ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Semester Thesis

Online Extrinsic Camera Calibration from Multiple Keyframes Using Map Information

Spring Term 2023

Supervised by:
Cornelius Von Einem
David Hug

Author:
Eric Tüschenbönner

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Contents

Abstract

Vision-based driver assistance systems play a critical role in improving the safety of railway vehicles, especially in the case of collision detection. In this work, we propose an online method for extric calibration of a camera mounted to a track vehicle across multiple frames. The method makes use of a preinstalled GPS sensor to localize the vehicle with respect to a map and a detail camera to identify obstacles between the railway tracks. In order to determine this region of interest, local railway tracks from the map are reprojected into camera view. For a precise reprojection, the camera pose with respect to the GPS sensor is required.

Our proposed method to estimate the 6 degree-of-freedom camera pose is based on an optimization approach. The railway map and elevation data is processed in order to reproject local railway tracks into the camera view of each frame, upon which the reprojection error between reprojected and detected tracks in the image is minimized, utilizing an iterative closest points (ICP) algorithm.

Evaluation across a variety of scenes shows that the proposed method is able to accurately and precisely estimate the camera pose, as compared to stereo camera calibration ground truth, at least in single-frame optimization. Given a decent initial height and depth estimate, the method is robust to different track shapes – keyframes with depth-variability of the track actually improve the result. When it comes to multi-frame optimization, the accuracy of the method is limited by sensor noise, in particular the RTK-GPS rotation measurements, since small angular errors lead to large reprojection errors. This limitation can likely be overcome by applying sensor fusion to improve the GPS state estimate with IMU and odometry data.

Symbols

Symbols

ϕ, θ, ψ	roll, pitch and yaw angle
b	gyroscope bias
Ω_m	3-axis gyroscope measurement
λ	...

Indices

K	Intrinsic parameter matrix
x	x axis
y	y axis
z	z axis
u	horizontal pixel coordinate
u_0	horizontal center pixel
v	vertical pixel coordinate
v_0	vertical center pixel

Acronyms and Abbreviations

ETH	Eidgenössische Technische Hochschule
GPS	Global Positioning System
ICP	Iterative Closest Points algorithm
IMU	Inertial Measurement Unit
OSM	Open Street Map
UTM	Universal Transverse Mercator coordinate system

Chapter 1

Introduction

Obstacle detection is crucial for safe operation of railway vehicles. A prerequisite for this is identifying a region of interest (ROI), in other words knowing where to look for obstacles, which means that the tracks ahead of the vehicle need to be correctly identified and located. This could be done by projecting a known railway map into camera view. However, this requires precise knowledge of the camera position and rotation – not typically available in the field or with existing datasets. Given the degree of accuracy required for long-range obstacle detection (LROD), this is a non-trivial task.

The aim of this semester project is to develop an extrinsic camera calibration and reprojection pipeline based on visual cues and map information, where the intrinsics of the camera are known. The available data includes a set of images with associated pose readings from a GPS sensor, as well as a global railway map. The images are taken from a camera that is mounted to the front of a vehicle, which is driving along the railway track. The GPS sensor is also attached to the vehicle, but in a different place. Moreover, the map information includes OpenStreetMap (OSM) data with the positions and properties of railway nodes and tracks, as well as elevation data.

A previous solution [1] proposes a geometric approach, applying a line-detection algorithm to extract image features such as vanishing points, railway sleepers, and poles from an image, which are used to compute the camera pose step-by-step. However, this approach is not generalizable as it only works on carefully-selected individual frames with straight tracks.

The solution presented in this report proposes to directly reproject the 3D map into camera view, and formulate an optimization problem to find the correct camera pose. The aim is to build a robust pipeline that works across multiple frames, thus alleviating the prior shortcomings.

This report is structured as follows. Chapter 2 provides background information in order to introduce relevant technical concepts for an improved understanding. Chapter 3 outlines the proposed method, describing all main components, processes, as well as inputs and outputs in detail. Chapter 4 discusses the implementation details, which includes the software architecture, methods, and data types so that others are able to operate and extend the pipeline. Chapter 5 presents and evaluates the results. Chapter 6 concludes the report and discusses possible future work.

Chapter 2

Background

This chapter summarizes useful background information to better understand the method and implementation of the project that will be outlined in the following chapters.

2.1 Coordinate Systems

Three different coordinate systems are used for the purpose of this project: world, GPS, and camera frame. The world frame is constant and in UTM coordinates, used by the map information. The GPS frame is the time-variant position of the GPS sensor, while the camera frame is the time-variant position of the camera on the train.

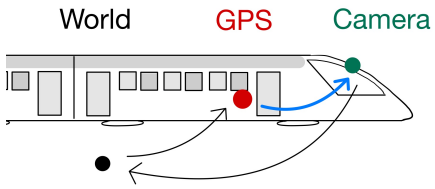

Figure 2.1: Coordinate systems illustrated

Table 2.1 provides an overview of the axis directions of the GPS and camera frames. In the GPS frame the X-axis points forward, the Y-axis to the right of the train, and the Z-axis upwards, whereas in the camera frame the axes are approximately as follows: the Z-axis points forward, the X-axis to the right of the train, and the Y-axis downwards. The camera frame is defined such that the camera is looking in the direction of the Z-axis, the width of the image is along the X-axis and the height of the image is along the Y-axis.

