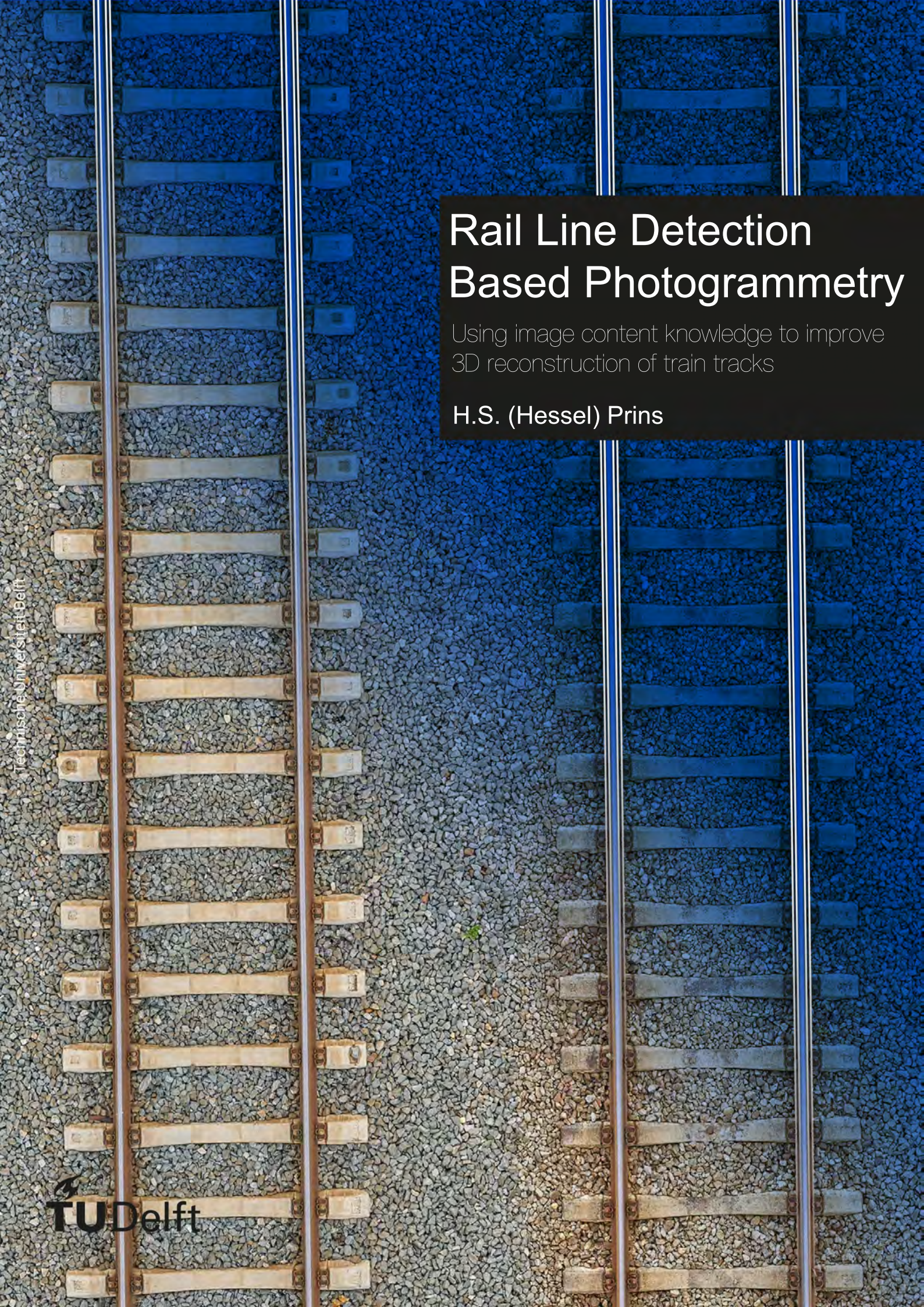


Rail Line Detection Based Photogrammetry

Using image content knowledge to improve 3D reconstruction of train tracks

H.S. (Hessel) Prins

Technische Universiteit Delft

TU Delft

Rail Line Detection Based Photogrammetry

Using image content knowledge to improve 3D reconstruction of railway tracks.

by

H.S. (Hessel) Prins

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Wednesday March 24, 2021 at 15:00.

Student number:	4096630	
Project duration:	November 1, 2020 - March 24, 2021	
Thesis committee:	Dr. R.C. Lindenbergh,	TU Delft, supervisor
	Dr. L. Nan,	TU Delft
	Dr. A.A. Núñez Vicencio,	TU Delft
	R. L. Voute,	CGI TU Delft

TU Delft

Abstract

An increasing load on the Dutch rail network requires new solutions for railway monitoring as increasing wear and tighter schedules limit the possibilities of regular maintenance. UAV (Unmanned Aerial Vehicle) based photogrammetry is such a solution which limits the presence of people near the railway and train hindrance. Photogrammetric feature extraction methods struggle with extracting features from the rusty rail and reflective roll band. Therefore line features are used to aid reconstruction. Based on SfM (Structure-from-Motion) camera parameters, a line-based reconstruction is created of two data-sets.

A line detection algorithm is used to find line segments in the images. Found segments are matched between pairs of images by pixel window cross-correlation and geometric boundaries. Matched segments are then compared to find sets of three matching images. Reconstruction is done by minimizing line reprojection error in an iterative least squares solution.

Results show promise for manually matched segments, however unfavourable automatic matching results prevent large scale reconstruction. A case study reveals matching struggles with camera rotation between images, matching problem mitigation leaves much room for improvement.

Contents

1 Introduction

1.1. Problem definition

The Dutch rail network endures an increasing load of more, heavier and faster trains. Following this, increasing wear in the ballast keeping rails in place, as well as subsiding foundation of rail superstructure can be expected. To combat this, regular maintenance is required, which is managed by the Dutch organisation ProRail. ProRail is responsible for building, maintaining, the operation and safety on and around the tracks. Activities which are all spread over several sub-contractors.

An important part of maintaining the railway tracks is monitoring. The complete network of tracks is regularly monitored by contractors of ProRail to check for parts which show wear or displacement. To guarantee a constant level of quality measurements, ProRail created a guideline RLN00296 which specifies the requirements for monitoring elements of the so called PVS, covering the geometry of the rail network. This guideline requires a maximal standard deviation of 15 mm on the XYZ coordinates of a track, 10 mm on the Z coordinate of switches, and 3 mm on the track cant. All measurements for ProRail have to be in the Dutch 'Rijksdriehoek' (RD) coordinate system, with the Z coordinate in the vertical datum 'Normaal Amsterdams Peil' (NAP).

