map. Up to scale problem means that while the relative positions of the reconstructed points are correct, their absolute positions in the real world are not known unless there is additional information, such as the size of an object in the scene or the distance between two known points. In order to deal with this problem, in each pair of images, this paper generates a number of matching candidates with different real world depths with the same interval per keypoint. And then, a plane-sweep powered network minimizes the loss function which is the position displacement of the features in the flow. This paper is able to find more and better matching points and therefore more accurate poses and depth maps, especially for textureless and occluded areas.

There is much more research in the feature matching step. [8], [3], and [2] are supervised deep learning based feature extractors that not only have outperformed handcrafted methods in terms of accuracy but also, could solve many challenges in this area like point of view challenges and multi-scale problems. Supervised learn-

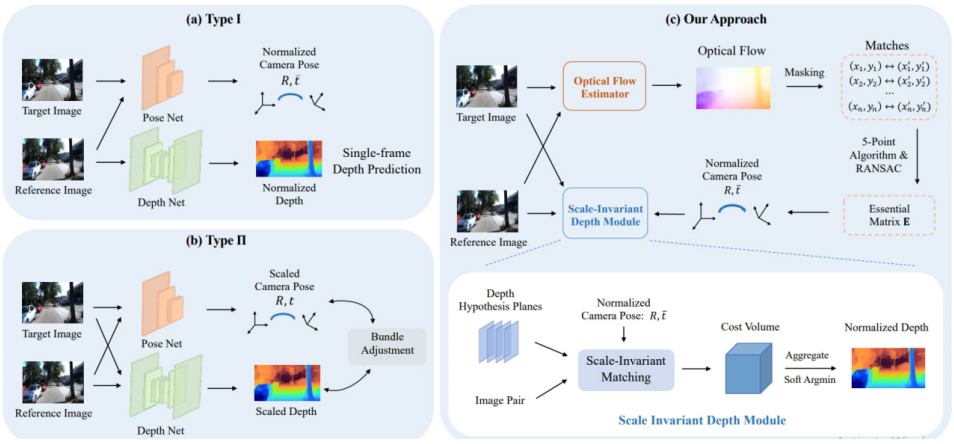


Figure 3.3: Paper: Deep Two-View Structure-from-Motion Revisited, [17]

ing means that the model has been trained based on the keypoints already defined in the training dataset manually. Their problem is that they have less performance for the kinds of features other than in the training dataset. Therefore, for every application, a special dataset must be prepared and the model must be trained, i.e. the lack of robustness from subject to another. Some of these papers have provided modules to train the model on your own customized features dataset.

3.2.2 Researches in Bundle Adjustment

Bundle Adjustment algorithms optimize the camera parameters and the position of 3D points by minimizing the geometric errors between the projected 3D points and the corresponding 2D image observations. For every new input image, the errors are accumulated, and so, it increases exponentially in the subsequent iterations. There are some disadvantages in this technique. Bundle Adjustment requires solving a large system of non-linear equations, which can be computationally expensive, especially for large datasets. This can be a significant drawback when it comes to real-time usage. In the experiment chapter, it will be shown that this is the most time-consuming part in Structure from Motion pipeline. Moreover, these non-linear equations have considerably a high number of local minima due to the noises in matches, and they are sensitive to the initial values. If the initial values are not accurate, the optimization may fail to converge. Here are some of the best papers focused on these issues.

BA-Net: Dense Bundle Adjustment Network ([13]) implemented feature-metric bundle adjustment that minimizes feature-metric errors instead of geometric erros between feature pyramids of images obtained by CNNs. As [5] says, there are two types of BAs:

1. Typical geometric BA with re-projection error(pixel coordinates): Only a few pixels, i.e. keypoints, are taken into account which comes with keypoints detection and matching challenges
2. Photometric BA algorithm(all aligned pixel, pixel intensities as error): It has good accuracy, especially at less textured scenes. However, the disadvantages would be sensitivity to camera exposure, illumination changes, and outliers such as moving objects. Also, considering all pixels would increase the computation dramatically.

BA-Net creates a pyramid of features for each image and aligns them. The feature pyramid is generated by multi-scale hierarchy of CNN(DRM-54, [20]). Then, the BA equation to minimize would be:

$$e^f_{i,j}(X) = F_i(\pi(T_i, d_j \cdot q_j)) - F_1(q_j) \tag{3.1}$$

where $F = \{F_i | i = 1...N_i\}$ are the feature pyramids of images $I = \{I_i | i = 1...N_i\}$. $d_j \in D = \{d_j | j = 1...N_j\}$ is the depth of a pixel q_j at the image I_1, and $d_j.q_j$ upgrades the pixel q_j to its 3D coordinate. The function π projects the points to image space. Thus, the optimization parameter is $X = [T_1, T_2...T_{N_i}, d_1, d_2...d_{N_j}]$.

For the generation of dense reconstruction, the depth of all pixels in all images are required. However, if all pixels are considered as independant variables, the computation of their values is super expensive. This paper, also, uses [14] and [19] approaches to deal with this issue. A set of arbitrary basis depth maps is created. Then, the final depth map is generated as a linear combination of these basis depth maps (”B”):

$$D = ReLU(w^T * B) \tag{3.2}$$

”w” will be optimized in our BA-Layer. Generally, this idea of this solution can be generalized into problems with a high number of values to calculate.

3.2.3 Other Papers

There are many papers that tackle Structure from Motion problems under Visual SLAM (visual simultaneous localization and mapping) and odometry subjects.

Deep Patch Visual Odometry [16], utilizes a pair of residual networks to extract a collection of patches from incoming sequential video frames. One network extracts matching features while the other extracts context features for each patch. They proposed a module called ”update operator” to update both poses and patches. The input to this module is the patch graph containing information of the patch trajectories, and the output is the updated graph. These inputs are passed through recurrent neural networks with adding residuals before they are passed to the differentiable bundle adjustment layer. The idea of using a sparse patch representation and training the networks with an existing patch dataset related to a specific use case improves efficiency. Their method is capable of running at 2x-5x real-time frame rates. It is because their main goal is to refine camera poses not the position of all points.

DROID-SLAM [15] uses neural networks to estimate dense flow fields which are subsequently used to optimize depth and camera pose. For each new video frame, DROID-SLAM produces a differentiable Dense Bundle Adjustment and Gauss-Newton solver to update camera poses and dense per-pixel depth to maximize their compatibility with the current estimate of optical flow.

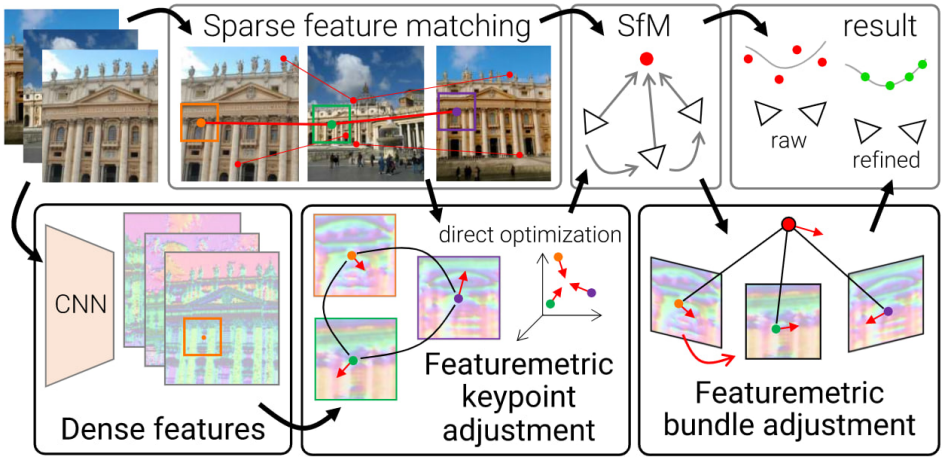


Figure 3.4: Paper: Pixel-Perfect Structure-from-Motion with Featuremetric Refinement, [6]

3.2.4 Pixel-Perfect Structure-from-Motion with Featuremetric Refinement

Among all reviewed papers, [6] has the best performance. The paper proposes two stages to improve the accuracy of structure-from-motion for 3D reconstruction. In the first stage, the initial keypoint locations in the 2D images are adjusted prior to any geometric estimation by optimizing a direct cost over dense feature maps obtained by Convolutional Neural Network. In the second stage, the bundle adjustment equations refine the 3D points and camera poses using a featuremetric error based on dense features map. The paper produced accurate reconstructions and scaled well to large scenes with thousands of images. Also, their contribution could be used as an extra refinement step in the SfM pipeline. So, it can be used along with other papers' contributions.