Table 2.1: Definitions of directions and rotations, with associated GPS and camera axes.

Direction	**Rotation**	**GPS axis**	**Camera axis**
Longitudinal (forward)	Roll	$+X_{GPS}$	$+Z_{cam}$
Lateral (sideways, right)	Pitch	$+Y_{GPS}$	$+X_{cam}$
Vertical (upwards)	Yaw	$+Z_{GPS}$	$-Y_{cam}$

2.2 Coordinate Transformations

In order to efficiently transform points between the different coordinate systems, namely those described in the previous section, it is important to understand the underlying methods. This section summarizes the most important concepts.

For the purpose of this project, homogeneous transformation matrices have been used most of the time since they are more intuitive. However, quaternions are used for the optimization, where they are dynamically adapted, since they are not prone to numerical singularities.

2.2.1 Homogeneous Transformation Matrix

Transformation, including both translation and rotation, of a vector to point P, from initial frame $\mathcal{A}$ to frame $\mathcal{B}$. This is achieved using the rotation matrix $R_{\mathcal{BA}}$ (notation: frame $\mathcal{A}$ to frame $\mathcal{B}$) and translation vector ${}_{\mathcal{B}}\boldsymbol{t}_{BA}$ (notation: from point A to point B, expressed in frame $\mathcal{B}$). To avoid computation issues, it is crucial to remember which frames the vectors are expressed in.

$$ {}_{\mathcal{B}}\boldsymbol{r}_{BP} = {}_{\mathcal{B}}\,\boldsymbol{t}_{BA} + R_{\mathcal{BA}} \cdot {}_{\mathcal{A}}\,\boldsymbol{r}_{AP} \tag{2.1} $$

This can also be combined as a homogeneous transformation matrix $H_{\mathcal{BA}}$.

$$ \begin{bmatrix} {}_{\mathcal{B}}\boldsymbol{r}_{BP} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} R_{\mathcal{BA}} & {}_{\mathcal{B}}\boldsymbol{t}_{BA} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}}_{H_{\mathcal{BA}}} \cdot \begin{bmatrix} {}_{\mathcal{A}}\boldsymbol{r}_{AP} \\ 1 \end{bmatrix} \tag{2.2} $$

To determine the inverse of a homogeneous transformation matrix, the translation vector need not only be reversed but also rotated to the new frame, while the rotation matrix is simply transposed.

$$ H_{\mathcal{AB}} = \begin{bmatrix} R_{\mathcal{AB}} & {}_{\mathcal{A}}\boldsymbol{t}_{AB} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} = \begin{bmatrix} R_{\mathcal{BA}}^{T} & -R_{\mathcal{BA}}^{T} \cdot {}_{\mathcal{B}}\,\boldsymbol{t}_{BA} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.3} $$

2.2.2 Quaternion Rotation

Definition of a quaternion $\boldsymbol{q}$ (4D vector) and its conjugate $\boldsymbol{q}^*$.

$$\boldsymbol{q} = q_w + q_x \cdot \boldsymbol{i} + q_y \cdot \boldsymbol{j} + q_z \cdot \boldsymbol{k} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} \qquad \boldsymbol{q}^* = \begin{bmatrix} q_w \\ -q_x \\ -q_y \\ -q_z \end{bmatrix} \tag{2.4}$$

Must be a unit quaternion (scaled to unit norm)

Rotation using the quaternion product $\otimes$ (equal to cross-product minus dot-product)

$$\begin{bmatrix} 0 \\ {}_{\mathcal{B}}\boldsymbol{r} \end{bmatrix} = \boldsymbol{q}_{\mathcal{B}\mathcal{A}} \otimes \begin{bmatrix} 0 \\ {}_{\mathcal{A}}\boldsymbol{r} \end{bmatrix} \otimes \boldsymbol{q}^*_{\mathcal{B}\mathcal{A}} \tag{2.5}$$

2.3 Camera Model

2.3.1 Reprojection

Pinhole camera model

Reprojection of coordinates (x, y, z) in the camera frame to pixel coordinates (u, v) in the image plane. The variables f_x and f_y are the focal lengths in pixels, while c_x and c_y are the principal point coordinates in pixels.

$$u = f_x \cdot \left(\frac{x}{z}\right) + c_x \tag{2.6}$$

$$v = f_y \cdot \left(\frac{y}{z}\right) + c_y \tag{2.7}$$

This can also be written in matrix form, with the camera instrinsics matrix K, where the variable λ is the depth scaling factor since infinitely many 3D points would project to the same 2D point.

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{K} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.8}$$

2.3.2 Undistortion

Equidistant model

$$r = \sqrt{u^2 + v^2} \tag{2.9}$$
$$\theta = \arctan(r) \tag{2.10}$$

$$\theta_d = \theta(1 + k_1 \cdot \theta^2 + k_2 \cdot \theta^4 + k_3 \cdot \theta^6 + k_4 \cdot \theta^8) \tag{2.11}$$

...

Done using OpenCV fisheye

2.4 Iterative Closest Points (ICP)

Optimization formulation

Chapter 3

Method

This chapter outlines the method of the reprojection and optimization pipeline. Following an overview of all components in section 3.1, the process of each component is described in detail. Implementation details will be covered in the next chapter.

