Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Vision
and Geometry Lab

Semester Thesis

Metric Reconstruction and Optimization

Autumn Term 2022

Supervised by:
Silvan Weder
Mihai Dusmanu

Author:
Haoran Chen

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Your Project Title

is original work which I alone have authored and which is written in my own words.[1]

Author(s)

Haoran	Chen

Student supervisor(s)

Silvan	Weder
Mihai	Dusmanu

Supervising lecturer

Marc	Pollefeys

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/plagiarism-citationetiquette.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Zurich, Sep 26th
Place and date

Haoran Chen
Signature

[1] Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Contents

Abstract

Structure-from-motion is one of the most powerful tools for 3D reconstruction. While many methods have been proposed to improve the quality of 3D reconstructions in terms of accuracy, robustness, completeness and efficiency, there is still a remaining problem of scale ambiguity in Structure-from-Motion. In this project, we reconstruct the 3D structures of neighborhoods and aim to recover the metric with prior knowledge of GPS data when recording the videos. We propose a GPS-based localization method to improve the retrieval efficiency and optimize our reconstructions using Pose-Graph-Optimization. The project is available at `https://github.com/haoranchen1104/gps_based_localization`.

Symbols

Symbols

K	intrinsic matrix
R	rotation matrix
I	identity matrix
T	translation
c	camera
w	world frame
q	pose (the homogeneous matrix 4×4)
p	position
I	image
G	GPS data
N	number n
v	global feature
f	local features
S	score matrix
D	summed distance in MSAC
$\odot$	dot product
O	observation
R	reconstruction
card.	function of cardinality
SG	matching function of SuperGlue
$\mathcal{T}$	trajectory
r	ratial

Indices

$\{X, Y, Z\}_w$	x, y, z axis in the world frame
$\{u, v\}_c$	pixel coordinate in the image plane

Acronyms and Abbreviations

SfM	structure from motion
SIFT	Scale-invariant feature transform

MSAC	M-Estimate Sample Consensus
fps	frame per second

Chapter 1

Introduction

Structure-from-Motion (SfM) is a popular photogrammetric tool to extract 3D information from 2D images, by estimating the motion of cameras corresponding to these images. Such technique can be used to recover 3D structures from unordered image collections [1, 2, 3, 4]. However, an inherent problem of scale ambiguity in SfM results in the metric information loss of 3D structures. Current reconstruction systems [5, 6, 7] follow a sequential pipeline to build 3D structures iteratively. A pipeline in a reconstruction system usually consists of feature extraction and feature matching, camera motion estimation, 3D structure reconstruction via optimization. In this project, we integrate GPS information into the pipeline as prior knowledge to recover the actual scale of 3D structures. With such information, we can also accelerate the image retrieval process in feature matching. In addition, we optimize the pose graph of photos from different sources in a large reconstruction scene.

1.1 Preliminaries of Structure-from-Motion

In this section, we firstly define SfM in math formulation and explain the issue of scale ambiguity. Then we describe different components in the procedure of SfM and introduce some of state-of-the-art learning-based methods.

1.1.1 Math Formulation of SfM

As shown in fig. 1.1, given a list paired points $(\{p^1_{c_1}, p^2_{c_1}, p^3_{c_1}, \dots\}, \{p^1_{c_2}, p^2_{c_2}, p^3_{c_2}, \dots\})$ from two images (c_1, c_2), SfM aims to calculate the transformation (R for rotation, T for translation) between c_1 and c_2. In some problems, the intrinsics (K_1, K_2) are also unknown, which need to be computed simultaneously.

$$\lambda^i_{c_1} \begin{bmatrix} u^i_{c_1} \\ v^i_{c_1} \\ 1 \end{bmatrix} = K_{c_1} \, [I|0] \begin{bmatrix} X^i_w \\ Y^i_w \\ Z^i_w \\ 1 \end{bmatrix} \tag{1.1}$$

$$\lambda^i_{c_2} \begin{bmatrix} u^i_{c_2} \\ v^i_{c_2} \\ 1 \end{bmatrix} = K_{c_2} \, [R|T] \begin{bmatrix} X^i_w \\ Y^i_w \\ Z^i_w \\ 1 \end{bmatrix} \tag{1.2}$$

The math formulations are shown in eq. 1.1, eq. 1.2, while u^i_*, v^i_* are the pixel coordinates in the image planes. X^i_w, Y^i_w, Z^i_w are the corresponding 3D points in the world frame (in this case, the world frame is the same as the frame of c_1).

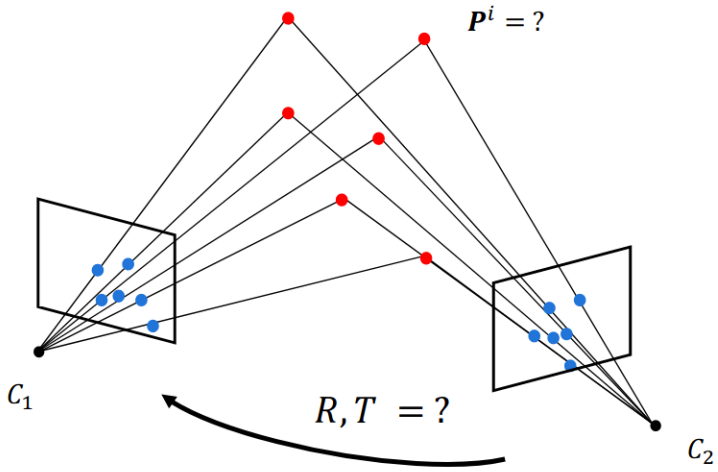


Figure 1.1: Illustration of SfM[1]. The 3D points are shown in red, and the blue points are the corresponding projections on the image plane. SfM estimates the camera poses of new images according to the 3D structure and augments the 3D structure by triangulating new 3D points.

Fig. 1.2 shows that 3D structures of different scales share the same projections on the image plane. In the process of SfM, scale is not obtained during the motion of cameras, but the rotation and the direction of translation can be determined by solving the math equations. Typical factorization-based methods, like eight-point algorithm [8] and five-point algorithm [9], are normally used to calculate the poses of cameras in the SfM problem.

1.1.2 Components of SfM

Feature extraction is the first step in SfM by detecting distinctive keypoints in every image and building corresponding descriptors. These keypoints and descriptors should be repeatable and invariant under certain geometric and photometric changes, such as rotation, view-point change, illumination, etc. One of the standard feature extraction algorithms is SIFT [10], which is mostly used in SfM due to its robustness. Many derivatives of SIFT [11], such as SURF [12], ORB [13], etc, have been developed to improve feature extraction in terms of quantity, accuracy and efficiency. Recently, learned features [14, 15, 16] have been proposed to improve the extraction speed and robustness, and have been the gold standard in SfM.
Feature Matching locate the mutual scene across different images by calculating the correlation between the features, as known as similarity metrics. Typical similarity metrices include Normalized Cross Correlation, Sum of Squared Distance, Sum of Absolute Distance and Census Transform. Different matching strategies are used in different SfM systems in terms of computational complexity, accuracy and robustness. A naive approach is to verify all image pairs to detect scene overlaps. However, large volume of false image pairs take too much computation resources and are time wasted. Thus, other strategies reduce the number of image pairs with image retrieval scheme by comparing the global feature of images and only testing image pairs that have similar global features. Vector of Locally Aggregated Descriptors (VLAD) [17] is usually used as global feature.