Multiple methods of monitoring are currently applied, for example levelling, GPS campaigns, terrestrial laser scanning, or mobile measurements using a specialised train. Every method has its pros and cons, on either time or accuracy requirements. Train based measurements often require a specialised train, which has to be scheduled in on an already overused network. Currently, ProRail is conducting research with sensors placed on passenger trains, evading this issue. Ground based measurements like GPS, TLS and levelling require people to work on and around the tracks, which is limited by passing trains. Therefore, a method is sought after to limit the presence of people around train tracks, the disturbance of passenger and freight train schedules and the amount of time spend monitoring.

Photogrammetry is one of the options considered, where photographic images are used to get qualitative measurements of an area of interest. In this case, an UAV is used to obtain a large set of photos of an area in the Dutch rail network. Photogrammetric software is used to extract features from the images, match those features, estimate the camera poses and, together with accurately measured ground control points, create a geo-referenced point cloud of the area. Rail fitting is then applied on the point cloud to identify individual rails and extract their RD/NAP location, [16].

Previous research show an accurate point cloud can be created using UAV based photogrammetry, which is measured on known locations of control points. However, improvements can be made in the reconstruction of rails itself. The created point clouds show a loss in detail around the rail locations, to

a degree where the supposed rail profile is reconstructed as a bell shape. A reason for the decrease of detail on rail profiles can be found in the homogeneous and reflective nature of a rail.

Photogrammetry algorithms like Structure-from-motion (SfM) have issues finding features and matches on texture-less, homogeneous and reflective surfaces, [25]. Structure-from-Motion is an algorithm to create a 3D model (Structure) from highly overlapping photos taken from different locations (Motion). Its main advantage is the ability to create a 3D model without prior knowledge. With only the input images, the algorithm provides in automatic feature extraction, matching, camera pose estimation and calibration to create a sparse point cloud, [27].

Automatic feature extraction methods, the first step of SfM, aim to find parts in images which are distinctive from other parts. SIFT and SURF aim to find blobs, which are regions with similar properties, using respectively a Laplacian filter or the Hessian matrix. As with corner detecting base alternatives like ORB and BRISK, the algorithms look for intensity changes in multiple directions, [21]. Rails are on a UAV photogrammetry scale beams of homogeneous rusty colour with a reflective rolling band. Therefore automatic feature detection methods struggle with rail resulting in a minimum amount of key-points.

As texture-less, homogeneous and reflective surfaces are plentiful in a real world environment, research has been done in several ways to find a suitable mitigation. Options include adding features to the objects, where one can think of a matte paint to cover reflective surfaces, or less destructive, adding points by either using a projector to cover texture-less surface with a grid of points, [19], or using laser points on a rotating table for smaller objects, [10]. Both are not usable in a larger scale environment.

A different option is found when inspecting man-made objects, which often consist of large planar homogeneous surfaces. Where texture is absent, and corners few, a lot of lines can be found. Already in 1999, Baillard et al. proposed a method which utilizes line detection in triplets of aerial photos to reconstruct roofs on buildings, [1]. Schindler et al. uses vanishing point detection in 2D images of buildings to classify each line as either vertical or in a horizontal set to reduce the number of parameters required for 3D line estimation, [17].

More recently, Nurutdinova and Fitzgibbon proposed a method to use 3D curves for camera pose estimation and sparse reconstruction, [11]. Using curves and some points in a low-texture environment, a bundle adjustment method is proposed. However, curve matching is not included. Hofer et al. did include a line matching algorithm, selecting possible matches based on epipolar lines. The research focussed on the dense reconstruction and proposed a Multi-View Stereo method to create a line based representation of a scene, [9]. Multi-View Stereo (MVS) is an algorithm to reconstruct a scene. It requires undistorted images and their camera location and orientation parameters, all estimated by a Structure from Motion algorithm. The sparse point cloud with matched key-points is used as ground control points by MVS.

None of these methods incorporates geo-referencing in their research and therefore only access relative accuracy to ground truth. To be able to geo-reference a SfM point cloud, multiple points are required with world coordinates (such as RD/NAP). These points can be divided in two types: Ground Control Points (GCP) and Air Control Points (ACP). GCPs are markers, for example a plane with a cross painted on, which are geo-referenced by dGPS or Total Station measurements. These points are then marked and matched on photos in the dataset, which result in points in the point cloud with known world coordinates. ACPs are the coordinates of the UAV while taking a photo, obtained by equipping the UAV with a high accuracy GNSS system like PPK. A common method is using the ACPs as initial estimates for the camera locations in the SfM algorithm, the resulting point cloud is then via a similarity transformation linked to the GCPs, [4].

1.2. Research question

This research aims to combine a Structure-from-Motion base with line detection and reconstruction to provide a pipeline capable of finding lines, matching them, find the camera parameters, and use ground and/or air control points to reconstruct a geo-referenced line based representation of rails. Therefore, the research question to be answered is:

What contribution can line recognition on the input images bring to the structure-from-motion based process to extract the geo-referenced 3D position of railway tracks?

As the research involves different parts leading to one end product, the sub-questions to help answer the research question have been split in three domains: Line recognition to find the rails in the input images, Line matching and Structure-from-Motion to match the lines and use them in a Structure-from-Motion algorithm and geo-referencing and validation to compare the results with existing point-clouds.

- **Line recognition in original images**
  - At what success rate can rails be detected in both orthogonal and oblique 2D images using line detection algorithms while limiting false positives?
  - What is the pixel accuracy of found rail lines in the images, validated with manual inspection?
- **Line matching and Structure-from-Motion**
  - What features can be used to match railway parts such as rails and sleepers between images?
  - Can constraints found in specifications of rail superstructures be used to improve the SfM model?
- **Geo-referencing and validation**
  - Which methods are available to geo-reference the sparse cloud resulting from the SfM process?
  - Which accuracy and precision in the RD/NAP coordinate-system can be found on the GCPs comparing traditional SfM and the proposed method?
  - Which accuracy, precision and data completeness in the RD/NAP coordinatesystem can be found on the rails comparing 3D rail detection in the traditional SfM point cloud and the proposed method?

2

Monitoring railways using photogrammetry

This chapter will provide the background details necessary to build the methods proposed in the next chapter. Section 2.1 explains the construction of railway infrastructure and currently used monitoring methods and requirements. Section 2.2 gives an introduction to photogrammetry and explains the photogrammetric process of Structure-from-Motion. It also covers used geometric concepts such as internal and external camera parameters.