They use the initial idea of [4] that separates the tracks for each keypoint, and adjusts their locations by optimizing the geometric cost over the input images in which the keypoint is seen. [4] improved SfM, but has limited accuracy and scalability. To solve that, This paper proposes a feature map for each image using convolution neural networks. The feature map condenses the dense information from surrounding pixels into a vector for each pixel. The reason behind this step is the fact that refining the position of the keypoint is a local operation, not based solely on the pixel value, and the dense information only needs to be locally accurate and invariant but not globally discriminative. For the first stage, i.e. keypoint

adjustment refinement, the locations of 2D keypoints belonging to the same track j is refined by optimizing its featuremetric consistency along tentative matches.

The error between image intensities at two sparse observations is $r = I_i[p_u] - I_j[p_v]$. I_i refers to the images i-th, and p_u refers to u-th pixel's intensity. The flow between two points in an image can be inferred from the local image derivatives through a gradient descent update:

$$T_{v \to u}[p_v] \propto -\frac{\partial I_j}{\partial p}[p_v]^\top . r \tag{3.3}$$

This can be used to optimize the photometric error. However, as it is said above, the feature map values are used instead of direct pixel intensities. Therefore, the simplified loss function used to adjust the locations of 2D keypoints belonging to the same track j:

$$E = \sum_{(u,v) \in M_j} \| F_u[p_u] - F_v[p_v] \| \tag{3.4}$$

In comparison to [4], optimizing by the cost of feature maps instead of the patch neighboring pixels has better efficiency.

In the second stage, i.e. bundle adjustment refinement, the typical approach uses the euclidean distance between the point and its reprojected 3D from another view. However, this paper considers the difference between the feature vectors of the point and the feature vector of the point where the 3D point is reprojected. For each track j, the error between its observations and a reference appearance f_j is as follows:

$$E = \sum_j \sum_{(i,u) \in \tau_j} \| F_i[\prod (R_i P_j + t_i, C_i)] - f^j \|_\gamma \tag{3.5}$$

Also, instead of acquiring the cost equation for each pair of views, one image is considered as a reference, and the cost equations are written between the reference image and the rest of the views. it reduces the number of residuals from $O(N^2)$ to $O(N)$. This is not good for sequential video frames. Because newer frames will be too far from the reference image. However, it is achievable if the new input is a batch of new images, and the refinement is applied to each batch separately.

As it is mentioned earlier, this paper can be integrated with other approaches. In their experiment, [8], [2], [3], and [2] are used as base feature detectors, and then

applied refinement. In ETH3D dataset, [11], it is reported that accuracy is improved with an average of 10 to 15 percent for each base feature detector. And, the runtime is decreased significantly in comparison to [4].

Chapter 4

Featuremetric Refined Structure-From-Motion and Point Cloud Registration for Localization

We will explore the application of Structure from Motion in a real-world scenario within the field of 3D computer vision. Our goal is to localize a robot in a large city environment using point cloud registration. Localization refers to the task of determining the position and orientation of a robot within its environment. And, point cloud registration is defined as finding the transformation, i.e. translation and orientation, that aligns the points in one point cloud with those in another.

Outdoor localization, today, relies heavily on GPS technology, accompanied by ground based augmentation systems to improve accuracy. However, in this thesis, we are going to explore a method that makes it offline, meaning eliminating the need for GPS or similar devices.

4.1 Point Cloud Generation

4.1.1 Image Acquisition

A GoPro9 camera is mounted on the head of a vehicle, and various videos are taken from the streets. The videos were recorded at a frame rate of 60 fps. The resolution of each image is 1920*1080 pixels, and the distance covered in each

video ranges from 80 to 120 meters. We used the predefined linear settings of the camera, meaning that the camera undistorts the videos automatically. We also, tested with distorted settings and manual calibration using checkerboard method (See chapter 5). However, using undistorted videos showed better results. The software we use for SfM, will also guess the calibration parameters and refine it during the reconstruction. Next, the frames are extracted from each video. The rate of the sampling is chosen at 3fps. And, each dataset comprises an average of 60 frames.

4.1.2 Reconstruction

COLMAP is an open source software, implemented by [9] and [10], that provides 3D reconstruction based on 2D images. There are 2 types of reconstruction:

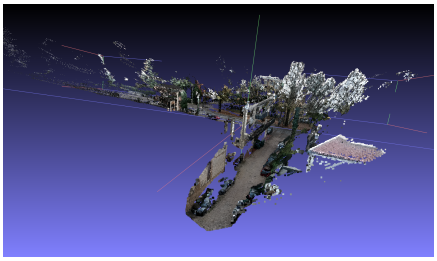
- Sparse Reconstruction estimates the positions of a selected set of of key-points detected on the input images
- Dense Reconstruction provides 3D positions of all pixels in the input images by generating depth maps. This can be achieved through techniques like stereo matching or depth estimation algorithms, like PatchMatch algorithm [1]. Dense reconstruction provides a richer representation of the scene but is more computationally demanding and requires higher memory storage.

Both spare and dense reconstructions are implemented in COLMAP. The choice between sparse and dense reconstruction depends on the specific application requirements and the trade-off between accuracy and computational resources. For our method, we need dense point clouds. Figure 4.1 shows examples of both reconstructions. Each step of SfM pipline can be executed separately. So, other algorithm can be replaced by its defaults. And, extra steps, like refinements, can be added to the pipeline.

4.1.3 Refinement

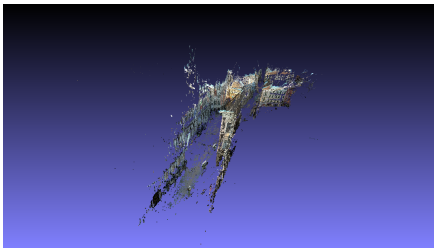
Pixel-Perfect paper [6] is decided to be used for our reconstruction refinement. First, in order to verify that this paper can actually refine the reconstruction, a test dataset is created from a cereal box on a table, and dense 3D point cloud is obtained by COLMAP and then, is refined by pixel perfect. In figure 4.3, it can be clearly seen that their approach is improving the reconstruction. Recalling stereo cameras, since the camera poses become more accurate, the epipolar lines are better aligned. So, there would be more matches and more accurate disparities. In our datasets, which contain points of street, and buildings, the refinement results in more 3D points of streets, figure 4.2. We will see later, that more coverage of the streets

(a) Example 1: Sparse Point cloud



(b) Example 1: Dense Point cloud

(c) Example 2: Sparse Point cloud



(d) Example 2: Dense Point cloud

Figure 4.1: Comparing sparse and dense reconstructions, generated by COLMAP [9] and [10]

(a) Before Refinement (b) After Refinement

Figure 4.2: Sparse reconstruction refinement using Pixel-Perfect paper, [6]

is crucial for our method of localization. In our pipeline, first, the initial sparse point cloud and camera poses are generated by COLMAP. Then, they are refined by pixel-perfect algorithm. And after that, the dense construction is executed for each dataset.

4.1.4 Aerial Point Cloud

An extensive collection of 3D points is obtained by a LIDAR sensor from the city of Padova using an airplane. The covered region spans 1600m by 1000m, and the average nearest neighbor distance of points is 0.63m. The dataset contains a total of 3,583,803 points, figure 4.4.