[1]The picture is selected from the course slides of "Vision Algorithms for Mobile Robotics" by Prof. Dr. Davide Scaramuzza.

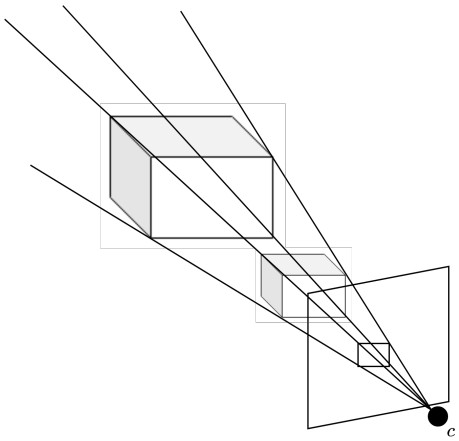


Figure 1.2: Illustration of scale ambiguity. The image shows that the large box and the small box share the same projection on the image plane. Thus, only with the information from images, SfM cannot recover the metric of 3D structures.

Camera motion estimation mainly calculates a projective geometry between images by estimating transformations which map the features from image pairs. Camera motion is estimated by computing the essential matrix E (or the fundamental matrix F) through epipolar geometry [18]. Normally, there are false matched features which are called outliers and can largely influence the estimation result. To alleviate the outlier-contamination, random sample consensus (RANSAC) [19] is often required to verify the geometric relation between matched features, which can improve the robustness of SfM especially in large scenes.
3D structure reconstruction can be recovered with the information of camera motion by triangulating new scene points incrementally. The scene points are crucial in SfM, as they are used to register new images and triangulate new scene points, thus greatly affect the quality the reconstruction result. Image registration and triangulation are coupled, and inevitably cause errors even though using RANSAC to ensure the local accuracy. Thus, the errors are propagated during the incremental reconstruction and can lead to large drift till a meaningless unstable state. Therefore, further refinement is introduced in most SfM systems for optimization. Bundle Adjustment (BA) [20] and pose graph optimization (PGO) [21] are the two most common tools, by minimizing the reprojection error and optimizing the pose graph.

1.2 Large Scene Reconstruction

In this section, we firstly introduce existing methods for large scene reconstruction. Then, we summarize the key steps.
Cohen et al. [22] merges visually disconnected SfM models by selecting connection point pairs with a loop closure constraint. However, this method can only be used in certain conditions and cannot be generalized to large scenes. Kühner and Kümmerle [23] proposes a pipeline for volumetric reconstruction by utilizing LiDAR sensors, which are commonly used in autonomous cars. Fang et al. [24] provides a traditional way of merging different SfM models. The authors detect the image pairs from multiple partial 3D reconstructions, and find the best transformation and the relative scale factor. With the best transformation, the large reconstruction can be built by combining camera poses and 3D point clouds.

For visual-only systems, the key steps include retrieving image pairs from different models, finding the relative transformation according the retrieved pairs, merging and optimizing the models based on the interrelationships.

Chapter 2

Method

Our SfM method for large scene reconstructions is developed in Python using learned-features. In this chapter, we firstly describe our method with an overview, and then explain the details in different components.

2.1 Overview

The project is divided into two phases, data capturing and scene reconstruction. In the first phase, all data are captured by GoPro7 [1], which can store the GPS information while recording videos. Therefore, we can utilize the GPS data to recover the metric of reconstructions. As shown in fig. 2.1, we mounted 5 GoPros on a helmet to capture all angles. Then, with this equipment, we recorded a whole neighborhood through N 3-4 min trajectories, while every trajectory was captured in 5 camera views and shared mutual scenes across other trajectories. Thus, the neighborhood was fully recorded in $5N$ videos, which were then sampled at 5fps to form a database of total $25N$ images.
In the second phase, we developed our method based on hloc [25]. As shown in fig. 2.2, our method firstly builds partial models for all videos, then we merge these models through the mutual scenes shared in different videos. Finally, we optimize the large scene reconstruction using Pose Graph Optimization. Different from hloc, our method recovers the metric of the whole scene, applies GPS data to accelerate the process of image retrieval, and improves the localization robustness by RANSAC.

2.2 Partial Reconstruction

In this section, we implement learning-based methods in [25] to reconstruct each individual trajectory for all camera views. This procedure includes feature extraction, feature matching, camera motion estimation and 3D structure reconstruction.
For feature extraction, we use SuperPoint [16] to extract local features which are generated from CNNs to provide a robust representation for matching in the later steps. Other than tranditional hand-craft descriptors like SIFT, SuperPoint simultaneously computes key point locations and associated descriptors, and thus uses less computational resources and time consumption.
For feature matching, we use SuperGlue [26] to assign features accurately by reasoning about the underlying 3D structure. The attention mechanism in SuperGlue

[1] https://gopro.com/en/us/update/hero7-black

Figure 2.1: Equipment of data capturing.

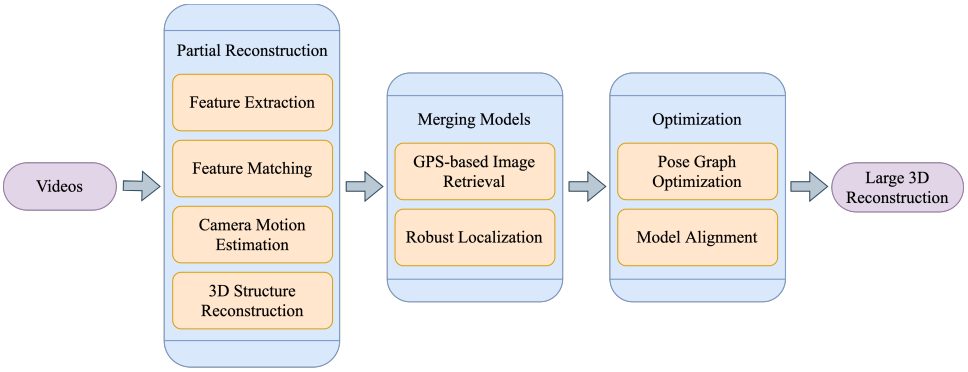


Figure 2.2: An overview of the method. There are mainly 3 steps to build a large scene reconstruction from recorded videos. We use SfM to build partial reconstruction of each video. Then, we merge all partial reconstructions by image retrieval and localization. We also utilize the GPS data to accelerate this step and make it robust. Finally, we optimize all camera poses with PGO and visualize the 3D structure.