These parameters are required for section 2.3 to describe correspondences between different images. The relation between images makes it possible to reconstruct a 3D model from 2D images. The section starts with an explanation of homogeneous coordinates, which because of their scale independence, are of great use for reconstruction, then describes the view correspondence techniques used, and ends with a summary of existing line reconstruction techniques. Section 2.4 explains geo-referencing which concerns the link between a 3D reconstruction and its real world coordinates. It starts with the coordinate system and height datum used in the Netherlands, and then introduces a method for coordinate transformation. The section ends with an explanation of various types of coordinate measurements used in photogrammetry. The section 2.5 explains quality measures used in this thesis. It provides definitions for accuracy and precision.

2.1. Railway construction and monitoring

In 2018 ProRail managed 7,114 km of railway tracks over which trains travelled 164 million km, [13]. Almost the entire network is made of ballasted tracks, which are used for some structures like bridges as well as modern high speed tracks. Ballasted, or traditional, tracks are constructed out of different layers. From top to bottom each layer has his own function to divide the forces a train exerts to the ground as seen in figure 2.1.

2.1.1. Construction

First, the point loads of the train wheels are horizontally distributed by the rails. In the Netherlands, 3 types of rail types are used: NP46 is the traditionally used rail type, it is only used found in older existing parts of the network. As a replacement, the internationally approved UIC54 is now used for regular tracks. The third type, UIC60 is used on high speed rail and some test tracks. All rail types have a similar shape, as shown in the cross section in figure 2.2, but slightly different dimensions. Table 2.1

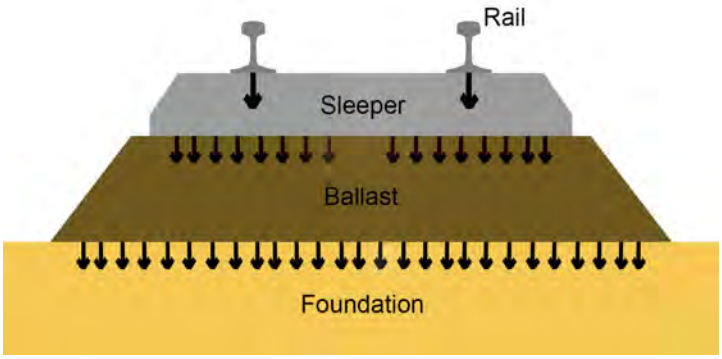


Figure 2.1: Cross-view of a typical rail bed construction with its force distribution. From top down the rails spread loads via the sleepers, ballast and finally in the foundation.

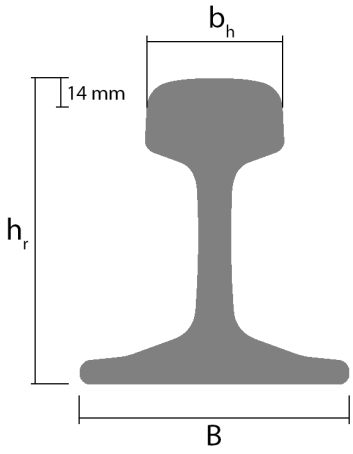


Figure 2.2: Cross-section of a generic rail with the parameters as shown in table 2.1, important parts are the height of the rail as well as the head width.

Profile		NP46	UIC54	UIC60
Height [mm]	h_r	142	159	172
Head width [mm]	b_h	72	70	72
Foot width [mm]	B	120	140	150

Table 2.1: Rail type specifications used in the Netherlands

shows the key dimensions of the rail types. The head width is defined as the width of the profile at a height 14 mm beneath the top. This height is also used to determine the track gauge, which is defined as the distance between the two inner points of both rails. In the Netherlands, as of most of Europe and the world, the gauge is 1,435 mm, which according to EN 13848-1 in practice should be in the range of 1,430 and 1,450 mm, [14]. Height is also used to determine cant, which is the difference in height between the left and right rail. This differences causes the railway to have an angle, which is useful in bends to direct the centripetal forces of the train to the foundation. Without cant, trains would derail cornering at high speeds.

The rails are kept in place by connecting them to sleepers. Different types of sleepers are found in the Dutch rail network, in both concrete and wood. The concrete monoblock sleeper is the most common type, which has the shape of a beam with a slightly narrower and lower middle part, [5]. In railway switches a solid beam is commonly found.

The sleepers are placed in the ballast, and opposed to the rail-sleeper connection, there is no construction to keep the sleepers in place. The ballast is made of loose rocks, but with enough roughness to

create friction. The rocks are very suitable to spread the vertical load to the foundation below, however it is the roughness which delivers the horizontal stability, [5]. Finally the ballast spreads the load over rail bed foundation. This is either a concrete construction or a bed of sand or gravel.

2.1.2. Monitoring

Partly because of the loose connection between sleepers and ballast, monitoring is required to check for deformations in the rail. The complete railway network is regularly monitored by contractors of ProRail to check for parts which show wear or displacement. To guarantee a constant level of quality measurements, ProRail created a guideline RLN00296 which specifies the requirements for monitoring elements of the so called PVS, covering the geometry of the rail network. This guideline requires a measurement every 2 meters with a maximal standard deviation of 15 mm on the XYZ coordinates of a track, 10 mm on the Z coordinate of switches, and 3 mm on the track cant. All measurements for ProRail have to be in the Dutch 'Rijksdriehoek' (RD) coordinate system with the Z-coordinate in the vertical datum 'Normaal Amsterdams Peil' (NAP).

Traditionally monitoring of railway infrastructure is done with well known surveying techniques, for example triangulation, levelling and GPS (Global Positioning System) campaigns. All can deliver highly accurate measurements, but do so in sparse points and take a lot of time to measure. Terrestrial laser scans provide dense measurements in the form of a pointcloud, but are expensive. More and more measurements are done with train based sensors such as a laser scanner mounted on the train. Their main advantage is the reduced presence of people on and around the tracks and the usage of trains riding their usual route. Disadvantages are problems regarding geo-referencing a moving scanner with GPS. Post-Processed Kinematics (PPK) solutions are preferred but hard to obtain with large monitoring areas. Also planning of mounting scanners on trains in practice require a lot of time and certificates.

2.2. Photogrammetry

Photogrammetry is the science of measuring objects by the use of photographic images. It involves all processes necessary to convert images to a form where information like distances can be determined, [6]. In this research these images are taken by a Unmanned Aerial Vehicle (UAV) mounted camera which, together with air-plane, helicopter or balloon based pictures, is called aerial photogrammetry. Common usage of aerial photogrammetry contains Digital Elevation Model (DEM), Digital Surface Model (DSM) and map generation, but it is also used in landslide monitoring, canopy mapping and soil bulk volume estimation.

Most applications of geoscience based Photogrammetry used a process called Structure-from-Motion (SfM), sometimes together with a Multi-View-Stereo (MVS) algorithm. SfM is build around using a set of highly overlapping images to create a 3D model of the area. The difference with more traditional photogrammetry methods is the lack of required prior information. SfM does not require camera calibration parameters, camera locations or selected matching points as an input but will determine all parameters itself, [27].