3.1 Overview

Figure 3.1 shows an overview of the four main components: railway processing, track detection, track reprojection, and pose optimization, as well as the global inputs and outputs, color-coded by the type of data. The global map data (green) includes *railway network* and *elevation data*, the frame-specific data (red) includes *image* and *GPS pose*, and the camera-specific data (blue) includes *camera intrinsics* and *camera pose*. All of these are inputs to the pipeline, and the output that is iterated over is the *camera pose*. The processes in each component are described in the following sections.

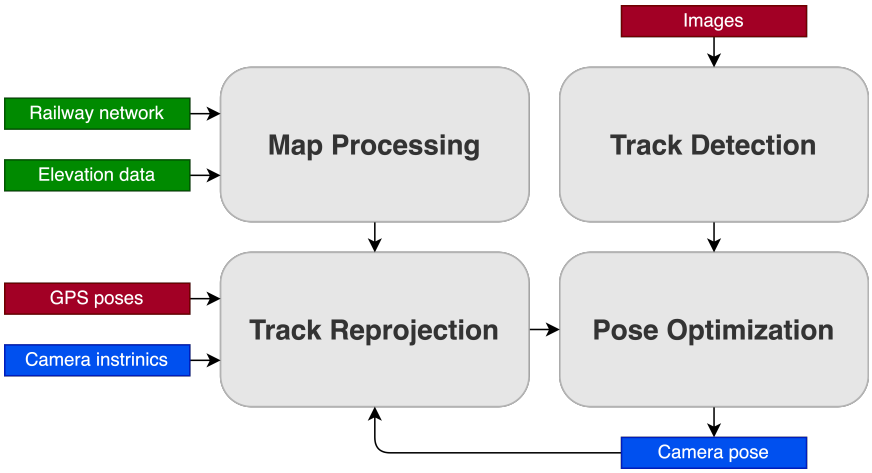

Figure 3.1: Overview of the components, interactions, and inputs/outputs color-coded by type (green: global map, red: frame-specific, and blue: camera-specific).

3.2 Map Processing

In this component, raw map data is processed into a 3D point cloud for each railway track that is more usable for downstream components, particularly Track Reprojection. The raw map data includes a railway network in the form of an OpenStreetMap (OSM) file (illustrated in fig. 3.2), as well as elevation data. The output is a set of 3D points for each railway track.

The process includes extracting the nodes and tracks from the railway map, converting them to 2D splines sorted by tracks, then filling the gaps between the points to achieve more regular spacing, and finally adding elevation data to obtain a 3D point cloud for each track. An overview of the input, process, and output is shown in table 3.1.

Table 3.1: Map processing component: input, process, and output.

Input:	• Railway network (OSM file) • Elevation data
Process:	1. Extract nodes and tracks from railway network 2. Convert nodes to 2D splines per track 3. Fill track gaps by 2D interpolation 4. Add elevation data to get 3D points
Output:	• Railway tracks as 3D point clouds

Overall, the initial data is enhanced, while also being reduced to what is actually needed for downstream components – only in the relevant area that is a combined map of all specified frames. Even though processing a combined map of all keyframes takes more time once, it is much faster than processing the railway surrounding each frame location individually, since points may overlap. Moreover, this is the only way to ensure that large railway gaps do not lead to a problem of missing data in the reprojection of any single frame, as would be the case otherwise.

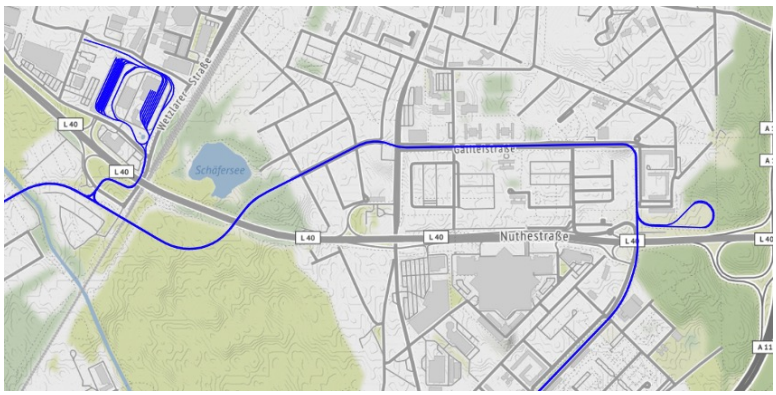

Figure 3.2: Railway network from OpenStreetMap (OSM) file.

3.3 Track Reprojection

In this component, local 3D railway points are reprojected onto each image, given the GPS pose of the current frame, the camera intrinsics, and camera pose estimate. The output is a set of dense, regularly-spaced 2D points on the image.

This is done by first finding local railway tracks, by searching within a radius of interest around the current frame GPS pose, then increasing the point density using 3D interpolation, transforming these points into the camera frame, filtering the points by angle to the camera (to obtain a more regular spacing in image space), and finally reprojecting these points into the image space. An overview of the input, process, and output is shown in table 3.2.

Table 3.2: Track reprojection component: input, process, and output.

Input:	• Railway tracks as 3D point clouds • GPS pose • Camera pose estimate • Camera intrinsics
Process:	1. Find local railway tracks – search radius of interest 2. Increase point density by 3D interpolation of tracks 3. Transform points into camera frame 4. Filter number of points by angle to camera 5. Reproject onto image
Output:	• Reprojected local tracks

In contrast to the previous component, this process is run for each frame individually, and the reprojection step is repeated at each iteration of the camera pose, which is iterated over in the Pose Optimization component. This means that the filtered, dense points in 3D space are constant throughout the iterations per frame, yet reprojected differently for each camera pose.