4.2 Point Cloud Registration

The objective is to align the ground point cloud with the aerial point cloud. Initially, we attempted to register the point cloud using traditional global and local algorithms like FPFH feature matching, RANSAC, and ICP. However, due to the significant differences in the nature of the datasets, the results were too poor. The aerial point cloud primarily consists of street points and buildings' roof, as it is captured from above. While the ground point cloud contains street points and building walls. Therefore, we had to find the common features and simplify the problem. The new pipeline could be described as follows:

1. It was observed that viewing both point clouds from the top point of view, aligned with the z-axis, provides valuable information. The first step is to

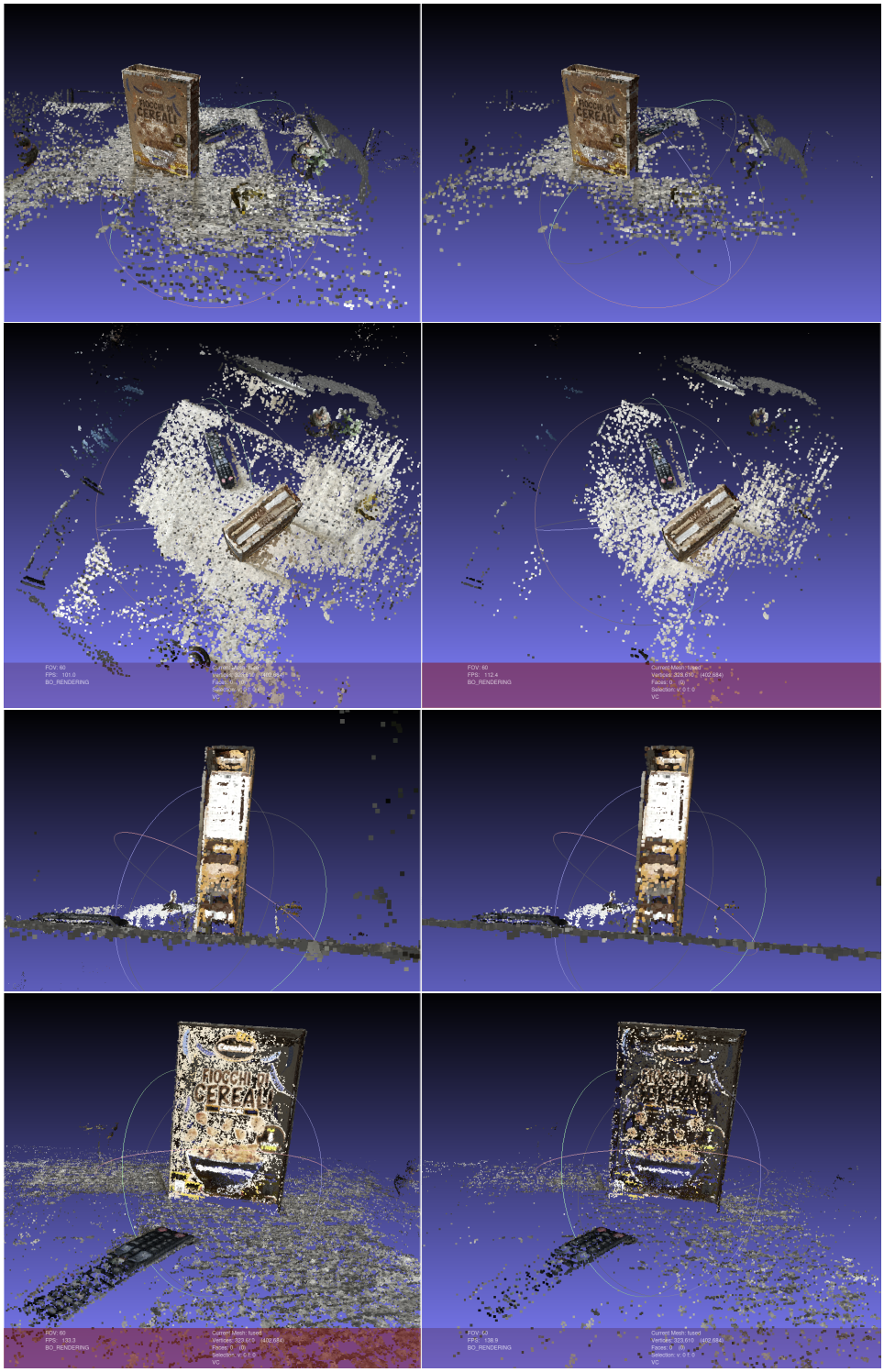


Figure 4.3: Comparing the point clouds generated from a cereal box before(right images) and after refinement(left images)

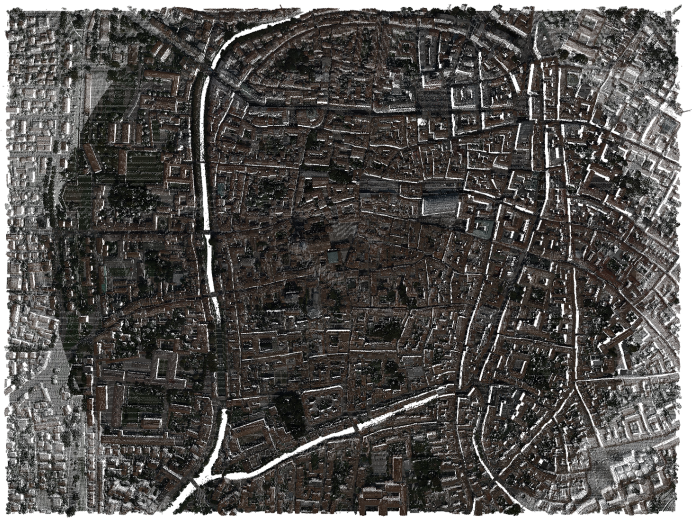


Figure 4.4: The aerial point cloud from the city of Padova

align the point clouds with the z-axis. The aerial point cloud is already aligned. The ground point cloud is segmented into planes, and the plane with the highest point count is identified as the street plane since it is the common plane among all streets and directions and so more points must fall on the ground plane. After that, the rotation matrix between the normal vector of the street plane and the (x=0, y=0, z=1) vector is computed, and then, the ground point cloud is rotated using the rotation matrix. This step determines the rotation around x and y axes, roughly.

2. From the top point of view, It is seen that the most discriminative feature among the datasets is the angles of streets and crossroads. So, the data of vertical walls in ground dataset and roofs in aerial dataset has no use. Therefore, both point clouds are sliced along the z-axis with a certain threshold and the half related to the streets is remained. In figure 4.5, the similarity of both sliced ground and aerial point clouds are visible.

3. Since the generated point clouds has the up-to-scale problem, it is needed to normalize the coordinate values across all datasets. This step does not scale the ground point clouds to the actual size in aerial point cloud. However, it ensures that the average minimum distance between the points is the same among all of the generated point clouds. Each point cloud is scaled by the factor of 1 divided by the current average of the minimum distance to the

(a) Ground Point Cloud (b) Aerial Point Cloud

Figure 4.5: Point clouds sliced from the top viewpoint

nearest point.

4. A 2D binary map is generated for each point cloud considering only x and y coordinates of all 3D points. The dimensions of the map are determined by calculating the difference between the maximum and minimum x and y coordinates of all points and dividing it by a predefined resolution value. Each cell in the map is assigned a value of 1 if at least one point's x and y coordinates fall within that cell, indicating the presence of points in that area. Conversely, if there are no points corresponding to the respective x and y coordinates, the cell is marked as 0. In the binary grid map for Ground Point Cloud, considering that each point cloud is generated from a sequence of video frames, the camera pose of the middle frame is regarded as the ground truth center position for the entire dataset. To evaluate the registration method's robustness and generalization, 3 aerial point clouds are considered for each ground dataset. These aerial point clouds share the same center as the aforementioned ground point cloud, but they differ in distance from the center. They are categorized as "easy," "medium," and "hard" with distances of 100, 150, and 200 meters from each direction, respectively. The binary grid map is generated for these aerial point clouds using the same logic as the binary grid map created for the ground point clouds.

5. To localize the ground grid map within the aerial grid maps, various methods were explored, including 2D feature matching, crossroad detection based on point counting, and training convolutional neural networks, etc. Among these approaches, template matching has the best results. The template matching process involves comparing a small template image, which represents the desired pattern (in this case, the ground grid map), with different regions of the target image (the aerial grid map), and identifying the region that has the highest similarity with the template. This is achieved by mov-

ing a window across the target image and comparing the template with each window. We extended the algorithm to compare also different scales and rotations of the template image. The template is rotated up to 360 degrees with an interval of 10 degrees and scaled from 10% to 200% of its initial size, increasing 10 percent per each attempt.