To do this a SfM pipeline usually has the following components:

- Image acquisition
- Feature extraction and matching
- Bundle adjustment

These components are explained in detail below. The camera parameters are the geometric components used to relate 2D pixels on a photo sensor with 3D points in the real world. Image acquisition is

explained to understand what is required for a good set of photos for reconstruction. Feature extraction and matching describes existing methods to find notable points in each image as well as method to match those points between different images. Bundle adjustment finally describes the iterative process used to estimate both the camera parameters as well as 3D points. The result is a sparse 3D model in image scale, which has to be scaled and transformed to real world coordinates in order to be useful for measurements.

2.2.1. Camera parameters

Each photo contains points with a 2D pixel location. Reconstruction of the 3D environment can be visualized as casting a ray from the camera centre through each pixel in the photo. Given different cameras containing the same environment, a 3D point is found by intersecting rays of the same object pixel. To be able to cast these rays, both the location and orientation of the camera has to be known, as well as the transformation from a pixel location to a real world point. This information is defined in the internal and external camera parameters.

External camera parameters The external parameters contain both the position and orientation of the camera. The 3D vector C being the location of the camera centre and a [3x3] matrix R_c being the rotation around its centre and thus the intrinsic rotation matrix which are visualised in figure 2.3.

Both are combined in an [3x4] extrinsic matrix $[R|t]$ which is a concatenation of the extrinsic rotation matrix R and the translation vector t. Both are computed from the intrinsic parameters using:

$$R = R_C^T, \quad t = -RC \tag{2.1}$$

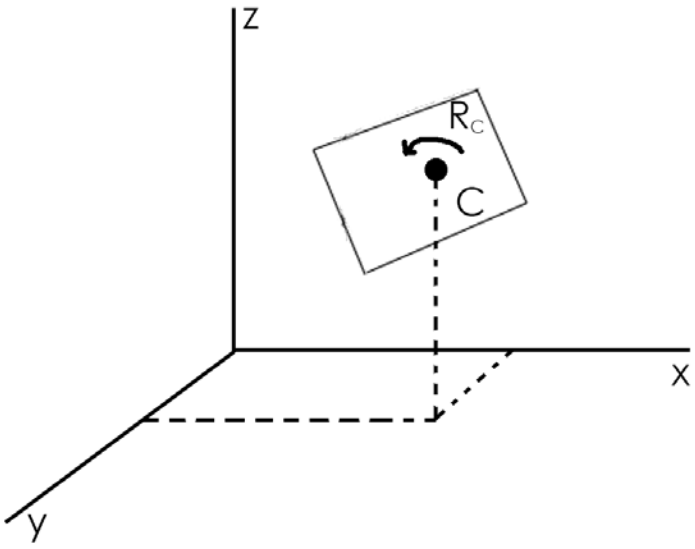


Figure 2.3: External camera parameters: intrinsic rotation R_C and camera center C

Internal camera parameters The internal parameters provide, as the name suggests, information of the cameras inside. This is stored in the camera matrix K which contains both sensor properties as size, resolution and principle point, and the lens' focal point. The matrix K is given by:

$$K = \begin{bmatrix} \alpha_x & 0 & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \alpha_i = f \cdot m_i \tag{2.2}$$

Here is α_i the focal length in pixels for both the x and y direction, which is calculated using the focal length f in millimetres and a scaling factor m_i being the amount of pixels per mm on the sensor. u_0 and v_0 are the pixel offset to the principal point of the sensor, which for a perfect camera would be in de sensor centre. Figure 2.4 visualizes these parameters.

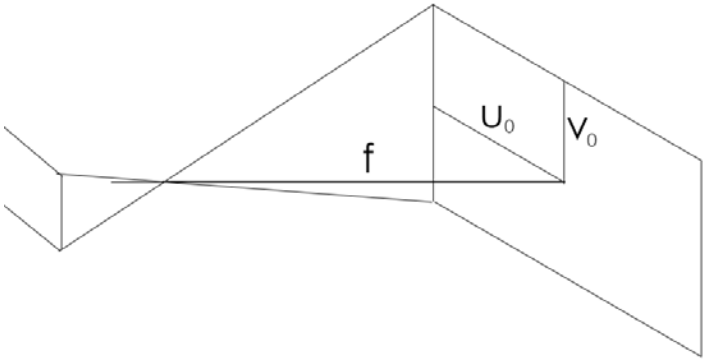


Figure 2.4: Internal camera parameters: focal length f, and offset u_0 and v_0.

A camera lens has another property besides the focal point: lens distortion. In a perfect pinhole camera, every real world point is projected in a straight line to the camera sensor. But due to optical and manufactural imperfections this is not true for actual cameras. Radial distortion is caused by lens imperfections and is divided in two categories: barrel and pincushion distortion. Both are named to the shape of their respective distortion patterns, which are shown in figure 2.5. It can be mitigated by calibrating the camera. This estimates parameters based on a distortion model which describes the error of each pixel as a function of their x and y coordinates.

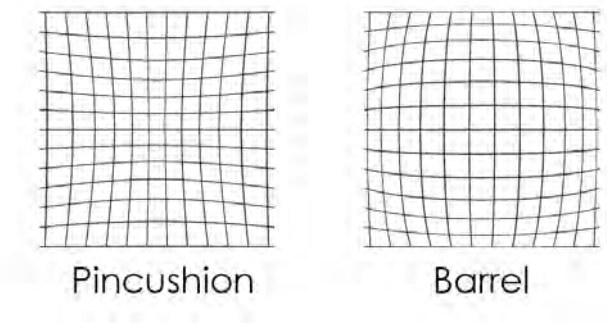


Figure 2.5: Barrel and pincushion distortion, showing the apparent smaller and larger middle of an image compared to its surroundings