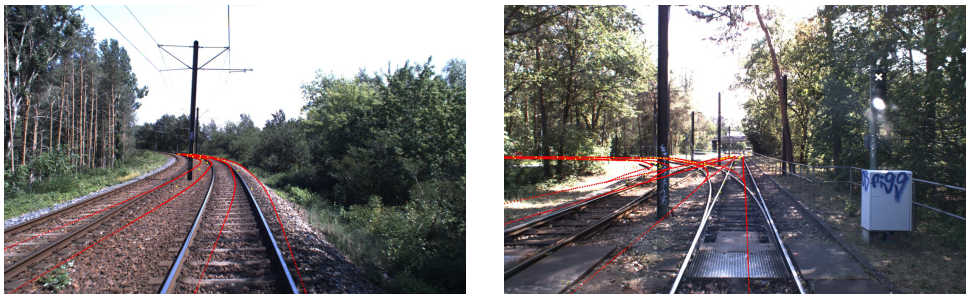

Figure 3.3: Reprojected enhanced railway tracks (red) and original nodes (yellow).

3.4 Track Detection

In this component, visible railway tracks are observed in each image, currently by manual annotation but expandable to automated detection. The output is a set of dense 2D points for each observed track.

The process consists of manually annotating points along the tracks in each image, converting these points to 2D splines, and increasing the point density by 2D interpolation. An overview of the input, process, and output is shown in table 3.3, while the annotated points and interpolated output is shown in fig. 3.4.

Table 3.3: Track detection component: input, process, and output.

Input:	• Image
Process:	1. Manually annotate 2D points in images 2. Convert to 2D splines 3. Increase point density by 2D interpolation
Output:	• Observed tracks as 2D points

This component is not yet implemented with machine learning since the focus of this project was on the optimization algorithm and reprojection pipeline. As such, manual annotation is used to obtain ground truth data for the evaluation of the Pose Optimization component. Nevertheless, track detection is a crucial part of the pipeline, and should be integrated in future work in order to obtain a fully automated and generalizable pipeline.

Figure 3.4: Annotated points (light blue) and interpolated splines (dark blue).

3.5 Pose Optimization

In this component, the camera pose is optimized by minimizing the error between the observed tracks and the reprojected local tracks. The output is an updated camera pose estimate.

This is done by using an iterative closest point (ICP) algorithm. At first, one-to-one correspondences are found between the observed and reprojected points from which residuals are computed, equal to the distance between corresponding points. Second, these residuals are added to the optimization problem and it is solved to obtain a new camera pose estimate. This process is repeated until convergence, with new correspondences found at each iteration. An overview of the input, process, and output is shown in table 3.4, while the correspondences are illustrated in fig. 3.5.

Table 3.4: Pose optimization component: input, process, and output.

Input:	• Observed tracks • Reprojected local tracks
Process:	1. Find one-to-one correspondences for each observed point 2. Compute residuals 3. Add to optimization problem 4. Solve optimization problem
Output:	• New camera pose estimate

Specifically, one-to-one correspondences are found as follows: for each observed point, the closest reprojected point is found. If a reprojected point happens to be associated to multiple observed points, only the closest observed point remains associated to the reprojected point, while all others are removed and left without correspondence. This is done to ensure that each observed point is associated to only one reprojected point, and vice versa.

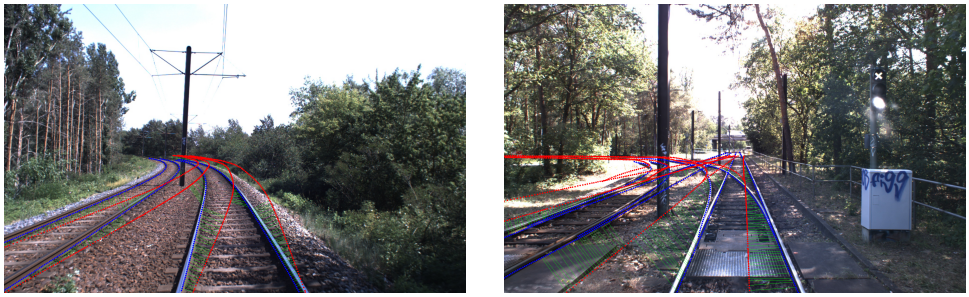

Figure 3.5: Initial correspondences (green) between reprojected points (red) and observed points (blue).

The optimization problem can be solved as single-frame or multi-frame, where the latter implies that multiple frames are optimized in parallel. More details regarding these approaches are provided below.

Single-frame vs. Multi-frame

In the single-frame approach, the optimization problem is solved for an individual frame. At first, the initial camera pose estimate is used to reproject the 3D points onto the image plane. Then, correspondences are found between the reprojected points and the observed points. These are used to compute the residuals and add them to the optimization problem, which is then solved to obtain a new camera pose estimate. This process is repeated until convergence, with new correspondences found at each iteration given the updated camera pose. An overview of the process is shown in fig. 3.6.