4.3 Final algorithm

The whole pipeline can be summarized as follows:

1. Take the video of the streets and extract the frames
2. Run sparse SfM algorithm
3. Refine the sparse point cloud and camera poses using Pixel Perfect algorithm
4. Generate a dense point cloud from the refined data
5. Align the point cloud along the z-axis.
6. Scale and Slice the ground point cloud, and keep the points with z coordinate less than a threshold, in our case, the threshold is 20% of the minimum z-coordinate
7. Generate the binary grid map
8. Slice Aerial point cloud along the z-axis with a certain threshold, in our case, the threshold is 20% of the minimum of z coordinate of all points
9. Generate Binary grid map for Aerial point cloud
10. Run Template Matching algorithm to localize the ground grid map inside the aerial grind map

4.4 Other Approaches

Correspondence-based registration is a method used in global point cloud registrations. If common 3D features could be detected, localizing ground point clouds within aerial point clouds would be easier. As it is mentioned before, the only common 3D points between those two point clouds are streets which are not distinctive features. Instead, crossroads are considered more unique. Therefore, we attempted to detect crossroads in both point clouds and register them, directly.

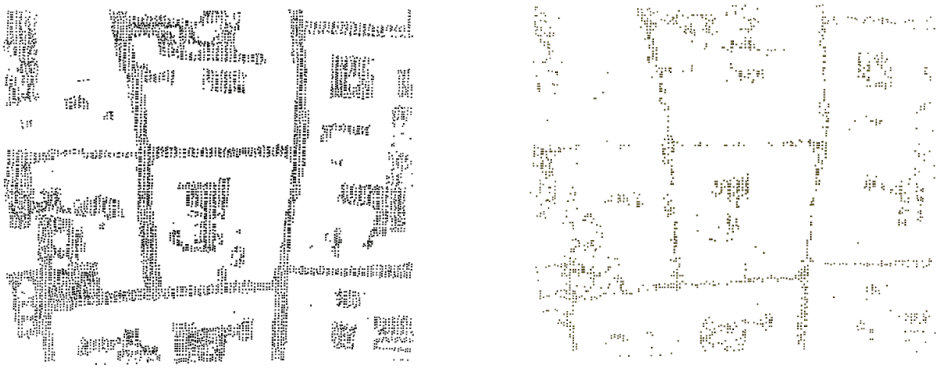


Figure 4.6: Left image: before erosion, Right image: after erosion

Crossroads detection based on handcrafted methods: The areas of crossroads are assumed to have more 3D points than the areas of only streets. Because crossroads represent the meeting point of two or more streets. Hence, one approach for crossroads detection is selecting points with a higher density of neighboring points. However, finding appropriate thresholds for density and the radius to select neighboring points were too challenging due to variations in street width, difficulty in distinguishing between flat grounds (e.g., yards and gardens) and streets, and differing 3D point densities in ground point clouds. We tried to simplify the problem using methods such as uniform point distribution and erosion techniques to remove border points while retaining the points related to the center of streets, potentially indicating crossroads. However, none of these methods provided enough accurate and robust results. Figure 4.6 shows the result of preprocessing methods and figure 4.7 shows the best results of crossroads detection using the aforementioned handcrafted methods.

Another idea involved segmenting the initial ground point cloud, which still contains points from buildings and walls, into planes representing the street and multiple building walls. The intersection of at least two building planes and the street plane could potentially be identified as crossroads. However, this approach lacked accuracy and robustness, particularly in open areas with limited walls. Figure 4.8 shows the ground point clouds segmented into planes of the street and walls

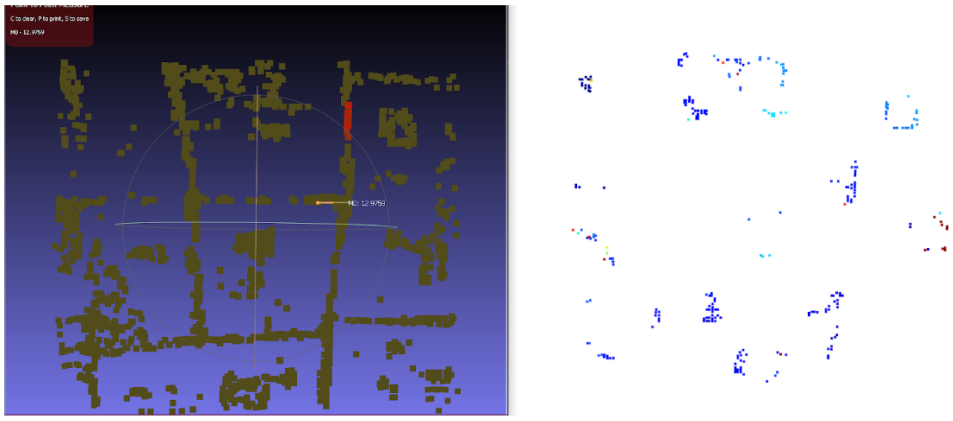


Figure 4.7: Left image: the actual point cloud, Right image: the same point cloud from the same viewpoint with only detected crossroads points using handcrafted methods

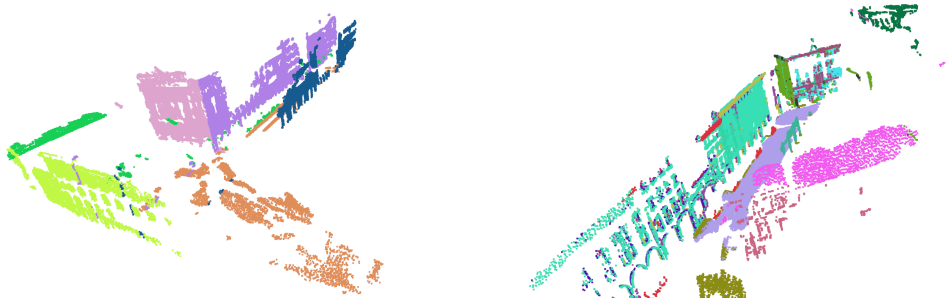
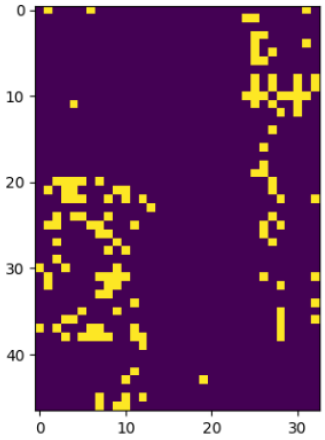
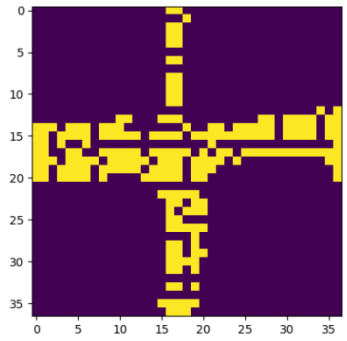


Figure 4.8: Ground point clouds segmented into planes of the street and buildings' walls



(a) Random non-crossroads binary grid map



(b) Random crossroads binary grid map

Figure 4.9: Samples of the CNN inputs

Deep Learning based approach: We also explored a Deep Learning approach, utilizing the binary grid maps introduced earlier, as inputs for a Convolutional Neural Network to classify each 3D point as either crossroads or non-crossroads. Figure 4.9 shows a pair of examples of the binary grid maps representing a top-down view of a crossroad and a non-crossroad. The model was trained by a small dataset containing 10 inputs of binary grid maps per each class. Then, we applied the trained model to all points in a small window of the aerial point cloud and filtered out points classified as non-crossroads. Figure 4.10 illustrates the points identified as crossroads which includes points of the 4 main crossroads out of 5. The results demonstrated excellent performance. However, to ensure robustness, a significantly larger dataset considering the big 1600m*1000m aerial point cloud with approximately 3 million points would be necessary for training.

Figure 4.10: Detected crossroads by CNNs: The red points are classified as crossroads

Chapter 5

Experiment

5.1 Dataset

Aerial Point Cloud: An extensive collection of 3D points is obtained from the city of Padova using an airplane. The covered region spans 1600m by 1000m, and the average nearest neighbor distance of points is 0.63m. The dataset contains a total of 3,583,803 points, figure 4.4.

Ground Point Clouds: 10 dense point clouds are created using the Structure from Motion algorithm. They are further refined by Pixel-Perfect algorithm. Each dataset comprises an average of 60 frames extracted from a 30-second undistorted video. The video was recorded at a frame rate of 60 fps. The resolution of each image is 1920*1080 pixels, and the distance covered in each dataset ranges from 80 to 120 meters.