Projection matrix The projection matrix **P** gives the relation between a 3D point in the real world and a 2D point on a camera sensor, [8]. It is the combination of both the internal and external camera parameters and therefore follows from equations 2.1 and 2.2:

$$\mathbf{P} = K[R|t] \tag{2.3}$$

The result is a 3x4 projection matrix **P**, which projects the 3D point **x** to the 2D point $\mathbf{x}$ via:

$$\mathbf{x} = \mathbf{P}\mathbf{x} \tag{2.4}$$

As the formula to obtain the fundamental matrix shows in section 2.3.2 it is sometimes useful to rewrite **P** as a matrix-vector combination which divides in a calibration and transformation section:

$$\mathbf{P} = [M| - MT], \quad M = KR, \quad T = -R^T t \tag{2.5}$$

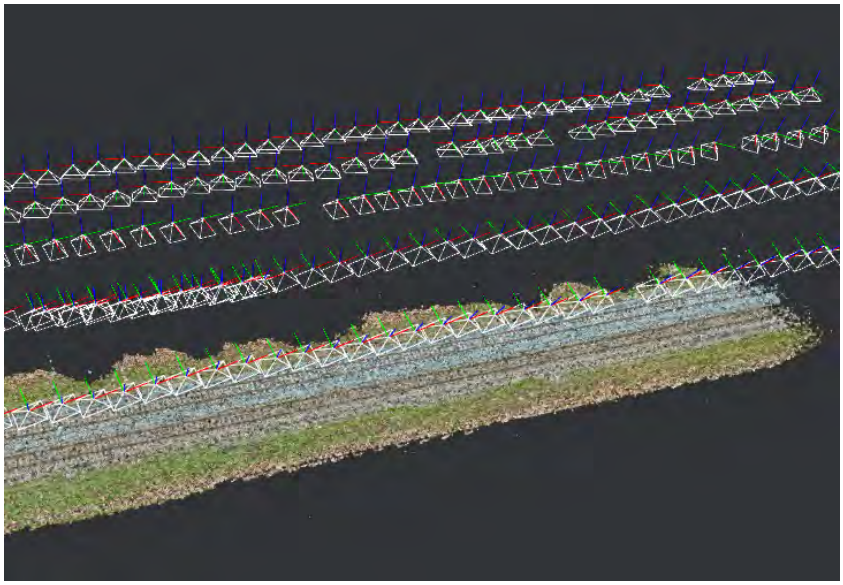


Figure 2.6: Image acquisition used for the Stroe dataset. Camera locations are the top of each pyramid which indicates the orientation of the camera. An oblique method is used, pointing the camera towards the subject.

Calibration External camera parameters are obtained by a combination of an inertial measurement unit and GNSS (Global Navigation Satellite System) measurements but are usually part of the estimates following a SfM workflow. Internal camera parameters are estimated by camera calibration which can be divided in three methods: Lab calibration, camera calibration using a test field and camera self-calibration. The first is the most accurate. In a lab special equipment is used to determine the relation between a viewing ray and the camera sensor. Despite being the most accurate this method is not often used because of its time consuming and costly nature.

The seconds method uses a 2D or 3D test field with known specifications. One the most simple forms being a chequerboard picture where the size of each rectangle is measured beforehand. One (for a large 3D test field) or more pictures are than taken of the board after which the keypoints in each photo are either manually or automatically determined to find the calibration parameters.

The last and most used method is the self-calibration where the calibration parameters are estimated as part of the bundle adjustment algorithm. This method is most used because of the requirement of the other methods to have a stable camera. A slight change in the set focus, focal range or in extreme cases even temperature can render the calibration parameters useless. For camera self-calibration the calibration parameters are determined for every picture and therefore counteract slight changes in the camera parameters, [6].

2.2.2. Image acquisition

A well thought image acquisition method is essential for creating an accurate 3D model. Without the correct photos it is possible a model cannot be created at all. Important parts of the acquisition plan are: flight plan of the UAV, nadir or oblique photos, amount of overlap between photos and the usage and placing of ground control points. Choices on these points depend on the required accuracy, subject to be scanned and the software which is available. Also important is the flight height in relation to the camera specifications. This determines theoretical accuracy described in Ground Sampling Distance (GSD). The GSD is the distance between samples on the ground which in case of photogrammetry is described as the distance covered by a single pixel. It depends on the distance between camera and subject. For UAV photogrammetry this translates to flight height and the focal length used which determine the angle of view and the size of a pixel on the sensor.

Figure 2.6 shows the image acquisition geometry of the Stroe dataset. An oblique method is used where the camera is oriented towards the subject. The pyramid outlines show the camera rotation and the acquisition position is located on the top of each pyramid.

2.2.3. Feature extraction and matching

After acquiring the set of photos, their relation has to be estimated. A computer just sees a set of unrelated pixels and does not see a relation between any pixels like humans do. Therefore each photo is processed to find a set of distinguishable areas where each has an unique descriptor.

The process of finding these areas is known as feature extraction. Features could be of any shape or size, but right now the most common methods are point based. Numerous methods exist such as SIFT, SURF, ORB, BRIEF. Each has a different way to find points in photos with distinctive features. Most methods try to find features which are robust to scaling, rotation and affine transformation, and are a trade off between the number of key-point and computational cost.

The most well known, and possibly most used, methods is SIFT, developed in 2004 by D.G. Lowe which is an abbreviation of Scale-Invariant Feature Transform. It is based on the calculation of Difference of Gaussian (DoG) on different scales. DoG is a approximation of the Laplacian of Gaussian which is a method often used to detect blobs in images. When a blob is detected on different scales on the same location the neighbourhood of the blob is stored as a unique descriptor of the key-point, [21].

With the features extracted each photo has a list of key-points with descriptors. Matching is done by comparing the lists and finding the corresponding matches. This will be rarely exact on all descriptors and therefore some margin is introduced in matching. This will also cause some false positives as well in the form of matches between key-points which are in reality not related. False positives are removed with a process called outlier detection.

2.2.4. Bundle adjustment

Bundle adjustment is the most commonly used algorithm to estimate the 3D position of keypoints found in the given set of images. This estimation includes both the position of the points as well as the external and internal camera parameters of the images. The external parameters cover the position (x, y, z) and the orientation (pitch, roll, yaw) of the camera. The internal parameters consist of the focal length, principal point and lens distortion of the camera.

The bundle adjustment algorithm starts with a list of images and a list of features. The algorithm selects pairs of images based on the amount of shared features. These pairs are than iteratively processed solving for the unknown 3D position of shared features and the camera parameters. The algorithm is iterative because of its non-linear model and is usually solved by assuming local linearity, [6].

2.3. Reconstruction

Reconstruction is the process to create a virtual model of a real environment. The model is sometimes also be called a digital twin of the real world. Traditional photogrammetry results in a reconstruction of the object which is photographed, where the virtual model is represented in a point cloud. Laser scans also reconstruct models in point clouds.

This section focusses on the 3D reconstruction of lines. It starts with a representation of lines in homogeneous coordinates. First in 2D as found in images and then in 3D as used in reconstruction. Next view correspondences explain several ways to describe geometric relations between images. These relations are used in the next chapter both to restrict the search area for matching lines between images

	2D	3D	Transformations
inhomogeneous	x	X	R
homogeneous	$\mathbf{x}$	$\mathbf{X}$	H

Table 2.2: Used notation for homogeneous and inhomogeneous coordinate entities.

and the reconstruction of these lines in 3D.

Then geo-referencing is explained where the digital model is scaled to match real world coordinates. This provides information about the coordinate system used for railways in the Netherlands and methods to transform between coordinate systems.