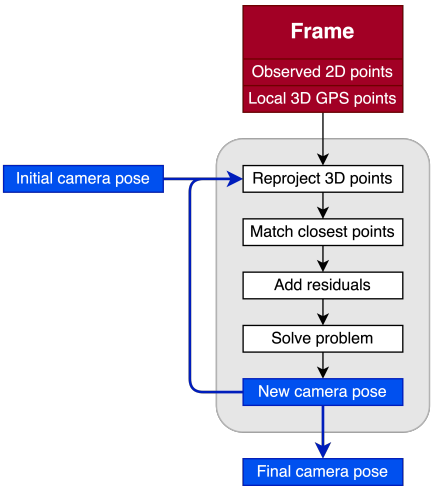

Figure 3.6: Single-frame optimization process.

In the multi-frame approach, the optimization problem is solved for multiple frames in parallel. At first, the initial camera pose estimate is used to reproject the 3D points onto the image plane for each frame. Then, correspondences are found between the reprojected points and the observed points for each frame. These are used to compute the residuals and add them to the same combined optimization problem, which is then solved to obtain a new camera pose estimate. This process is repeated until convergence, with new correspondences found for all frames at each iteration given the updated camera pose. An overview of the process is shown in fig. 3.7.

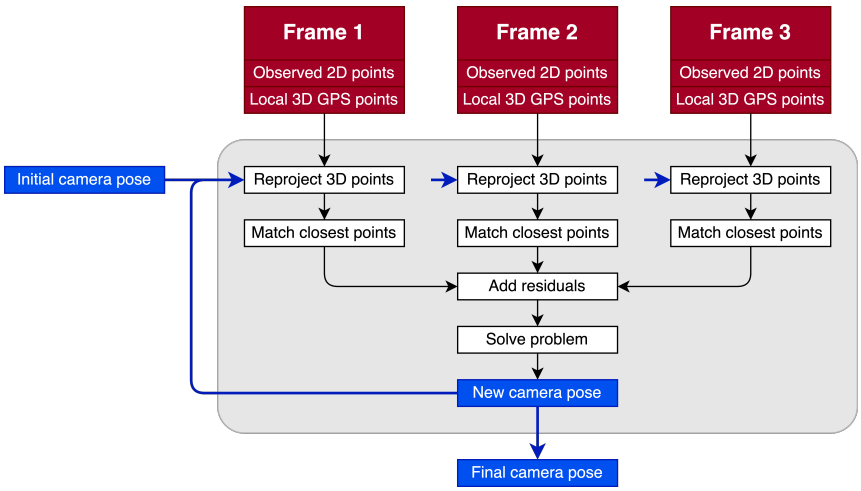

Figure 3.7: Multi-frame optimization process.

Chapter 4

Implementation

This chapter dives into the implementation details of the method described in the previous chapter

The pipeline is designed to be modular, such that each component can be replaced by a different implementation, as long as the inputs and outputs are compatible. This allows for easy experimentation with different approaches, as well as the possibility to use the pipeline in different contexts.

Objects, classes & interactions

Algorithmic implementation, efficiency, speed

Flowchart of code (files, classes, methods)

Better as table ???

Using Python for most tasks

C++ for optimization with Ceres

Libraries: OpenCV, NumPy, Ceres, ...

4.1 Python Classes & Objects (Methods, Data)

... main file & sequence

4.1.1 Railway

Which methods & data types enable the process as described in method

For efficiency: build using combined map of relevant keyframes

4.1.2 Keyframe & GPS

Image, annotations

4.1.3 Camera

4.1.4 Transformation

4.2 C++ Optimization: Ceres Solver

4.2.1 Cost Function

4.2.2 Residuals

4.2.3 Parameters

4.3 Required Input Data

Data file + any files imported and exported

Add to ReadMe: where to specify file paths / how to get data

railway map data ... from OSM file ?

elevation data ... from file ?

images & poses ... export from ROS Bags

Annotations as CSV ... using Website ? to create annotations

saving Railway object to file

4.4 Output Data

File paths to specify to save visualizations etc.

Chapter 5

Results & Evaluation

This chapter presents the results of the proposed method, and evaluates its performance in terms of accuracy, robustness, and generalization. Focus is put onto the single-frame optimization in Section 5.1, but the adaptation to multi-frame optimization is also discussed in Section 5.2. In both cases, the results are evaluated regarding different aspects and recommendations are given to optimize performance.

5.1 Single-frame

This section evaluates single-frame optimization with respect to convergence, robustness, and accuracy on a variety of different scenes, using the same initial camera pose. The converged results are shown in Figure 5.1.

Figure 5.1: Converged correspondences after single-frame optimization.

It can be seen that the final alignment between reprojected and observed tracks is quite good in all cases. Even if the tracks are not perfectly aligned, as is the case with some intersections (e.g. bottom right image) which is likely due to inaccuracies in the railway map, the track directions on all sides of the intersection is still very well aligned.

Convergence & Initial Estimates

The method converges predictably and reliably, given a decent initial height and depth estimate. During analysis, the method always converged after 10-50 ICP iterations (i.e. number of correspondence updates), depending on the scene complexity and initial pose estimate. In this context, convergence implies that the alignment does no longer noticeable improve with further iterations.