5.2 Metrics

In our matching problem, there are four key parameters for evaluation:

- the 2D coordinates (x, y) of the detected pose of the center,
- the scale
- the rotation

of the ground binary grid map within the source binary grid map.

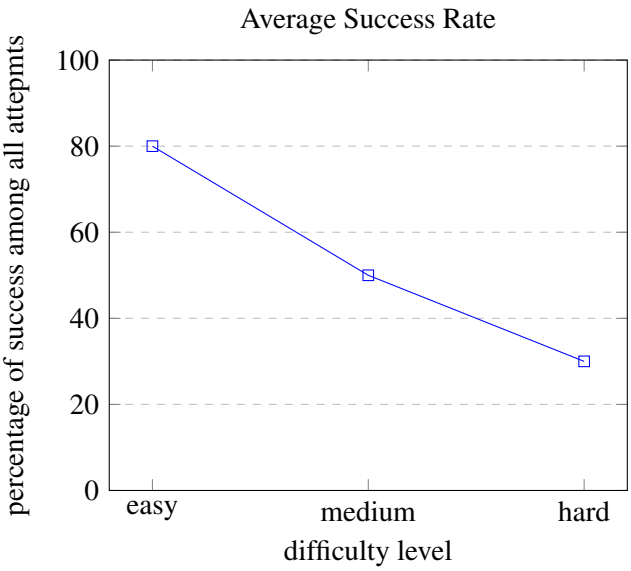


Figure 5.1: Average Success Rate for template matching

The results of the global registration are binary in nature, meaning that whether the template point cloud is successfully found in the source or not. Hence, the first metric used is the Average Success Rate.

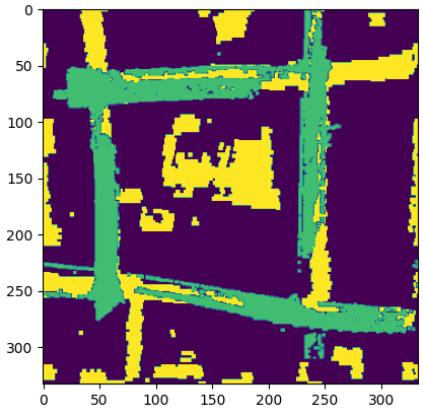
Average Success Rate: For each ground binary map and its corresponding aerial binary map, if the difference between the four key parameters and their ground truth values is less than a certain threshold, it is considered a successful match. In our experiment, the euclidean distance between the center of the ground binary map and the ground truth, i.e. which is the center of each aerial binary grid map, should not be more than 25 meters. An acceptable scale factor is between 80% to 130% of the ground truth. And, the rotation angle should not be more than 10 degrees, otherwise, in most cases, the wrong street is detected.

5.3 Results

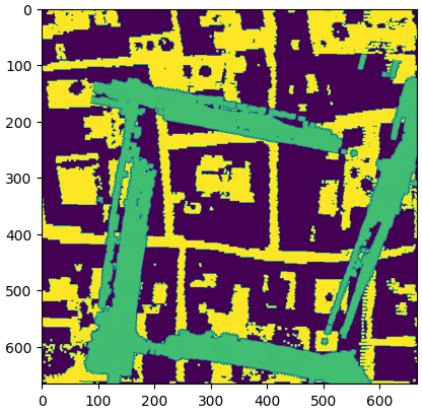
There are 10 ground datasets, and for each dataset, there are three aerial datasets categorized as easy, medium, and hard. Figure 5.1 shows the average success rate, in percentage, for all ground datasets based on the difficulty level of their corresponding aerial datasets.

Here are the final results of template matching. For each ground dataset, 3 images are provided illustrating the result matches in easy, medium, and hard aerial point clouds. The yellow pixels refer to the aerial point cloud(source) and green pixels are related to the ground point cloud(template). The axes of the graph are scaled in meters multiplied by a resolution value. This resolution is either 3.3 or 6.6 and is because of variations in the density of points and for a more accurate representation of the grid maps.

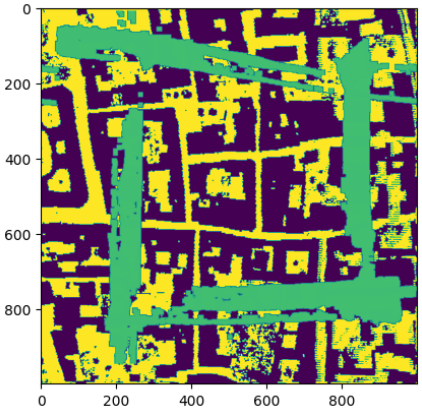
For each ground dataset that has at least one successful result, a table of detailed errors of transformation in meters, i.e. distance between the center of source and template images, scale in percentage, and angle in degrees is provided.



(a) Easy: Success



(b) Medium: Failed

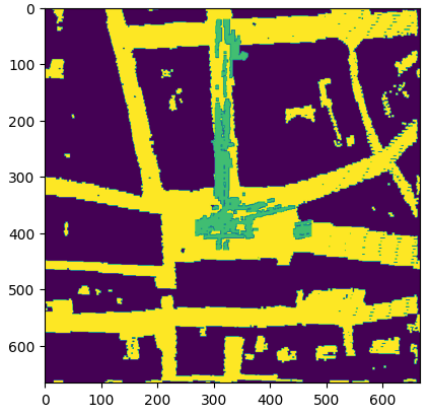


(c) Hard: Failed

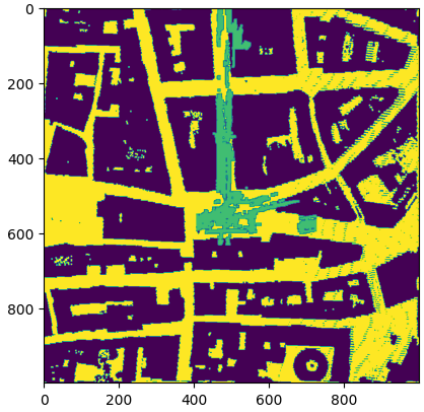
Figure 5.2: Dataset 1

Errors:	**Transformation**	**Scale**	**Rotation**
Easy	0.50m	+3.5%	+2°
Medium	-	-	-
Hard	-	-	-

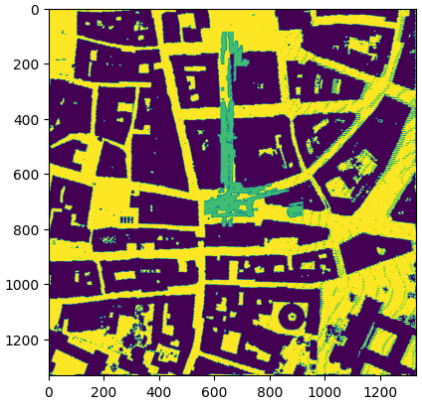
Table 5.1: Detailed errors of Dataset 1



(a) Easy: Success



(b) Medium: Success

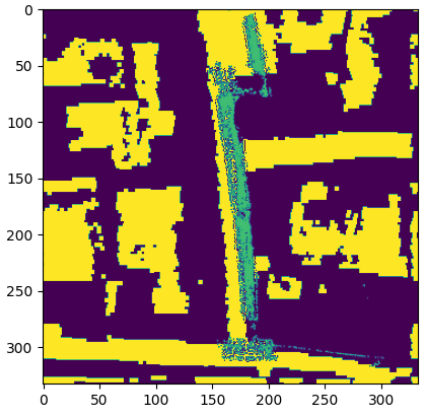


(c) Hard: Success

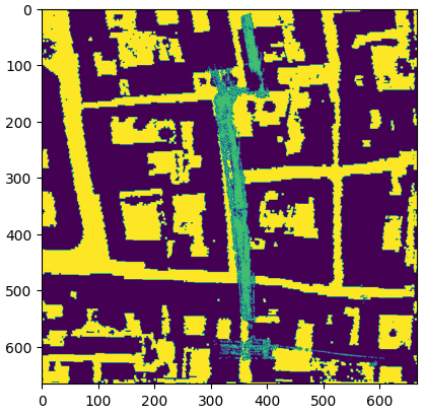
Figure 5.3: Dataset 2

Errors:	Transformation	Scale	Rotation
Easy	3.16m	-8.7%	-3°
Medium	6.66	+7.14%	-3°
Hard	26.6	+16.6%	-3°