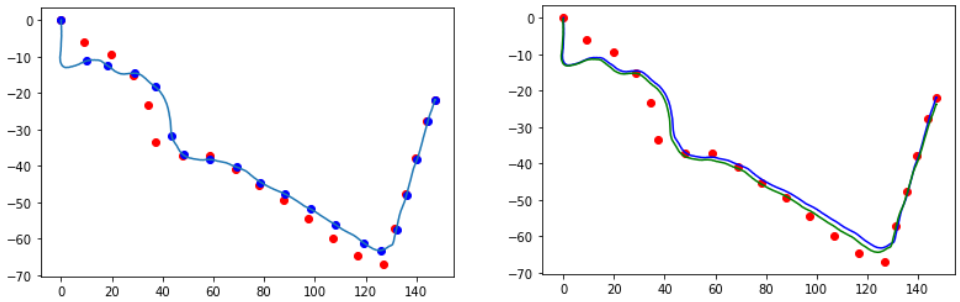
can successfully disambiguate mismatches and is vital for robust matching [26]. Moreover, it can run in real time (70ms) on GPU for an image pair.

Before feature matching, we also have to find image pairs. A naive way is compare all image pairs: for N_i images, there are $\frac{N_i(N_i-1)}{2}$ image pairs. In large scene reconstruction, the number of images in each video is up to 3000, while the number of all image pairs is around 4.5 million, which takes around 4 days. Thus, we use NetVLAD [27] to only consider image pairs that have similar global features. In our project, we extract 5 image pairs for every image, which only takes around half an hour for matching.

For camera motion estimation and 3D structure reconstruction, we use tools in Colmap [5] with its python bindings - pycolmap [2]. The tools firstly verify the geometric relations, then register new images for triangulation in the next step. At last, colmap runs bundle adjustment to refine the 3D model.

Finally, we try to rescale the reconstruction model to the actual scale with the GPS data we have. In this step, we firstly align the GPS data timestamp with every

[2] https://github.com/colmap/pycolmap



(a) Red dots: the recorded GPS data. Blue line: the trajectory of the original model. Blue dots: the corresponding positions in the model.

(b) Red dots: the recorded GPS data. Blue line: the trajectory of the original model. Green line: the recovered trajectory after GPS alignment.

Figure 2.3: An example of model alignment with GPS.

image. Then we apply the model aligner in colmap to recover the model metric. Fig. 2.3 shows an example of how to align the reconstructed model with the GPS data to recover its own metric.

2.3 Merging Partial Reconstructions

In this section, we aim to merge all partial reconstructions that share the same scenes. There are mainly two steps for this part. Firstly, we randomly select two models as the base model and the query model. Secondly, we extract possible image pairs from the base and query models that have similar global features. Thirdly, we try to localize the images in the pairs that are selected in the query model. Finally, we utilize RANSAC to eliminate the outliers after localization.

2.3.1 GPS-based Image Retrieval

For two partial observations $\mathrm{O}_A = \{\mathrm{I}_1^A, \mathrm{I}_2^A, \mathrm{I}_3^A, \ldots, \mathrm{I}_{N_A}^A\}, \mathrm{O}_B = \{\mathrm{I}_1^B, \mathrm{I}_2^B, \mathrm{I}_3^B, \ldots, \mathrm{I}_{N_B}^B\}$, it is common that A, B only share a same segment of scene in their reconstructed models. It's not reasonable to naively extract all image pairs though the two images are apparently taken in different locations.

Therefore, with the GPS information, we can determine whether an image pair $(\mathrm{I}_i^A, \mathrm{I}_j^B)$ is valid or not based on their corresponding GPS data $\mathrm{G}_i^A, \mathrm{G}_j^B$. We assume the image pair $(\mathrm{I}_i^A, \mathrm{I}_j^B)$ is invalid if $\|\mathrm{G}_i^A - \mathrm{G}_j^B\| > 10m$. For simplicity, we divide the whole map into $12 \times 12m^2$ grids, while the centers of these grids are only $10m$ away from each other to have enough overlaps with neighboring grids. We only consider image pairs that are located in the same grid. In this way, we can include most of image pairs that are less then $10m$ away, and the whole image retrieval process can be computed very quickly.

With the GPS-based image retrieval strategy, we can avoid many false image pairs that do not share a same scene but have similar patterns in the image plane, like walls, green grass, etc. In addition, the new strategy takes much less time. The time consumption of the naive retrieval strategy is proportional to $N_A N_B$, however, the new strategy only depends on the number of images in one grid and the number of grids. Therefore, the larger the partial reconstructions are, the more time the GPS-based strategy can save.

Fig. 2.4 shows an example GPS database of images. We adapt our algorithm

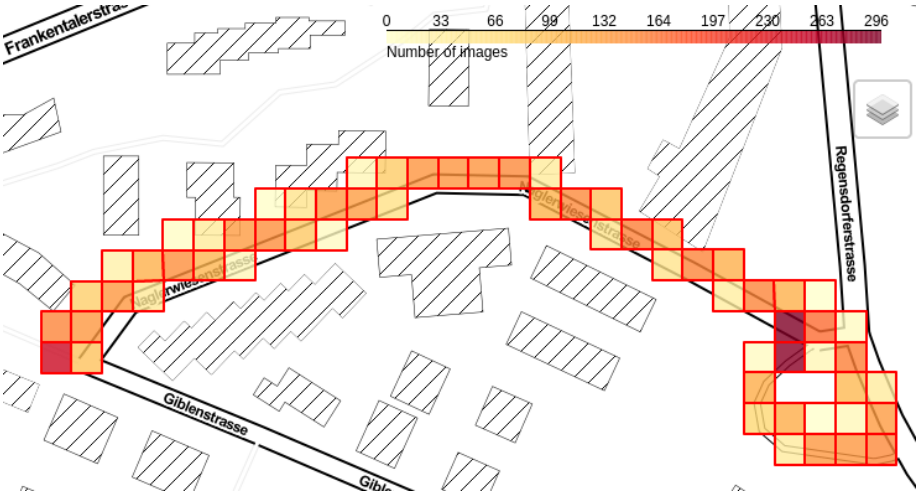


Figure 2.4: The database of GPS information for a single trajectory.

from hloc [3] [25] to find valid image pairs between the query and the base model. We directly use the global features of both models, i.e. $V_A = [v_1^A, v_2^A, v_3^A, \ldots, v_{N_A}^A]$, $V_B = [v_1^B, v_2^B, v_3^B, \ldots, v_{N_B}^B]$, calculated using NetVLAD during partial reconstruction. We compute the score matrix S as below eq. 2.1:

$$S = \begin{bmatrix} v_1^A \odot v_1^B & v_1^A \odot v_2^B & \ldots & v_1^A \odot v_{N_B}^B \\ v_2^A \odot v_1^B & v_2^A \odot v_2^B & \ldots & v_2^A \odot v_{N_B}^B \\ \vdots & \vdots & \ddots & \vdots \\ v_{N_A}^A \odot v_1^B & v_{N_A}^A \odot v_2^B & \ldots & v_{N_A}^A \odot v_{N_B}^B \end{bmatrix} \tag{2.1}$$

where $\odot$ represents dot product, while S_{ij} represents the similarity of v_i^A and v_j^B. Then, we apply a GPS mask M_{GPS} to ignore pairs that are not from a same grid.