2.3.1. Line representations

Homogeneous coordinates Homogeneous coordinates are a way of representing geometric entities such as points and lines. They are defined as being invariant to multiplication by any value $\lambda \neq 0$, [6]. This means all points $\lambda\mathbf{x}$ with $\lambda \neq 0$ are coordinates of the same entity. The relation between homogeneous and Euclidean coordinates is defined by the multiplication λ, which is also called the scale factor. For a 2D Euclidean point x the relation is be written as:

$$x = \begin{bmatrix} x \\ y \end{bmatrix} => \mathbf{x} \begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix} => x = \begin{bmatrix} x/\lambda \\ y/\lambda \end{bmatrix} \tag{2.6}$$

Notation is chosen to follow Wröbel, [6]. Here inhomogeneous coordinates are displayed in bold italic and homogeneous in bold as visible in table 2.2.

There are several advantages of using homogeneous instead of Euclidean coordinates. The most important is their possibility of straight line preserving transformations such as translation and rotation but also projection. This last characteristic is particularly useful in photogrammetry as it simplifies projection of 3D points to the 2D camera sensor to a matrix-vector multiplication. A second practical characteristic for photogrammetry is the possibility of points in infinity. Where $\lambda \neq 0$ is required to have a relation between a homogeneous and euclidean point, choosing $\lambda = 0$ gives a point in infinity which is used for calculating epipolar lines.

Lines can also be represented in homogeneous coordinates. In a 2D Euclidean space a line is represented in its Hessian form as follows:

$$x \cos\phi + y \sin\phi - d = 0, \tag{2.7}$$

where ϕ is the angle with positive x-axis and d the orthogonal distance of the line to the origin as shown in figure 2.7. In its Hessian form a line is per definition of infinite length as no endpoints are represented in its coordinates.

For matrix-vector multiplication the Hessian form has to be rewritten as a vector. This is straightforward by storing x and y in a homogeneous point $\mathbf{x}$ which makes it possible to rewrite the Hessian form to:

$$\mathbf{x}^T\mathbf{l} = 0, \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \mathbf{l} = \begin{bmatrix} \cos\phi \\ \sin\phi \\ -d \end{bmatrix} \tag{2.8}$$

This shows the line $\mathbf{l}$ can be written as a 3-vector with the three Hessian components $\cos\phi, \sin\phi$ and d which is invariant to scale as multiplying $\mathbf{x}^T\mathbf{l}$ with some λ will be equal to $\lambda \cdot 0 = 0$ for any $\mathbf{x}$ on line $\mathbf{l}$. A line vector $\mathbf{l}$ is rewritten to its Hessian form by taking out the multiplication via:

$$\mathbf{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \pm\sqrt{a^2 + b^2} \begin{bmatrix} \cos\phi \\ \sin\phi \\ -d \end{bmatrix} \tag{2.9}$$

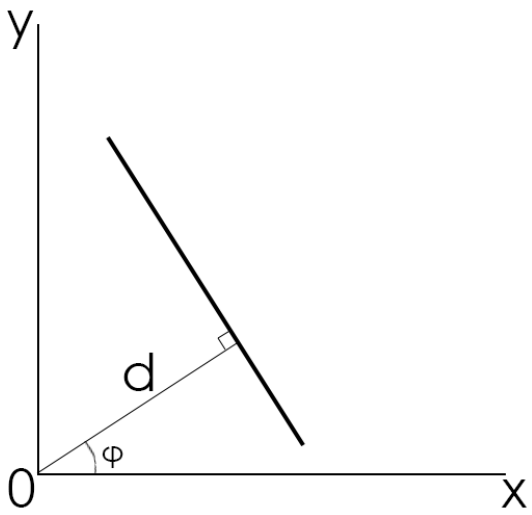


Figure 2.7: A 2D line in Euclidean space, with its parameters ϕ and d

Using its three parameters a, b and c, the Hessian parameters of line $\mathbf{l}$ are found using:

$$\phi = \text{atan2}(b, a), \quad d = -\frac{c}{\sqrt{a^2 + b^2}} \tag{2.10}$$

In 3D an extra dimension has to be added. For Euclidean points this is straightforward and similar to their 2D variant:

$$X = \begin{bmatrix} x \\ y \\ z \end{bmatrix} => \mathbf{X} = \begin{bmatrix} \lambda x \\ \lambda y \\ \lambda z \\ \lambda \end{bmatrix} => X = \begin{bmatrix} x/\lambda \\ y/\lambda \\ z/\lambda \end{bmatrix} \tag{2.11}$$

Homogeneous line coordinates are somewhat different. A 3D line L is defined using four parameters in 2 ways:

As an intersection of two planes which can be rewritten as the combination of the intersection lines between the planes and the XY and XZ planes therefore requiring two parameters per plane.

As a connection between two 3D points. Similar to planes this can be rewritten as a combination of plane intersection and therefore requires the x and y coordinates of one point and y and z of the other.

To be able to form a homogeneous system Plücker came up with a homogeneous vector notation using 6 parameters constructed of 2 orthogonal 3D vectors, [6]. Figure 2.8 shows a visualization of a Plücker line. Vector $\mathbf{L}_h$ represents the direction of line L. Vector $\mathbf{L}_0$ is called the moment of L and is obtained by the cross product between the line direction and any point **P** on this line. It is a vector crossing through the origin and is perpendicular to both L and its orthogonal vector through the origin. The redundant 2 parameters are constraint by the requirement of both vectors being orthogonal, called the Plücker constraint.

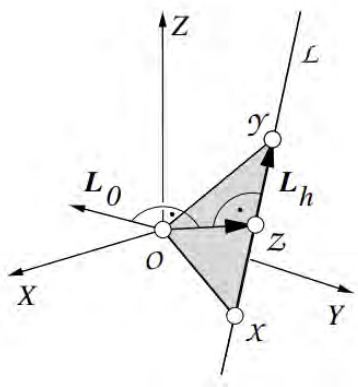


Figure 2.8: Visualization of Plücker vectors L and $\mathbf{L}_0$ as drawn by Wrobel, [6]

Given two homogeneous points **X** and **Y**, their connecting Plücker line is constructed as follows:

$$L = \begin{bmatrix} \mathbf{L}_h \\ \mathbf{L}_0 \end{bmatrix} = \begin{bmatrix} t_x \begin{bmatrix} u_y \\ v_y \\ w_y \end{bmatrix} - t_y \begin{bmatrix} u_x \\ v_x \\ w_x \end{bmatrix} \\ \begin{bmatrix} u_x \\ v_x \\ w_x \\ t_x \end{bmatrix} \times \begin{bmatrix} u_y \\ v_y \\ w_y \\ t_y \end{bmatrix} \end{bmatrix} \tag{2.12}$$

This 3D homogeneous line vector is a basis of this thesis as it gives the possibility to linearly project L to a 2D form $\mathbf{l}$. This is used to estimate reconstruction later in section 3.2.3.