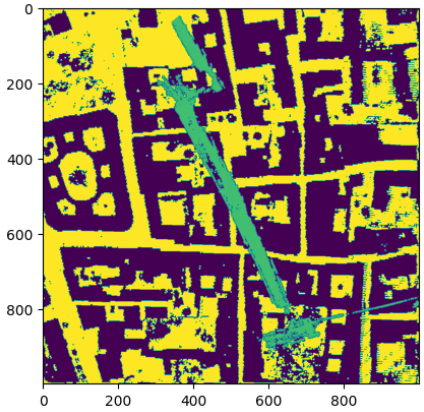
Table 5.2: Detailed errors of Dataset 2



(a) Easy: Success



(b) Medium: Success

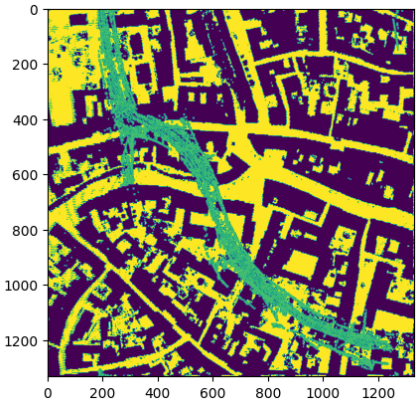


(c) Hard: Failed

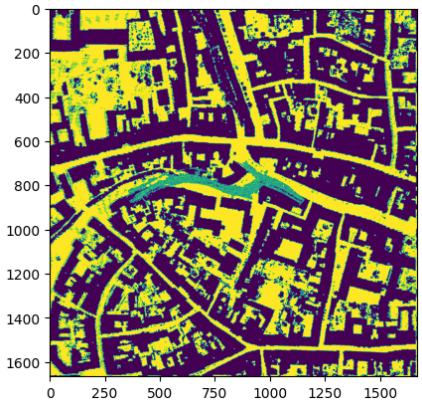
Figure 5.4: Dataset 3

Errors:	**Transformation**	**Scale**	**Rotation**
Easy	1.6m	-4.4%	1°
Medium	4.83m	+13.3%	1°
Hard	-	-	-

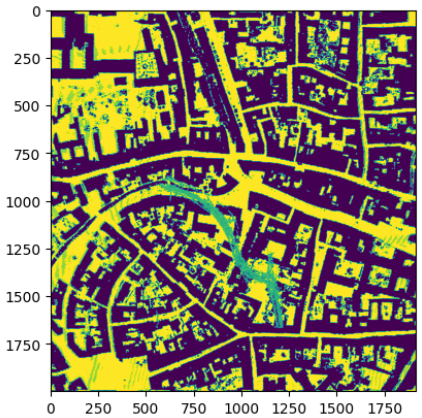
Table 5.3: Detailed errors of Dataset 3



(a) Easy: Failed



(b) Medium: Success

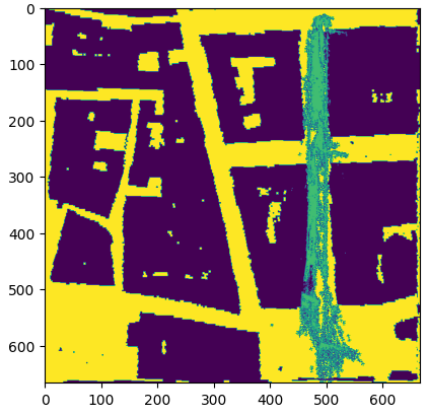


(c) Hard: Failed

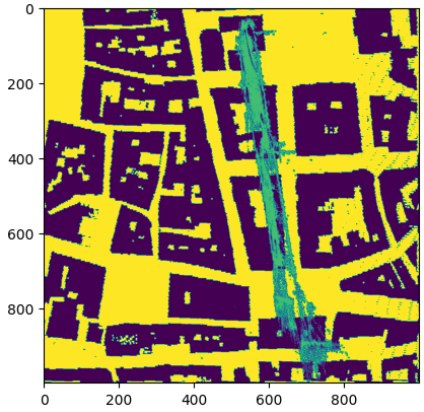
Figure 5.5: Dataset 4

Errors:	Transformation	Scale	Rotation
Easy	-	-	-
Medium	5.62m	-9.11%	-3°
Hard	-	-	-

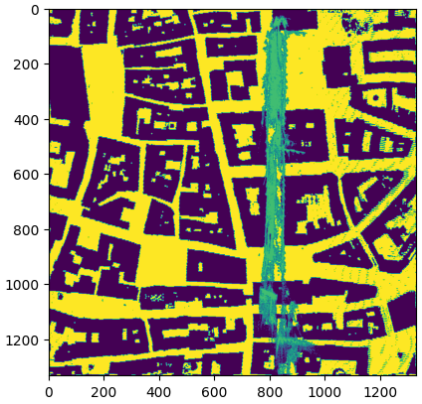
Table 5.4: Detailed errors of Dataset 3



(a) Easy: Success



(b) Medium: Success

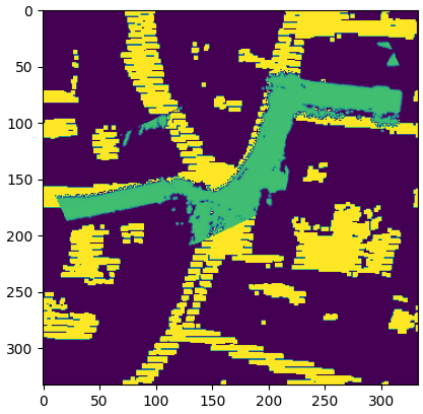


(c) Hard: Success

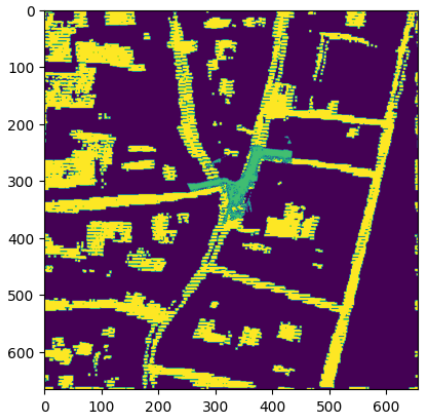
Figure 5.6: Dataset 5

Errors:	Transformation	Scale	Rotation
Easy	¡1m	-9.1%	-3°
Medium	5.44m	+27.2%	+7°
Hard	8.63m	+81.2%	-3°

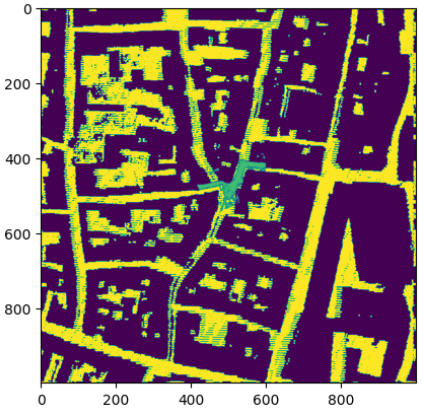
Table 5.5: Detailed errors of Dataset 5



(a) Easy: Success



(b) Medium: Success

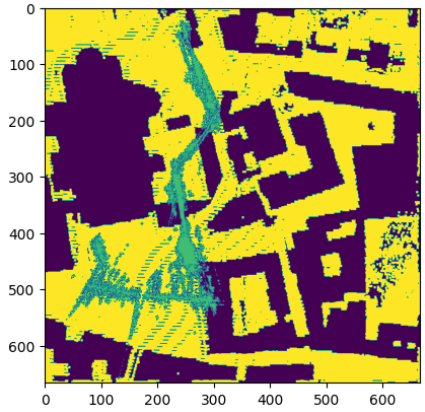


(c) Hard: Success

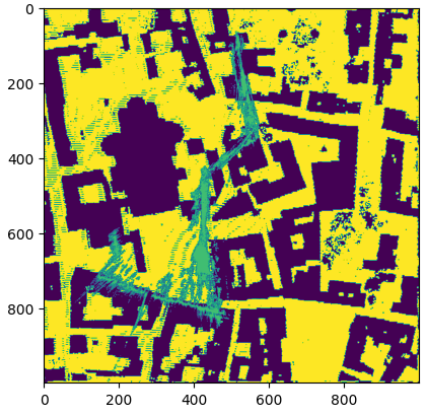
Figure 5.7: Dataset 6

Errors:	**Transformation**	**Scale**	**Rotation**
Easy	1.4m	+6.1%	+3°
Medium	3.6m	+3.3%	+3°
Hard	3.5m	+3.3%	+3°