$$S_{GPS} = S \otimes M_{GPS} \tag{2.2}$$

where $\otimes$ represents Hadamard product, M_{GPS} is the GPS mask, and $\mathbb{I}(*)$ represents the indicator function.

$$M_{GPS} = \begin{bmatrix} \mathbb{I}(d_{th} - d_{11}) & \mathbb{I}(d_{th} - d_{12}) & \ldots & \mathbb{I}(d_{th} - d_{1N_B}) \\ \mathbb{I}(d_{th} - d_{21}) & \mathbb{I}(d_{th} - d_{22}) & \ldots & \mathbb{I}(d_{th} - d_{2N_B}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{I}(d_{th} - d_{N_A1}) & \mathbb{I}(d_{th} - d_{N_A2}) & \ldots & \mathbb{I}(d_{th} - d_{N_AN_B}) \end{bmatrix} \tag{2.3}$$

$$\mathbb{I}(x) = \begin{cases} 1 & \text{if } x >= 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{2.4}$$

$$d_{ij} = \|\mathrm{G}_i^A - \mathrm{G}_j^B\| \tag{2.5}$$

Finally, for every query descriptor in A, we select 5 pairs from S_{GPS} that have the highest scores if exist.

[3] https://github.com/cvg/Hierarchical-Localization/blob/master/hloc/pairs_from_retrieval.py

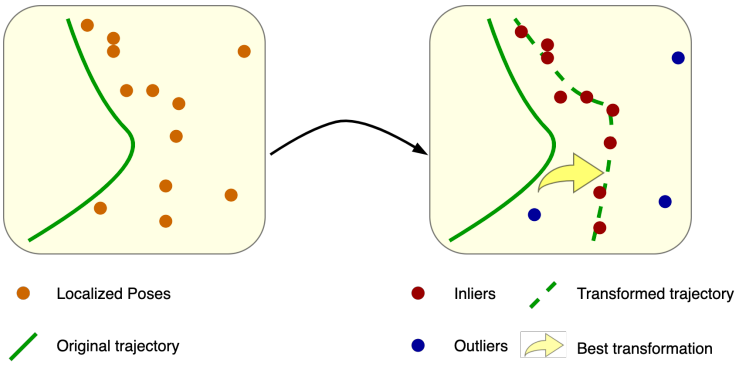


Figure 2.5: The illustration of our RANSAC algorithm. Left: green line - the original trajectory of the query model; orange dots - the localized poses. Right: red dots - inliers; blue dots - outliers; yellow arrow - the best transformation.

2.3.2 Robust Localization

To localize the query model in the frame of base model, we need not only the image pairs but also the matches on the feature level. With the image pairs $\{(\mathrm{I}_i^A, \mathrm{I}_j^B)\}_{i,j}$ retrieved from partial reconstructions $\mathrm{R}_A, \mathrm{R}_B$, we use SuperGlue to match the local features,i.e. SuperPoint f_i^A, f_j^B, in $(\mathrm{I}_i^A, \mathrm{I}_j^B)$.

After computing the relations on the local level, we can estimate the poses of cameras in A by solving the PnP problem with tools in colmap. However, it is inevitable that there are many outliers of the localized poses. We design a RANSAC algorithm to alleviate the outlier-contamination.

As shown in fig. 2.5, the RANSAC algorithm aims to model the best 1-point correspondence transformation that would have the most inliers. The 1-point correspondence transformation $q_{i_{BA}}$ is calculated by one camera pose q_i^A in the query model and its corresponding localized pose $q_i^{L_A}$ as shown in e.q. 2.6.

$$q_{i_{BA}} = q_i^{L_A} q_i^{A^{-1}} \tag{2.6}$$

where L_A represents the localization of camera pose c_i^A in B. After computing the transformation, we evaluate all localized poses with a distance metric. If the distance between the localized position with the transformed position is less than a threshold, then we consider this localized pose is an inlier, otherwise an outlier. However, it is hard to set the threshold for all reconstruction models, which largely affects the result. Therefore, we improve the metric function using MSAC [28] by comparing the summed distance D to find the best transformation.

$$D_i = \sum_j \max\left\{\mathrm{abs}(q_{i_{BA}} p_j^A - p_j^{L_A}), d_{th}\right\} \tag{2.7}$$

Eq. 2.7 shows how to calculate the summed distance, where D_i represents the summed distance with the transformation $q_{i_{BA}}$, p the homogeneous coordinates of camera positions, j represents the index of poses that are localized and d_{th} represents a relatively large distance threshold. In this way, the threshold affects less to the final result. To determine the inliers and outliers, we use a heuristic of $5.0m$. If $\mathrm{abs}(q_{BA}^* p_j^A - p_j^{L_A}) < 5.0m$ under the best transformation q_{BA}^*, we assume the localized pose $p_j^{L_A}$ is an inlier, otherwise an outlier.

2.4 Pose Graph Optimization

After merging the partial reconstructions, we have many query-base localization pairs. However, it's hard to set the initial base model because we cannot know a completely accurate reconstruction as our prior knowledge. Instead, we try to obtain the accurate large scene reconstruction through an optimization problem by taking these query-base localization pairs as cross-constraints between different partial reconstructions. In addition, we apply self-constraints which can be extracted directly from partial reconstructions. The optimization problem is solved using the PGO tool in the python binding (PyCeres [4]) of Ceres [29].

Cross-Constraints are set between different partial reconstructions based on the relations of localization. For every localized inlier $q_i^{L_A}$ in the query model, we identify the image $\mathrm{I}^B_{m_i^A}$ in the base model that has the most feature matches, as shown in eq. 2.8.

$$m_i^A = \mathrm{argmax}_j \left\{ \mathrm{card} \left(\mathrm{SG}(f_i^A, f_j^B) \right) \right\} \tag{2.8}$$

where m_i^A represents the index of which the image in B that has the most matches with I_i^A, card represents the function of cardinality and SG represents the matching function of SuperGlue. Then, we compute the transformation between $q_i^{L_A}$ and $q^B_{m_i^A}$, and set the transformation as the relative cost in the pose graph. We call these constraints as localization constraints.

Self-Constraints are set within each partial reconstruction to fully utilize the information of visual odometry and GPS data. All images are sampled sequentially from the videos. Thus adjacent images usually capture similar objects and therefore have strong relationships. The transformation of two local poses are reliable, and should be used to constrain the partial reconstruction to have reasonable local accuracy. We call these constraints as odometry constraints. Apart from odometry constraints, we would like to consider the information of GPS data. Since the partial reconstructions are already aligned by the colmap model aligner, we only have to constrain all poses to stay as close as to their original ones. We call these constraints as GPS constraints.