2.3.2. View correspondences

The previous section mentions the relation between a 3D object and a 2D object on the sensor of a camera. As reconstruction requires multiple views of the same object, relations between these views help in the reconstruction process. These relations are for example used in the matching algorithm of section 3.2.

Fundamental Matrix. The fundamental matrix F describes the relation between points in two images. For each pair of points $\mathbf{x}_i$ in image i and $\mathbf{x}_j$ in image j, their relation is written as:

$$\mathbf{x}_j^T \mathrm{F} \mathbf{x}_i = 0 \tag{2.13}$$

The right part of this equation $\mathrm{F}\mathbf{x}_i$ gives the epipolar line of $\mathbf{x}_i$ in image j. This line contains all possible projections of $\mathbf{x}_i$ on j, [8].

Deduction of the fundamental matrix is based on the epipolar line equation. Two 3D points are chosen which both project to the same 2D point on image i. The first point is the camera location T_i of i and the second is an arbitrary point **x** in infinity (a point with a zero value scale factor: $t_x = 0$). Rewriting the projection matrices using the rotation M and translation T parts as described in section 2.2.1 give projection matrices $P_i = [M_i | -M_i T_i]$ of image i and $P_j = [M_j | -M_j T_j]$ of image j. The projections $\mathbf{p}_{1j}$ and $\mathbf{p}_{1j}$ of the camera location T_i and the arbitrary point in image j are given by:

$$\mathbf{p}_{1j} = M_j(T_i - T_j), \quad \mathbf{p}_{2j} = M_j M_i^{-1} \begin{bmatrix} u_x \\ v_x \\ w_x \end{bmatrix} \tag{2.14}$$

A line is defined by the cross product of both points $\mathbf{p}_{1j}$ and $\mathbf{p}_{1j}$. As both points project to the same point on image i,this defines the epipolar line:

$$l_e = M_j(T_i - T_j) \times M_j M_i^{-1} \begin{bmatrix} u_x \\ v_x \\ w_x \end{bmatrix} = M_j^{-1} M_i [M_i(T_i - T_j)]_\times \begin{bmatrix} u_x \\ v_x \\ w_x \end{bmatrix} \tag{2.15}$$

Here $[]_\times$ is a skew-symmetric matrix representing a cross-product as $\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b}$. The equation $l_e = M_j^{-1} M_i [M_i(T_i - T_j)]_x \mathbf{x}$) gives the epipolar line for each point $\mathbf{x}$, which is used to determine the fundamental matrix:

$$F = M_j^{-1} M_i [M_i(T_i - T_j)]_x \tag{2.16}$$

Trifocal Tensor Where the fundamental matrix describes the relation between points in two images, the trifocal tensor does something similar with three views. Given projections of a line or point in two images, the trifocal tensor gives the location of this line in the third view via:

$$l_k = \begin{bmatrix} l_i^T & T_1 & l_j \\ l_i^T & T_2 & l_j \\ l_i^T & T_3 & l_j \end{bmatrix} \tag{2.17}$$

Where T_n are 3x3 slices of the 3x3x3 Trifocal tensor. Given the projection matrices $P_i = [M_i | - M_i T_i]$, $P_j = [M_j | - M_j T_j]$, $P_k = [M_k | - M_k T_k]$ of images i, j and k via:

$$\mathbf{e}_i = M_i(T_k - T_i) \tag{2.18}$$

$$\mathbf{e}_j = M_j(T_k - T_j) \tag{2.19}$$

$$\mathbf{A} = M_i M_k^{-1} \tag{2.20}$$

$$\mathbf{B} = M_j M_k^{-1} \tag{2.21}$$

Which are then used to calculate each slice n of T, taking the n^{th} column of $\mathbf{A}$ and $\mathbf{B}$, [15]:

$$T_n = (\mathbf{e}_i \mathbf{B}_n^T)^T - \mathbf{e}_j \mathbf{A}_n^T \tag{2.22}$$

2.3.3. Existing methods for 3D line reconstruction

Point based features are mainstream and therefore many different techniques exist. But there has been some research in line based reconstruction as well. Baillard et al. proposed a method to reconstruct building roofs with aerial photos using line features, [1]. With sharp edges and large planar surfaces, roofs are an excellent example of line heavy objects. Their method uses geometric constraints for line matching between views together with pixel based correlation scores. Reconstruction is done using a planar search method where a rotating plane around a line is checked to match with other lines.

Extending on roofs, most man-made structures have plenty of lines. Schindler et al. proposed a method for street-view photos which, thanks to the lower viewpoint, uses vanishing point detection to classify lines being either vertical or horizontal, [17]. After manual matching the vanishing point classification is used as an extra constraint in the 3D reconstruction and optimization. Instead of the general 4 parameters required for line reconstruction, known orientation reduces this to 2.

Hofer et al. proposed a different method combining the matching and reconstruction problem, [9]. The matching is done purely based on geometric constraints. For all lines a set of possible matching segments is created by calculating the epipolar lines and looking for similar segments in other photos. Using the epipolar lines a set of possible matches is found, and the combination of all sets is then used to find a 3D reconstruction which optimizes the sets of matches, [9].

This thesis builds on the matching method proposed by Baillard, both utilising the geometric constraints to limit the set of possible matching and the pixel based correlation to find the best match. Because rails have a more complicated shape, the reconstruction method is not suitable. Instead reconstruction is done by implementing a method of Bartoli and Sturm which propose a method using Plücker lines for an iterative quasi-linear reconstruction, [2].

2.4. Geo-referencing

Geo-referencing is the process to link coordinates within a digital format in the form of either pixels, points or lines with coordinates in the real world. These coordinates should be specified in a coordinate reference system and a height datum, which in this thesis is the Dutch Rijksdriehoeksstelsel with NAP as a height datum. However, the known positions of the photos are obtained via GNSS measurements which use the WGS84 reference system. Therefore coordinate transformation has been applied. Several algorithms exist to apply this process, but all determine a transformation based on control points. This section will answer the following subquestion:

- Which methods are available to geo-reference the sparse cloud resulting from the SfM process?

Reference systems Coordinates express the location relative to a certain origin. In its most simple form a point is expressed in it's X and Y coordinate with an additional Z for vertical displacement. Reference systems define a way to describe how coordinates is determined based on origin, orientation and scale. On Earth coordinates are generally easier to write in a spherical model, where latitude and longitude describe a location on a sphere [22]. WGS84 uses coordinates based on a reference ellipsoid which approximates the shape of the Earth. Vertical datums similarly describe the relative height to an origin which usually is a surface defined as zero height.

