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

Preface

...

Abstract

... short summary

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 the safe operation of railway vehicles. A prerequisite for this is knowing where to look for obstacles, which means that 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 this is a non-trivial task.

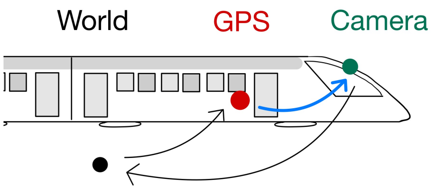
The aim of this semester project is to develop a continuous calibration and reprojection pipeline to estimate the extrinsic parameters of a camera, whose intrinsic parameters are known, given a set of images and associated pose readings from a GPS sensor that is also attached to the vehicle. Moreover, different map data is available, including OpenStreetMap (OSM) data with the positions and properties of railway nodes and tracks, elevation data, as well as a positions of poles that are located next to all railway tracks. The challenge here is to make best possible use of the data combined with visual cues to design an optimisation framework that converges to accurate results.

Chapter 2

Background

...

2.1 Coordinate Systems



World: UTM coordinates

GPS

Camera frame

Actual camera frame will change, depending on the orientation of the camera. This

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	$+Z_{GPS}$	$-Y_{cam}$

is an initial approximation.

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 $R_{\mathcal{B}\mathcal{A}}$ (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.

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

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

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

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.

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

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)

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

2.3 Camera Reprojection & Image Undistortion

2.3.1 Reprojection via the 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 Image 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

Chapter 3

Method

This chapter outlines the method of the pipeline developed during this project in order to solve the task at hand. Details regarding the general logic of each component are described in detail, while practical implementation details will be covered in the next chapter.

To simplify, the method can be divided into four main components: railway processing, track detection, track reprojection, and error minimization. Figure 3 shows an overview of these main components, the overall inputs/outputs to the pipeline (classified by type), and the interactions between the components. Each component box also specifies how often that component is executed and what the component goal/output are.

3.1 Railway Processing

In this component, a railway map (data from an OSM file) is processed in order to obtain a set of relevant global 3D points that represent the railway tracks. This is done by extracting the nodes and tracks from the railway map, converting the nodes to points sorted by tracks, then filling the gaps between the points to achieve more regular spacing, and adding elevation data to get 3D points.

Inputs

- Railway map (OSM data)
- Elevation data

Outputs

- Railway: tracks as 3D points

Process

1. Extract nodes and tracks from railway map
2. Convert nodes to points per track
3. Fill track gaps by 2D interpolation
4. Add elevation data to get 3D points

Overall, the initial data is enhanced, while also being reduced to what is actually

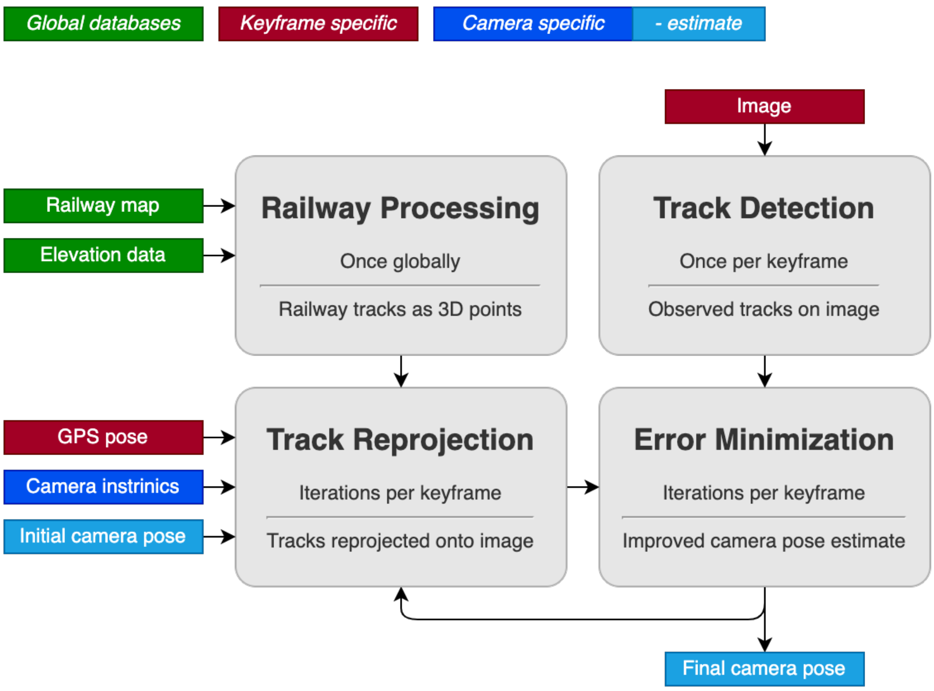


Figure 3.1: Overview of main components, their interactions, and inputs/outputs.

needed for downstream components – only in the relevant area that is a combined map of all specified keyframes. Even though processing a combined map of all keyframes takes more time once, it is much faster than processing the railway surrounding each keyframe 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 view of any single keyframe.

3.2 Track Reprojection

In this component, the railway points ahead of the current keyframe GPS location are reprojected into the image. This is done by first selecting railway tracks with local points from the global railway tracks, transforming these points into the camera frame, then filtering the points by distance from the camera (to obtain a more regular spacing in image space), and finally reprojecting these points into the image space.

Inputs

- Railway: tracks as 3D points
- Keyframe: GPS pose
- Camera: intrinsics
- Camera: pose estimate

Outputs

- Local railway tracks as reprojected 2D points

Process

1. Find local railway tracks & 3D points
2. Increase point density by 3D interpolation of tracks
3. Transform points into camera frame
4. Filter number of points by distance from camera
5. Reproject onto image

Loop over ...

3.3 Track Detection

Not fully implemented due to time constraints. For now only annotated images are used

Inputs

- Keyframe: image

Ideas from Nicolina's methods

Probably requires machine learning model to be robust to illumination changes, etc.

3.3.1 Railway Tracks

3.3.2 Poles

3.4 Error Minimization

Inputs

-

Outputs

-

Process

1.

Chapter 4

Implementation

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

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 mehods & 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 visualisations etc.

Chapter 5

Results

...

5.1 Evaluation

Reprojection errors (residuals)

Multiple cameras

Chapter 6

Conclusion

...

Algorithm requires a variety of track shapes to yield accurate results. ... Obvious since the residual is in 2D but the pose estimate is 3D 6DoF

When it comes to robustness (to different track shapes) ...

Possible extensions to this project include the implementation of a track detection algorithm, which would allow the pipeline to be used on any railway ...

Bibliography