After specifying all constraints, we determine the variance of these constraints in the optimization problem. However, the variance is also designed by heuristic. As for cross-constraints, the localization results filtered by RANSAC can be seen as relatively reliable. Thus, in the optimization, we set the uncertainty to $1.0m$ as relative cost. As for self-constraints, we notice that the GPS data sometimes deviate far from the accurate location, which would make the model drift a lot. Therefore, we set large uncertainty $10.0m$ on the GPS constraints as absolute cost. For the odometry constraints, the relation between neighboring poses have already been optimized during partial reconstruction. Consequently, we set the uncertainty to only 5.0% of the distance between neighboring poses as relative cost.

In addition, we determine the loss functions in PGO. With the same consideration above, we use the trivial loss for the odometry constraints, and use Cauchy loss for the localization constraints and the GPS constraints.

[4] https://github.com/cvg/pyceres

Chapter 3

Results

In this chapter, we will show large scene reconstruction result of a whole neighborhood. Total 9 trajectories with 5 camera views were recorded by GoPro 7. Firstly, we present the results of partial reconstructions from 45 videos. Then, we show our GPS-based retrieval and robust localization result by comparing it with the result using the original strategy. Finally, we investigate into the heuristics in the PGO result.

3.1 Partial Reconstruction

Table 3.1: Reconstruction results.

	c_1	c_2	c_3	c_4	c_5
$\mathcal{T}_1$	s	s	s	s	p
$\mathcal{T}_2$	s	s	s	p	p
$\mathcal{T}_3$	f	s	s	p	s
$\mathcal{T}_4$	s	s	s	s	p
$\mathcal{T}_5$	s	p	p	s	s
$\mathcal{T}_6$	s	s	s	s	s
$\mathcal{T}_7$	s	p	p	s	p
$\mathcal{T}_8$	s	p	s	s	s
$\mathcal{T}_9$	s	p	s	s	p

Table 3.1 shows an overview of all partial reconstructions, where $\mathcal{T}$ represents a trajectory, c_i represents camera i, s represents successful reconstruction, p represents a trajectory is only partly reconstructed, and f represents a failed reconstruction. Fig. 3.1 shows successful complete reconstructions of different trajectories. As shown in fig. 3.2, all trajectories share common scenes with each other. Based on these GPS data, we then use the model aligner in colmap to rescale all reconstruction results, of which the 2D visualization is similar to the GPS data shown in fig. 3.2. We also investigate into the reasons of some failed reconstructions. Table 3.1 shows the results when the sample rate is 5 fps, and the success rate of complete reconstruction is 69.9%. Fig. 3.3 shows the possible reasons that might cause the failed reconstructions. These reasons could be avoided by increasing the sample rate of videos. And this idea is proved by the lower success rate 44.4% when we at first sampled the videos at 3 fps. Thus, increasing the sample rate can improve colmap reconstruction by showing similar scenes more frequently. However, more images can take more time for feature extraction and matching. Therefore, we stick to

Figure 3.1: Successful reconstructions of all trajectories. 69.9% of them are reconstructed with completeness. Different camera views may have different influence on the reconstruction according to the environments. In general, it's relatively easy to reconstruct a model with more texture features captured by the camera.

Figure 3.2: GPS data of all trajectories shown in a map. All the GPS data are decoded from the corresponding video files. Thus, there are total 45 GPS files recording the 3D movement (latitude, longitude, elevation) of cameras. This figure only visualizes the GPS information of camera c_1.

Figure 3.3: Possible reasons for failed reconstructions. Large turning may cause the adjacent images do not share enough common features, and thus make SfM fail to estimate new camera poses or augment new 3D structure. Similar pattern may cause SfM to fail, especially in close scene. Many neighboring images have similar features, which make it hard to output correct estimation of new camera poses. Exposure decreases the number of accurate matches between features, therefore make SfM fail to track the camera.

sample rate of 5 fps to achieve a relative high success rate while not increasing the time of computation too much.

3.2 GPS-based retrieval

Table 3.2: Number of pairs retrieved.

	GPS-based	Original
Total N of pairs	2225678	13414100

Before performing GPS-based retrieval, we firstly build the database of GPS data for all images after aligning all partial reconstructions. We extract all camera locations and store them together to form a database, as shown in fig. 3.4.
With the GPS database, we can perform the GPS-based retrieval algorithm. As

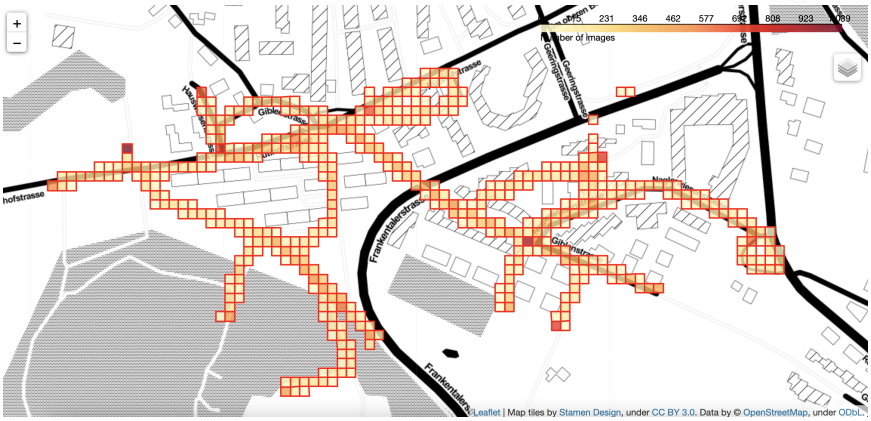


Figure 3.4: The GPS database of all images.

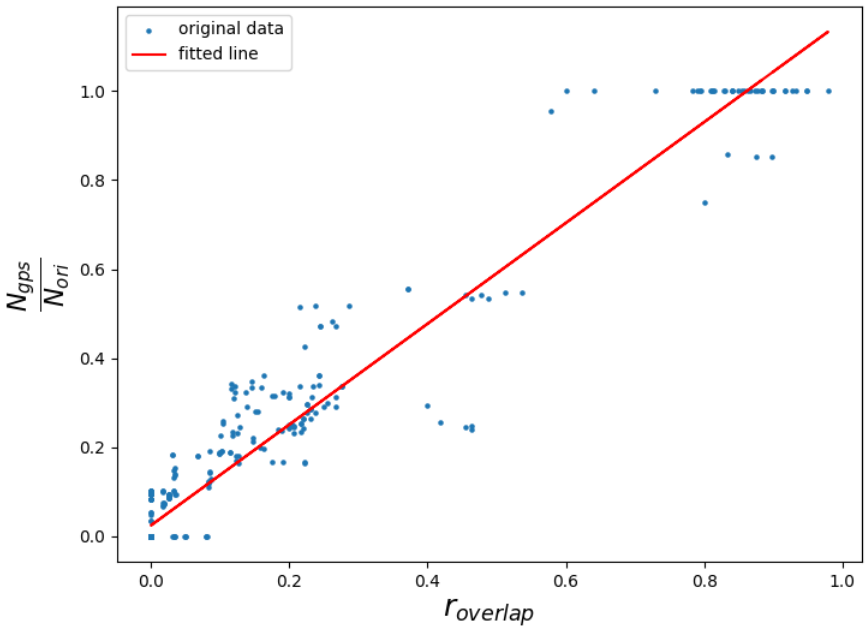
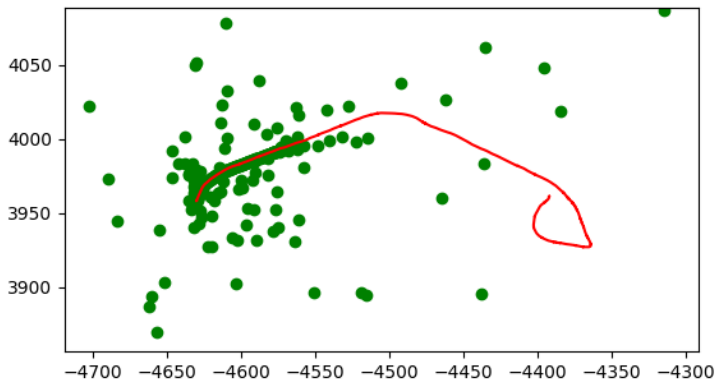