However, it is important to have a close initial height and depth position estimate, which are both hard to optimize for. Height is difficult to optimize for because a slight change ($\pm$ 1m) leads to a very different reprojection that may not converge to the global minimum. Depth, on the other hand, is difficult to optimize for because the reprojection error is very flat in this dimension, i.e. a small change in depth hardly leads to any change in the reprojection, since the track does not change as quickly with depth. Nevertheless, the height and depth positions are easy to approximate beforehand.

When it comes to all other initial guesses, including the lateral position and all rotations, they were simply set to zero in the camera frame – implying that the camera faces perfectly forward at the same lateral position as the GPS sensor – without any convergence issues. Overall, the method is resilient to incorrect initial guesses, as long as the height and depth estimates are decent.

Robustness & Generalization

The method is also robust to scene changes and can be generalized to new data without problems. Even if the initial ICP correspondences are predominantly incorrect, i.e. the wrong tracks are associated with each other, they still adjust themselves during optimization. An example of this is shown in Figure 5.2. This self-correcting of correspondences to the correct tracks ocurrs relatively quickly within the first few iterations, after which the pose is only refined.

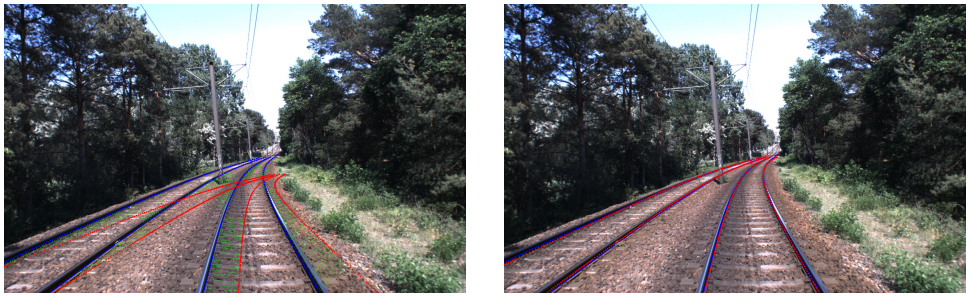

Figure 5.2: Initially incorrect correspondences (left) adjusting themselves (right).

Generalization to different scenes and new datasets is also possible, yet keyframes with higher track complexity should be selected to achieve better results. Higher track complexity means that the scenes should have a depth-variability of the track, as is the case in curves or intersections, as opposed to simple straight tracks. Examples of such scenes are the chosen frames in this chapter. In general, more complex tracks also lead to better results because more information is available to optimize for in different dimensions. With these prerequisites, the method is robust and generalizes well.

Accuracy & Precision

The accuracy of the pose estimation can be evaluated using the stereo camera calibration data, which provides a ground truth for the relative pose between the stereo cameras. At first, the stereo cameras are optimized separately, using the same optimization method as for the single-frame optimization, to get their poses relative to the GPS sensor. Then, the relative pose between the stereo cameras is computed and compared to the calibration ground truth.

Table 5.1: Stereo calibration compared to relative camera transformation obtained from different frames.

	ΔX [m]	ΔY [m]	ΔZ [m]
Calibration	0.307	0.002	0.010
Frame 1	0.289	-0.006	0.163
Frame 2	0.261	-0.047	0.128
Frame 3	0.300	0.010	0.133

As can be seen from Table 5.1, the accuracy of the calculated stereo camera transformations is relatively high. The horizontal offset ΔX and vertical offset ΔY are both within less than 0.05 m of the calibration value, while there's a slightly higher error of up to 0.16 m in the depth offset ΔZ. The horizontal and vertical offset accuracies reflect the accuracy of the reprojection and optimization pipeline. Moreover, the results are very consistent, with all three frames producing similar results, which indicates that the pose estimation is very precise as well.

The lack of accuracy in the depth estimate can be explained by the fact that the track does not change very quickly with depth (at least not on a centimeter-scale), which makes it hard to accurately estimate and sensitive to the initial guess. Nevertheless, for the same reason this is also the least critical dimension to estimate accurately, since small errors don't lead to visual reprojection errors.

When it comes to rotation, the results are all quite accurate and close to zero, as expected. If this were not the case, the reprojections would look very different since small angular errors can lead to large reprojection errors.

5.2 Multi-frame

The multi-frame optimization is evaluated on two sets of 3 frames, one with consecutive frames and another with keyframes of different scenes. All frames use the same initial camera pose estimate, as outlined in single-frame optimization above.

Multiple Keyframes

The first set of frames consists of three keyframes with different scenes that are optimized simultaneously. The initial estimates are shown in Figure 5.3, while the converged outputs are shown in Figure 5.4. A comparison to the equivalent single-frame optimization outputs is shown in Figure 5.5.

Figure 5.3: Initial correspondences between reprojected and observed points.

Figure 5.4: Converged correspondences after multi-frame optimization.

Figure 5.5: Equivalent converged correspondences after single-frame optimization.

When looking at Figure 5.4, the multi-frame results don't have a very good alignment. It seems that there are conflicts in the data, since the same camera pose estimate leads to inconsistent reprojections across frames. This becomes especially aparent when comparing to the equivalent single-frame optimization results in Figure 5.5, which are much more consistent.

Consecutive Frames

To test if these issues persist with more similar frames, a second set of frames is used that consists of three nearby frames with a similar scene that are optimized simultaneously. The initial estimates are shown in Figure 5.6, while the converged outputs are shown in Figure 5.7.