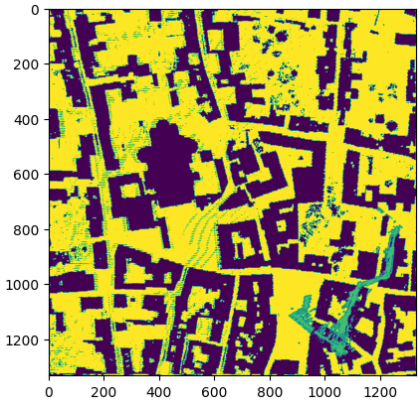
Table 5.6: Detailed errors of Dataset 6



(a) Easy: Success



(b) Medium: Failed

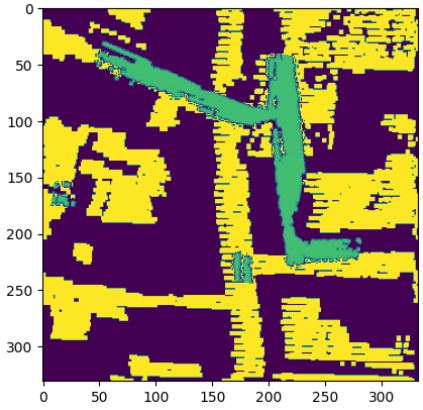


(c) Hard: Failed

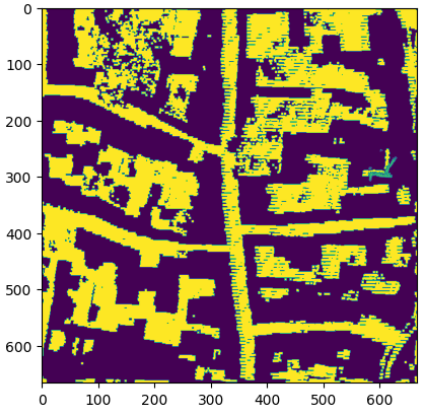
Figure 5.8: Dataset 7

Errors:	Transformation	Scale	Rotation
Easy	34m	-5.9%	+4°
Medium	-	-	-
Hard	-	-	-

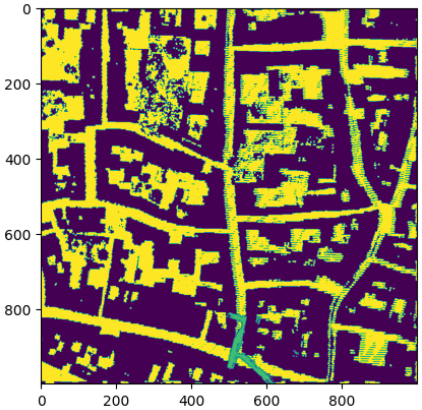
Table 5.7: Detailed errors of Dataset 7



(a) Easy: Success



(b) Medium: Failed

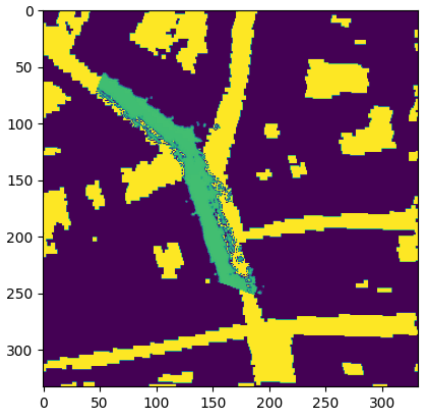


(c) Hard: Failed

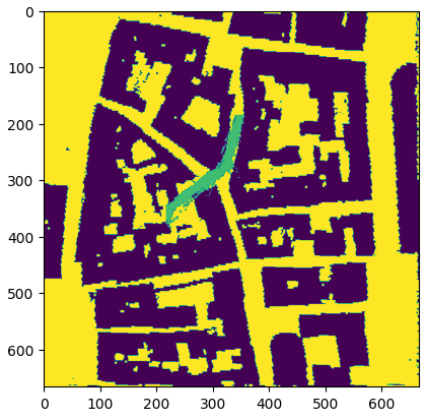
Figure 5.9: Dataset 8

Errors:	**Transformation**	**Scale**	**Rotation**
Easy	17.2m	-5.2%	+1°
Medium	-	-	-
Hard	-	-	-

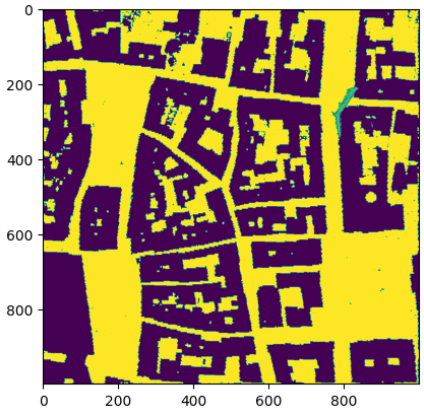
Table 5.8: Detailed errors of Dataset 8



(a) Easy: Success



(b) Medium: Failed

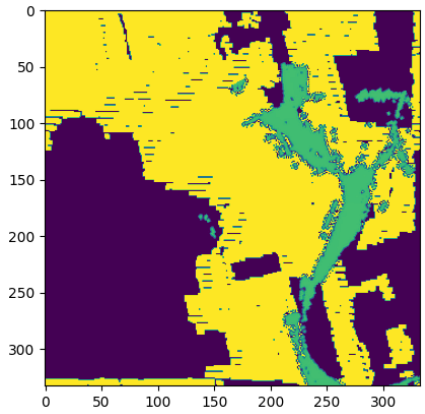


(c) Hard: Failed

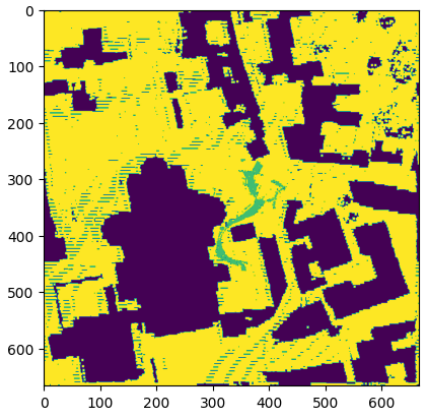
Figure 5.10: Dataset 9

Errors:	Transformation	Scale	Rotation
Easy	¡1m	-1.33%	-3°
Medium	-	-	-
Hard	-	-	-

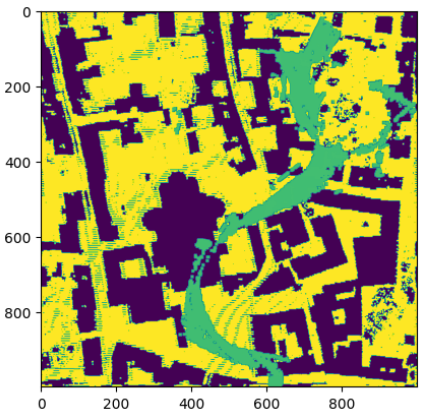
Table 5.9: Detailed errors of Dataset 9



(a) Easy: Failed



(b) Medium: Failed

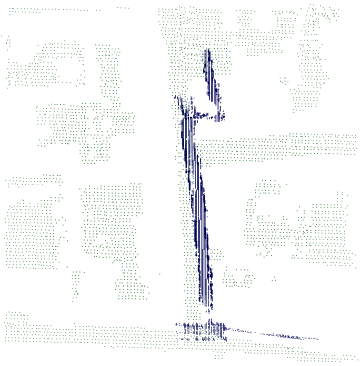


(c) Hard: Failed

Figure 5.11: Dataset 10

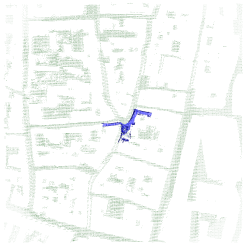
As expected, localizing in wider windows of the Aerial Point Cloud becomes more challenging. Also, in open spaces where 3D points represent flat surfaces and do not precisely shape the street geometry, identifying their pattern in the aerial point cloud is difficult, like in dataset 10. Moreover, localizing streets with simple geometry is unlikely. For instance, if the generated point cloud only represents a straight alley, its binary grid map would be a simple rectangle. Considering the template matching algorithm, a rectangle as the template binary map could be matched with any streets, especially in small scales. We expect that the algorithm would be able to localize more accurately after passing a crossroads or when the street's direction changes.