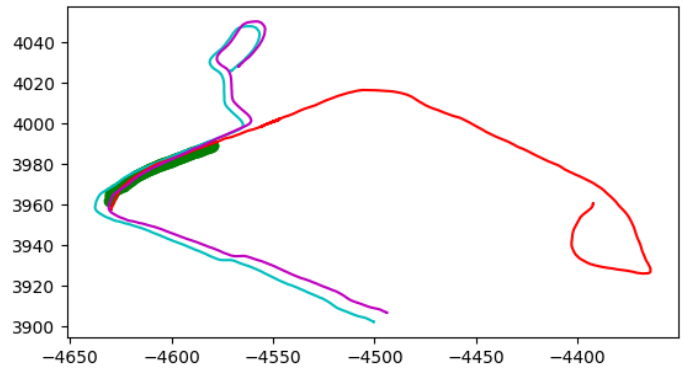
Figure 3.5: The ratio of pairs against the overlap ratio between trajectories. The y-axis is $\frac{N_{gps}}{N_{ori}}$, the ratio of the number of retrieved pairs by two methods. The x-axis is $r_{overlap}$, the overlap rate of two trajectories.

shown in table 3.2, the GPS-based retrieval algorithm only generates 16.6% of pairs that are generated by the original algorithm. For one feature matching of a pair, SuperGlue takes about 70ms. Thus, with the GPS-based retrieval algorithm, feature matching takes 43.3h in total, while it would take around 260.8h using the original method. In addition, GPS information helps to avoid many false matches and improve the accuracy in the later robust localization.

As shown in fig. 3.5, we also investigate into the relation between the ratio of the number of retrieved pairs by two methods and the overlap rate of two trajectories, while the overlap rate is calculated as



(a) 2D plot of localization. The base model is $\mathcal{T}_1c_2$, while the query model is $\mathcal{T}_2c_3$. Red line: 2D plot of the base model. Green dots: part of the localized poses in the query model.



(b) The RANSAC result. Red line: 2D plot of the base model. Blue line: 2D plot of the original query model. Purple line: 2D plot of the transformed query model after RANSAC. Green line: inliers of localized poses.

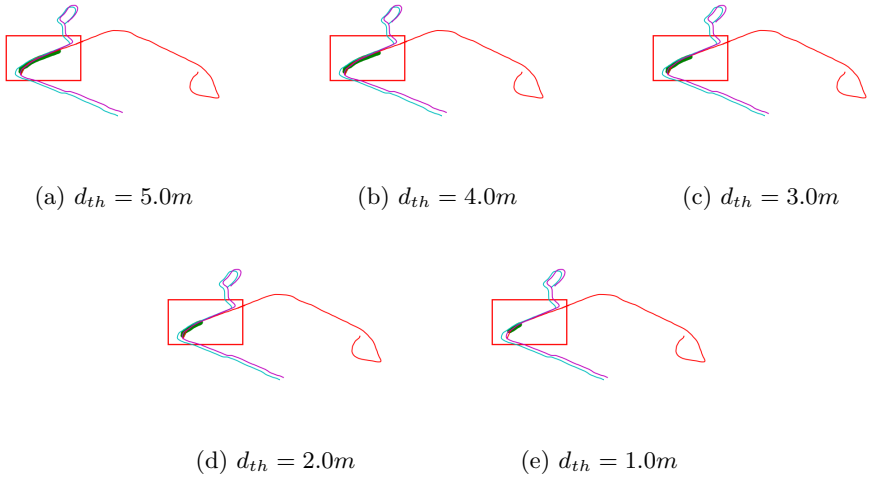
Figure 3.6: An example of robust localization result.

$$r_{overlap} = \frac{N_{GPS}(A \cap B)}{N_{GPS}(A)} \tag{3.1}$$

$N_{GPS}(*)$ represents a function of calculating the number of GPS grids in which a model has images.

$$\frac{N_{gps}}{N_{ori}} = k * r_{overlap} + b \tag{3.2}$$

N_{gps} is the number of image pairs retrieved by GPS-based method, while N_{ori} is the number of image pairs retrieved by the original method. We can obviously find that $\frac{N_{gps}}{N_{ori}}$ is positively correlated to $r_{overlap}$. We fit the data using eq. 3.2, while $k = 1.1315$, $b = 0.0248$ and the *p-value* is $7.6894e - 296$. Therefore, the less two trajectories are overlapped, the fewer pairs are needed to be retrieved.



(a) $d_{th} = 5.0m$ (b) $d_{th} = 4.0m$ (c) $d_{th} = 3.0m$

(d) $d_{th} = 2.0m$ (e) $d_{th} = 1.0m$

Figure 3.7: MSAC results of different thresholds. Red line: 2D plot of the base model. Blue line: 2D plot of the original query model. Purple line: 2D plot of the transformed query model after RANSAC. Green line: inliers of localized poses. From the results of different thresholds, we can find that the localized poses are similar, with the only difference of the number of inliers.

3.3 Robust Localization

After feature matching for all image pairs, we use the localization tool in hloc. Fig. 3.6a shows an example of localization result that many localized poses are obviously outliers. Thus, we cannot directly calculate the transformation of query model in the base frame. As described in section 2.3.2, we apply a RANSAC algorithm to output the inliers, as shown in fig. 3.6b.
In addition, we compare different settings of heuristic in MSAC to determine our best choice. Fig. 3.7 shows MSAC results of different thresholds. The transformations of query model are very similar. The thresholds only have an impact on the number of inliers. Considering all models would be optimized in the optimization step, we decide to have more localized poses after RANSAC to form the cross-constraints in PGO.