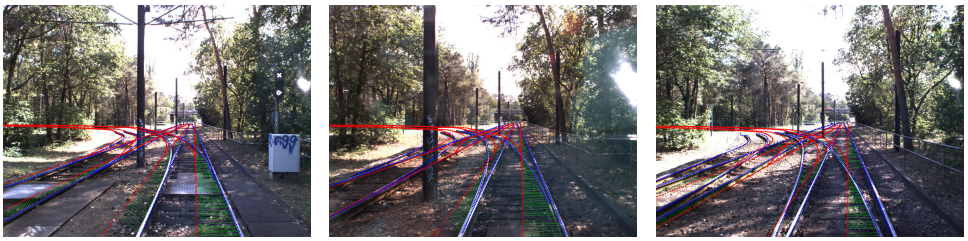

Figure 5.6: Initial correspondences between reprojected and observed points.

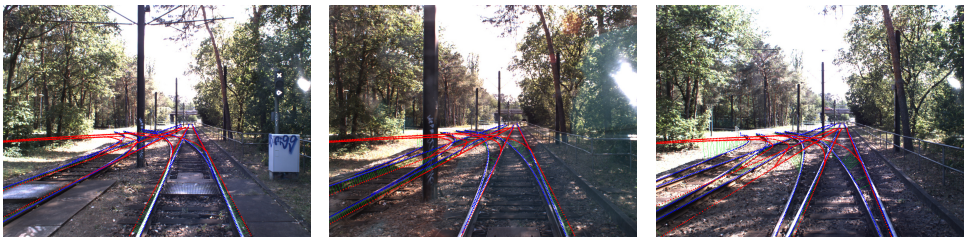

Figure 5.7: Converged correspondences after nearby multi-frame optimization.

While the alignment in Figure 5.7 seems slightly better than in Figure 5.4, it is also not very precise. The same issues with inconsistent reprojections across frames are still present. The fact that even nearby frames lead to inconsistent reprojections is a strong indicator of the data being too imprecise.

Analysis

From the results presented in this section, it has become clear that the data is not very precise across frames to be optimized together for consistent reprojections. The measurements tend to be somewhat noisy, which leads to inconsistent reprojections across frames. This is likely due to a combination of different factors, including RTK-GPS sensor noise, elevation data, and the railway nodes.

While the RTK-GPS sensor is very accurate globally, it is not always precise locally. Figure 5.8 shows RTK-GPS position measurements, while Figure 5.9 shows RTK-GPS rotation measurements.

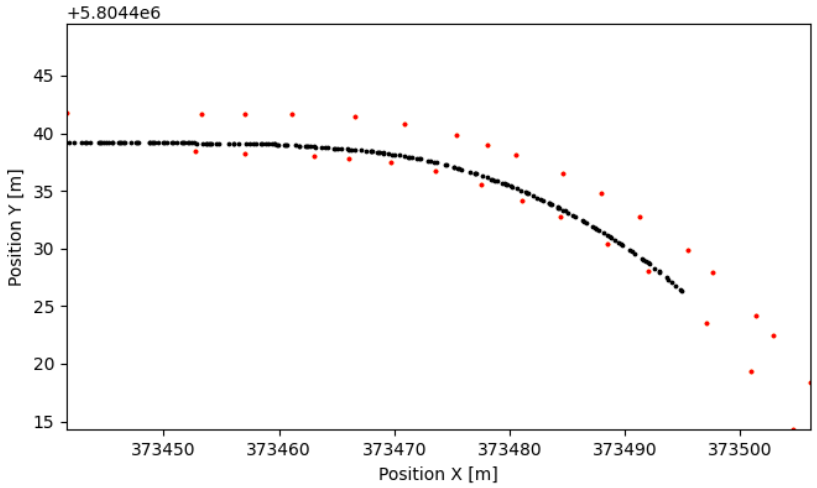

Figure 5.8: GPS position measurements (black) compared to railway nodes (red).

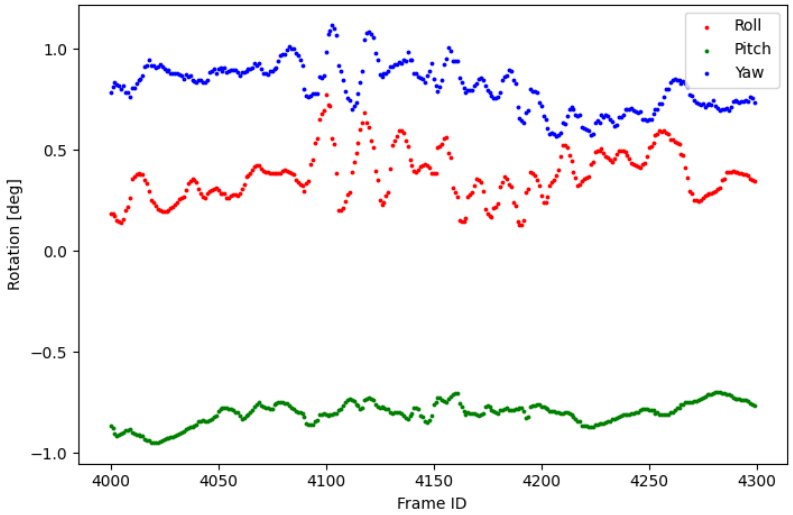

Figure 5.9: GPS rotation measurements for roll (red), pitch (green), and yaw (blue).