Next, by mapping the resulting 2D coordinates to the aerial point cloud's reference frame, and applying the scaling and rotation factors, the ground point cloud is located inside the aerial point cloud. Then, since the successful results are close enough to the ground truth, we used Iterative Closest Point (ICP), which a common local point cloud registration algorithm for final refinement. Figure 5.12 shows a few of the best results:



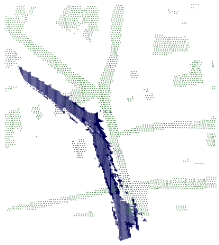
(a) Dataset 3: Before ICP

(b) Dataset 3: After ICP

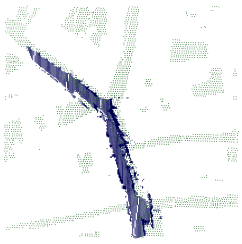


(c) Dataset 6: Before ICP

(d) Dataset 6: After ICP



(e) Dataset 9: Before ICP



(f) Dataset 9: After ICP

Figure 5.12: Moving the template matching results to the aerial point cloud reference frame

(a) Distorted frame

(b) Corrected image using calibration parameters

Figure 5.13: Samples of distorted and corrected video frames

5.4 Reconstruction Details

Calibration and its impact: One of our case studies was if giving wider but distorted images with manual calibration parameters to the SfM pipeline could enhance the accuracy and increase the number of generated point clouds of a wider field of view. We used the camera with 2.7k resolution, 60fps, and wide and distorted settings. The camera was calibrated using the checkerboard method. Figure 5.13 demonstrates an example of undistortion of video frame, and Figure 5.14 illustrates the resulting sparse point clouds. The generated point cloud has a notable error in the street angles, meaning that, from the top point of view, the streets should have been perpendicular to each other. However, the result displays an angle that is less than 90 degrees. Also, the points associated with the ground truth plane are excessively spread out from the generated plane. Furthermore, the execution time experienced an exponential increase. For instance, it took approximately 10 minutes to construct a sparse point cloud from the undistorted frames shown in Figure 5.14 , whereas the same procedure for the distorted frames required more than 2 hours.

Boosting Reconstruction pipeline: The process of Structure from Motion is computationally heavy and not suitable for real-time usage, particularly when it comes to dense reconstruction. Table 5.10 presents the execution time of the sparse reconstruction for each step of the SfM algorithm running on our datasets.

As can be seen, bundle adjustment is the bottleneck. During our experiments, we noticed that providing an initial estimation of camera poses can greatly reduce the number of iterations required for bundle adjustment. Hence, we explored the use of existing visual Simultaneous Localization and Mapping (vSLAM) methods

(a) From distorted frame (b) From undistorted frame

Figure 5.14: Point clouds generated from distorted and undistorted video frames from top point of view

Inputs:	Max num of features	Feature Extraction	Feature Matching	Bundle Adjustment	total
300 frames(1080p)	30000	0.85mins	1.01mins	21.5mins	23.36mins
300 frames	5000	0.15mins	0.23mins	6.74mins	7.12mins
300 frames(Known poses)	30000	0.73mins	1.05mins	1.65mins	3.43mins

Table 5.10: The execution time of the sparse reconstruction for each step of the SfM algorithm using Nvidia RTX3090 graphics

to approximate the initial camera poses. The best solution we found that using COLMAP sparse reconstruction with low configuration parameters, such as lower image resolution and limiting the maximum number of keypoints and matches, just to obtain camera poses. Then, COLMAP's sparse reconstruction is re-executed with the initial camera poses and high-quality configuration. This trick reduced the execution time significantly such that a dataset of 300 frames was reconstructed in 10 minutes.

Chapter 6

Conclusion

We developed a pipeline for the localization of a moving vehicle using images or videos captured by a camera. Our process involved capturing videos of the streets in the city of Padova, extracting video frames, and applying the Structure from Motion algorithm to generate point cloud of streets, called ground point cloud. We used [6] to refine our reconstruction. Furthermore, a few handcrafted methods are applied to the ground point clouds as preprocessing, like aligning on the z-axis, filtering, and slicing. On the other hand, we already had an extensive point cloud captured by an airplane from the whole city as an aerial (source) point cloud. In order to simplify our problem, for each point cloud, a binary grid map is created, where each pixel is set to 1 if at least one 3D point's x and y coordinates fall within that pixel coordinates. By having 2D images of binary maps for both point clouds, a template matching algorithm is utilized to localize the ground binary map in the aerial grid map. We extended the template matching algorithm in order to return not only the coordinates but also the scale and rotation of the template image. Next, we scaled and transformed our initial ground point cloud with these results into the aerial point cloud, and then, the Iterative Closest Point (ICP) algorithm is employed for local point cloud registration, getting even more accurate results. As it is discussed before, it is unlikely to register directly the ground point cloud in the aerial point cloud since the only common 3D points between these two are streets and ground points, and these points are not so discriminative. However, our method showed a reasonable performance in local point cloud registration.

As it is observed in the template matching results, the bigger the window of the aerial point cloud is, the harder it is to localize. Therefore, we may think of another approach for global registration. In our experiments, it is noticed that by

looking at both point clouds from the top viewpoint, crossroads are differentiable. So, if we could detect them as keypoints and provide a proper descriptor for each crossroad, it is possible to register by correspondence the ground point cloud in wider windows of the aerial point cloud. Our deep learning approach showed it is possible to filter the crossroads by using the binary grid maps as the input to our model. However, we had to prepare a huge and comprehensive dataset of crossroads which needed longer time for this thesis.

To conclude, nowadays, with the advancements in cameras and powerful computers, the computation of accurate and real-time 3D data is becoming increasingly feasible. This will lead to a rapid expansion of 3D applications like autonomous driving and augmented reality. Structure from Motion has proven to be a powerful tool for generating accurate and detailed 3D structures. Therefore, improving the accuracy and performance of this algorithm plays an important role in the future.

Bibliography

[1] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009.

[2] D. DeTone, T. Malisiewicz, and A. Rabinovich. Superpoint: Self-supervised interest point detection and description, 2018.

[3] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler. D2-net: A trainable cnn for joint detection and description of local features, 2019.

[4] M. Dusmanu, J. L. Schönberger, and M. Pollefeys. Multi-View Optimization of Local Feature Geometry. In *Proceedings of the European Conference on Computer Vision*, 2020.

[5] J. Engel, T. Schoeps, and D. Cremers. Lsd-slam: large-scale direct monocular slam. volume 8690, pages 1–16, 09 2014.

[6] P. Lindenberger, P.-E. Sarlin, V. Larsson, and M. Pollefeys. Pixel-Perfect Structure-from-Motion with Featuremetric Refinement. In *ICCV*, 2021.

[7] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[8] J. Revaud, P. Weinzaepfel, C. D. Souza, N. Pion, G. Csurka, Y. Cabon, and M. Humenberger. R2d2: Repeatable and reliable detector and descriptor, 2019.

[9] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[11] T. Schöps, T. Sattler, and M. Pollefeys. BAD SLAM: Bundle adjusted direct RGB-D SLAM. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[12] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016.

[13] C. Tang and P. Tan. Ba-net: Dense bundle adjustment network, 2019.

[14] K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction, 2017.

[15] Z. Teed and J. Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras, 2022.

[16] Z. Teed, L. Lipson, and J. Deng. Deep patch visual odometry, 2022.

[17] J. Wang, Y. Zhong, Y. Dai, S. Birchfield, K. Zhang, N. Smolyanskiy, and H. Li. Deep two-view structure-from-motion revisited. *CVPR*, 2021.

[18] J. Wang, Y. Zhong, Y. Dai, K. Zhang, P. Ji, and H. Li. Displacement-invariant matching cost learning for accurate optical flow estimation. *Advances in Neural Information Processing Systems*, 33, 2020.

[19] N. Yang, R. Wang, J. Stückler, and D. Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry, 2018.

[20] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks, 2017.