3.4 PGO

With all complete partial reconstructions and localization inliers, we can form the PGO optimization problem. Total 31 partial models are included to obtain a large optimized scene. There are total 206 valid combinations of models that share a same scene and are computed to generate the inliers of localized poses.
To get the optimized poses of all these partial reconstructions, our PGO algorithm took 1884.9s. And we visualize the PGO results as shown in fig. 3.8. From the heatmap, we can see that the intersections of different paths are optimized the most to align all partial reconstructions to obtain global consistency of a large scene.
Fig. 3.9 shows the histogram of distances between original poses and optimized poses after PGO. As seen from the table 3.3, most of the camera positions are changed within 2m, and the orientations of cameras tend to stay the same with a maximum angular change of 1°.

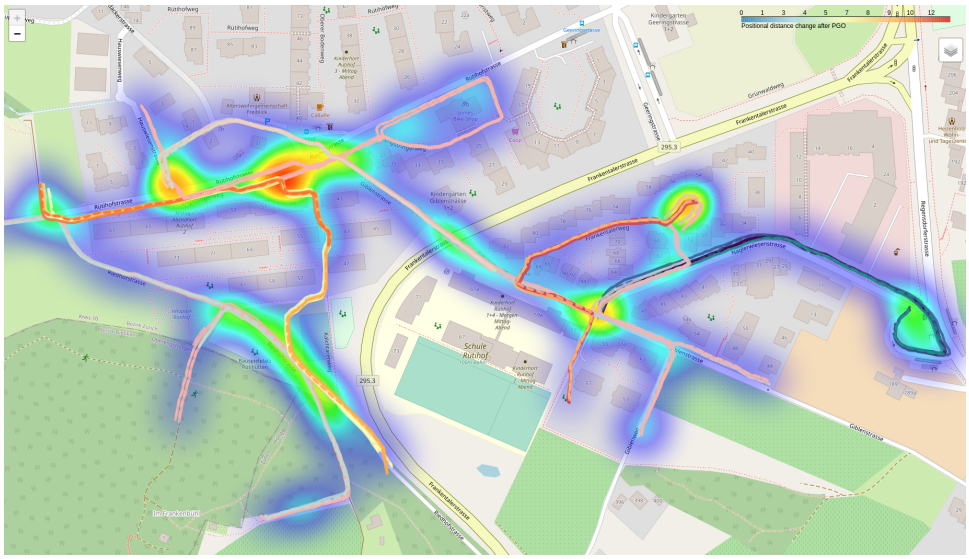
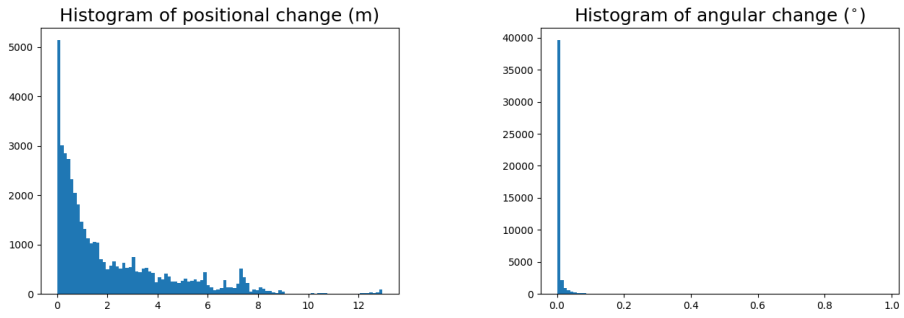


Figure 3.8: The visualization result after PGO. Solid line: trajectory of a original model. Dashed line: trajectory of an optimized model. The heatmap visualizes the distance change after PGO.



(a) The histogram of positional distances.

(b) The histogram of angular distances.

Figure 3.9: Histograms of distances between original poses and optimized poses after PGO.

Table 3.3: Distances between original poses and optimized poses after PGO.

	mean	median	max
Positional Change (m)	2.09	1.13	12.96
Angular Change (°)	5.71e-3	1.12e-4	0.97

Fig. 3.10 shows the final reconstruction result of the whole neighborhood. Since PGO only generates the optimized camera poses, we cannot directly visualize the large scene reconstruction. To fully visualize the whole neighborhood, not only the camera poses are needed, but the corresponding 3D structure also has to be optimized. It is possible to apply bundle adjustment for each partial reconstruction when triangulating 3D points with the optimized poses, which can time-consuming. Considering the angular change can be neglected, for simplicity, we only use the position of cameras to apply model-aligner for each partial model. Then, we vi-

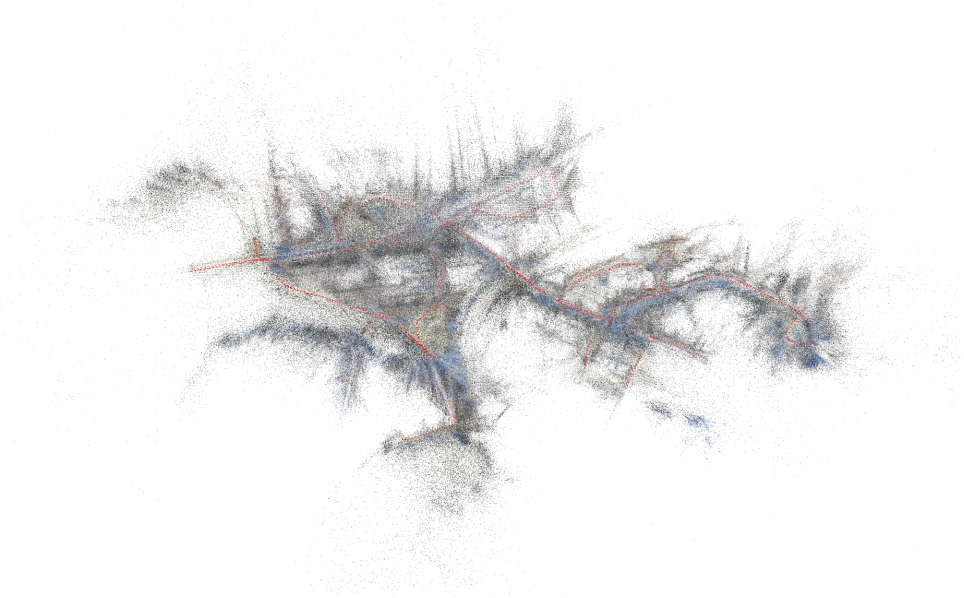


Figure 3.10: The final reconstruction result of a large scene. All camera poses, shown in red, and 3D structures are visualized.

sualize all partial reconstructions together to generate a global view of the whole neighborhood.

Table 3.4: Runtime statistics.

Module	time
Partial Reconstruction	155.38h
Merging Partial Reconstructions	46.72h
Pose graph optimization	0.52h

Finally, we show the runtime of each module in table 3.4.