The position measurements seem very precise, since they all lie along a continuous line that follows the railway track. However, it could be that there are longitudinal

jumps between frames, meaning that the data might not be as precise in measuring the position along the track. This could explain some inconsistencies in Figure 5.7, where the position does not seem to be consistent with the reprojected tracks.

When it comes to RTK-GPS rotation measurements, the data is also not very precise. In all three dimensions it varies by up to 0.5 degrees between frames while it should effectively stay the same on this straight track segment. Combining this small error in all three dimensions and considering the fact that small rotations can lead to large reprojection errors, it is likely a big factor in the inconsistent reprojections. This would explain the inconsistencies in Figure 5.4, where the rotation does not match the reprojections.

Comparing the RTK-GPS height measurements (blue) with the local elevation data (red) in Figure 5.10, it appears that the former are very precise, while the latter varies by up to 0.2 m across frames. Yet, small errors in the elevation data (used for creating the 3D point clouds per railway track) are unlikely to be a big factor in the inconsistent reprojections, since all frames are affected equally. Also note that, for the RTK-GPS height measurements, there have been some outliers in the data (not visible in the graph), which are likely due to bad signal or else. Avoiding these outliers is important, but otherwise the GPS height measurements do not seem to be the issue.

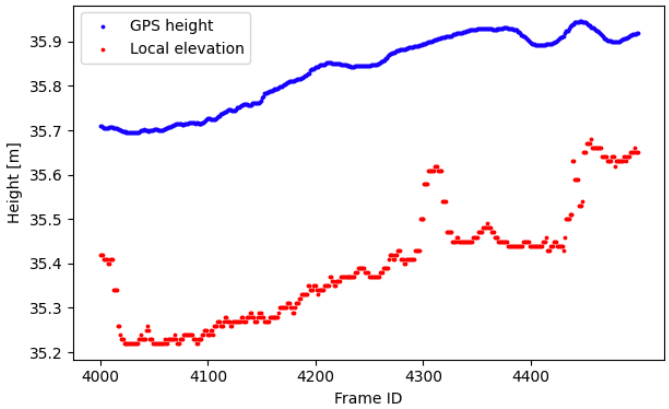

Figure 5.10: GPS height measurements (blue) vs. local elevation data (red).

Finally, the railway nodes from the OpenStreetMap data are also a possible source of error. However, similar to the elevation data they are unlikely to be a big factor in the inconsistent reprojections, since all frames are affected equally. Figure 5.8 also shows that the railway nodes look relatively precise, in addition to being globally accurate.

In conclusion, if the above data sources were more precise, the reprojections would be more consistent across frames. This would lead to better results when optimizing multiple frames together.

Recommendations

In order to overcome the limitations of the data and obtain more precise state estimates, it would be very useful to implement a sensor fusion algorithm, such as

a Kalman Filter. This means combining the RTK-GPS sensor with available IMU and odometry data. The RTK-GPS sensor is very accurate globally, but not very precise locally. The IMU and odometry are precise locally, but drift over time. By combining all three, it is possible to obtain a more precise state estimate that is both precise locally and globally. This would lead to reduced reprojection errors across frames.

A low number of only 3 frames has purely been chosen for evaluation purposes, since this made it easier to visualize the results. In practice, the method can be applied to a much larger number of frames, which would merely take longer to optimize but also not necessarily lead to better results. Using a selection of about 5-20 different keyframes is likely a good balance between accuracy and optimization time.

Chapter 6

Conclusion

This report has presented a complete extrinsic camera calibration and reprojection pipeline for a camera attached to a railway vehicle, given a set of images with associated GPS measurements as well as global map information. The 6 degree-of-freedom camera pose is estimated using an optimization formulation, where the railway map and elevation data is processed in order to reproject local railway tracks into the camera view of each frame, upon which the reprojection error between reprojected and detected tracks in the image is minimized, utilizing an iterative closest points (ICP) algorithm.

TODO: summarize implementation chapter

Evaluation of single-frame optimization on a variety of scenes has shown that the method predictably converges within 10-50 ICP iterations, depending on the scene complexity and initial pose estimate. Given a decent initial height and depth estimate, the method is robust to different track shapes and still manages to converge if the initial ICP correspondences are incorrect. On the contrary, keyframes with depth-variability of the track actually improve the result and should be given preference as input for the optimization. It has been demonstrated that the proposed method is able to both accurately and precisely estimate the camera pose, as compared to stereo camera calibration ground truth. Especially the horizontal and vertical positions are very accurate, while the depth error is slightly larger yet without noticeable effects. When it comes to multi-frame optimization, while it does converge, reprojections are inconsistent across frames. This can only be due to sensor noise, in particular the RTK-GPS rotation measurements, where small angular errors lead to large reprojection errors. If it were not for this limitation, the method should be able to consistently and accurately estimate the camera pose across multiple frames, since this is simply an adaptation of the single-frame optimization that averages out the noise.

Possible extensions to this project include track detection using machine learning as well as sensor fusion of the GPS measurements with IMU and odometry data. Automated track detection would allow the method to be applied to arbitrary scenes without manual annotation, while sensor fusion would improve the accuracy of the state estimate and thus the reprojection error and multi-frame optimization results.

Bibliography

[1] N. Spiegelhalter, C. Von Einem, and D. Hug, "Online estimation of camera extrinsics using map information," 2023.