3.5 Additional result of ETH Hönggerberg

We also test our pipeline by reconstructing the campus at ETH Hönggerberg. We record total 30 videos of 6 trajectories from 5 camera views. Following the pipeline, we can firstly build the partial reconstructions for each video and align them with the GPS data. Then we try to build the GPS database in the part of merging partial reconstructions. However, the recorded GPS data are susceptible to large errors since some parts of the video were recorded under roofs. Thus, we use the camera positions from aligned reconstruction models to build the GPS database instead of using the GPS data directly. After merging partial reconstructions, we finally run PGO to get the final large scene reconstruction of the campus.
Fig. 3.11 shows the final PGO result of ETH Hönggerberg. However, there are obvious artifacts shown in the red boxes. These are from the same trajectory but with different camera views, thus the positions should stay close to each other. There are several reasons that may cause such errors. The images in red boxes are probably from different GPS grids, thus no image pairs from them can be retrieved. Another reason may lie in partial reconstruction that colmap computes the intrinsic

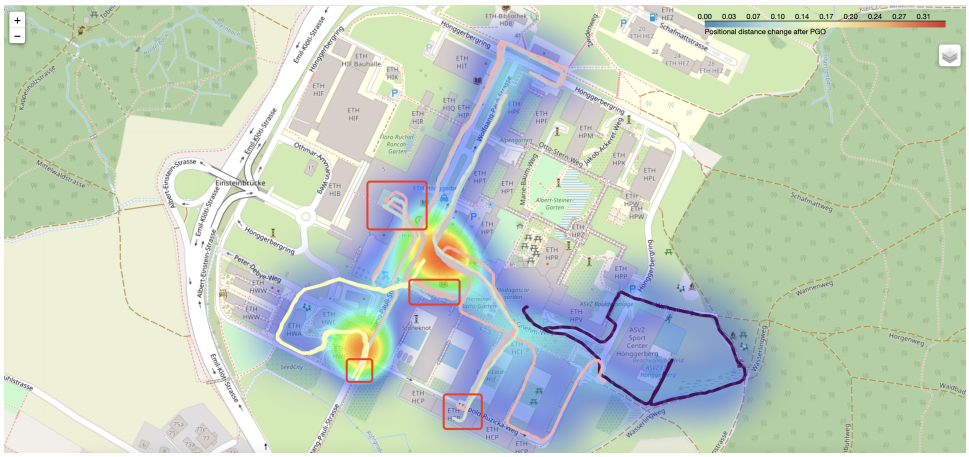


Figure 3.11: The PGO result of ETH Hönggerberg. Solid line: trajectory of a original model. Dashed line: trajectory of an optimized model. The heatmap visualizes the distance change after PGO. Red box: area with obvious artifacts.

matrices of 5 cameras with relatively large differences, which explains why some reconstruction paths are longer than others.
We also test the pipeline with a large grid size of $20m$ to alleviate the issues in the GPS-based retrieval. The same artifacts still remain in the final PGO result. Therefore, more improvements should be made in the part of partial reconstruction.

Chapter 4

Conclusions

This project mainly introduces a pipeline for a large scene reconstruction with metric. We firstly utilize tools in hloc to reconstruct partial scenes and use the GPS information to recover the metric, and then we build a GPS database to accelerate the image retrieval process by ignoring invalid image pairs and localize camera poses using RANSAC. Finally, we optimize all camera poses by PGO with cross and self constraints.
We did qualitative analysis of our pipeline and the experiments showed that our method can improve the original method in terms of speed and robustness. In addition, we developed visualization tools for this project. However, our method strongly relies on the GPS data's accuracy, thus our method is only suitable for outdoor scene reconstruction. Additionally, there is still lack of tools to evaluate the reconstruction result quantitatively. In most times, we only evaluate our method qualitatively based on the visual effect on the map. It is expected to present a systematic way of evaluation when there is no ground truth data.
For future work, we can develop applications based on large scene reconstructions. We expect to introduce NeRF [30] to large outdoor scenes with all camera poses generated by our pipeline. There are many challenging problems to scale NeRF up. Our pipeline provides possibilities for NeRF in large scenes, which could be an impressing tool for virtual city touring.
In summary, we present our pipeline for large scene metric reconstructions. By fully utilizing the GPS information, we can improve the accuracy, robustness of 3D models and accelerate the process of reconstruction. We believe that, with this pipeline, many applications can be developed for even city-scale 3D reconstruction.

Bibliography

[1] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: exploring photo collections in 3d," in *ACM siggraph 2006 papers*, 2006.

[2] F. Schaffalitzky and A. Zisserman, "Multi-view matching for unordered image sets, or "how do i organize my holiday snaps?"," 04 2002.

[3] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from internet photo collections," *International journal of computer vision*, 2008.

[4] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell *et al.*, "Detailed real-time urban 3d reconstruction from video," *International Journal of Computer Vision*, 2008.

[5] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[6] w.-y. Lin, S. Liu, N. Jiang, M. Do, P. Tan, and J. Lu, "Repmatch: Robust feature matching and pose for reconstructing modern cities," 10 2016.

[7] K. E. Ozden, K. Schindler, and L. Van Gool, "Multibody structure-from-motion in practice," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[8] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, 1981.

[9] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE transactions on pattern analysis and machine intelligence*, 2004.

[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, 2004.

[11] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: A survey," *Foundations and Trends® in Computer Graphics and Vision*, 2008.

[12] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *European conference on computer vision*, 2006.

[13] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*, 2011.

[14] M. Brown and S. Winder, "Discriminative learning of local image descriptors," *IEEE transactions on pattern analysis and machine intelligence*, 01 2011.

[15] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "Lift: Learned invariant feature transform," in *European conference on computer vision*, 2016.

[16] D. Daniel, M. Tomasz, and R. Andrew, "Superpoint: Self-supervised interest point detection and description," *CoRR*, 2017.

[17] R. Arandjelovic and A. Zisserman, "All about vlad," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2013.

[18] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[19] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, 6 1981.

[20] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment — a modern synthesis," in *Vision Algorithms: Theory and Practice*, B. Triggs, A. Zisserman, and R. Szeliski, Eds., 2000.

[21] Y. Li, Y. Ushiku, and T. Harada, "Pose graph optimization for unsupervised monocular visual odometry," *CoRR*, 2019.

[22] A. Cohen, T. Sattler, and M. Pollefeys, "Merging the unmatchable: Stitching visually disconnected sfm models," in *IEEE International Conference on Computer Vision (ICCV)*, 12 2015.

[23] T. Kühner and J. Kümmerle, "Large-scale volumetric scene reconstruction using lidar," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6261–6267.

[24] M. Fang, T. Pollok, and C. Qu, "Merge-sfm: Merging partial reconstructions," in *BMVC*, 2019.

[25] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From coarse to fine: Robust hierarchical localization at large scale," in *CVPR*, 2019.

[26] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperGlue: Learning feature matching with graph neural networks," in *CVPR*, 2020.

[27] R. Arandjelovic, P. Gronát, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: CNN architecture for weakly supervised place recognition," *CoRR*, 2015.

[28] P. Torr and A. Zisserman, "Mlesac: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, 08 2000.

[29] S. Agarwal, K. Mierle, and T. C. S. Team, "Ceres Solver," 3 2022.

[30] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," 2020.