The Rijksdriehoekstelsel is the coordinate reference system used in the Netherlands. It is a projection to a plane and therefore ignores the curvature of the Earth. Its origin is based in the city of Amersfoort, although this is not specified as the coordinate pair [0, 0] but has the coordinates [155000, 463000]. This shift has two reasons: all coordinates within the border of the Netherlands have positive values and for every point the y-coordinate is larger than the x-coordinate.

The vertical datum used is the Normaal-Amsterdams-Peil or Amsterdam Ordnance Datum, which is based on the average high water level in the river IJ in the 17th century. In the 19th century this was deemed to be the national height datum for the Netherlands and via several levelling campaigns a network of NAP points was created. During the last campaign between 1996 and 1999, a combination of optical and hydrostatic levelling was used in combination with GNSS measurements. This created a way to connect the orthometric height obtained by levelling, and the ellipsoidal height measurement by GNSS, [24].

Transformation and projection As mentioned before, the GPS-coordinates of the photos are given in the WGS84 reference system with EGM96 height datum, while the coordinates of the rails are required in the RD system with NAP height. This means the coordinates have to be transformed, and in the case of RD also projected.

The first step is to transform the coordinates from WGS84 to ETRS89. This is the reference system linked to the European stable plate. Because this plate moves relative to the WGS84 reference points, the coordinates shift over time. Both systems are connected by several realisations, the so called reference frames. The connection is specified as transformation parameters on a given time. To transform coordinates, first the shift caused by the time difference has to be corrected, after which the transformation is applied, [24].

Transforming between different systems is done with a 7-parameter transformation. This process, also called the Helmert transform, uses 7 parameters: a scale factor S, 3 rotation angles $\Omega_x, \Omega_y, \Omega_z$ which is rewritten as the product of 3 rotation axis combined in matrix **R** and 3 translation axis t_x, t_y, t_z which form the translation vector **t**. The Helmert transformed coordinates X', Y', Z' then follow from their original coordinates X, Y, Z via:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = S \cdot \mathbf{R} \cdot \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} + \mathbf{t} \tag{2.23}$$

The Helmert transform is also used for transforming the lines from image scale to real world RD/NAP coordinates. Helmert transform is regularly used in geo-referencing. It uses points with known coordinates to estimate the transformation parameters. These are then used to transform the whole point cloud accordingly. Major photogrammetry software developers such as Agisoft, Pix4D and Reality Capture use possibly different methods, but as all off them do not disclose their source or methods, it is not possible to implement.

Control points Control points are points in the dataset with known real world coordinates. These points should be easy to find in the data, and are usually obtained by taking accurate GNSS measurements or triangulation.

One can divide the control points in two groups, the most used are the ground control points (GCPs) which are markers placed on the ground. Figure 2.9 shows an example used for the Stroe dataset. Coordinates for these points are obtained either by triangulation from a known point, or using GNSS. Modern markers sometimes have a built in GNSS receiver.

Another type of control points are the so called ACPs or Air Control Points. This type is more often available in UAV based projects as the vehicle is usually equipped with a GNSS receiver. Using GNSS data, the location of the measurement device is known. Logging this information simultaneously with taking a photo creates a control point in the air.

A normal GNSS receiver has an accuracy in the order of meters, which is not adequate for high accuracy remote sensing. Therefore a technique called Post-Processed Kinematics is used, where simultaneously with UAV based measurements also GNSS measurements are obtained from a known location. Post-processing the combination of both measurements eliminates important error sources and enable an accuracy in the order of millimetres.

In photogrammetry, one of the products of Structure-from-Motion is the estimation of camera poses. Applying a transformation to these poses geo-references the resulting dataset.

2.5. Quality assessment

In order to assess the reconstruction quality, it is required to set some objective measurements to quantify the error of the used methods. Three quality measurements used in this thesis are accuracy, precision and data completeness.

Accuracy Accuracy describes the difference between measured data and the actual true value of that data. This difference is a combination of a bias and a random spread in measurements, and is measured via the sample mean square error, [23]:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - x)^2, \tag{2.24}$$

Figure 2.9: Ground control point example for Stroe dataset

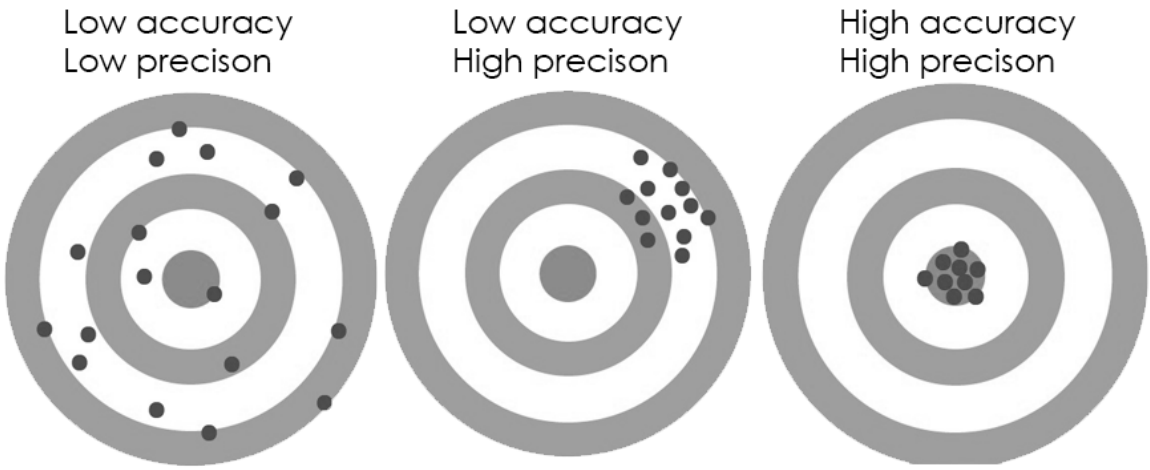


Figure 2.10: Accuracy and precision examples

where N is the amount of measurements, y_i the i^{th} measurement and x the true value. For readability, the root mean square is used, as the root value has the same units as the measured data.

Precision Precision is part of accuracy, as it is a measurement of the random spread. A dataset which contains a large bias, but very little spread, has low accuracy but high precision. Precision is sometimes called repeatability, as it is an indication of repeated measurements resulting in similar data.

2.6. Conclusion

This chapter introduced the parameters required for railway monitoring, as well as the current methods to monitor. It explained the parts found in rail infrastructure and which specifications are important for monitoring. Followed by an introduction to the researched monitoring method: photogrammetry. This section introduced the photogrammetry workflow from image acquisition to feature extraction and matching after which 3D reconstruction is done with bundle adjustment. It also introduced internal and

external camera parameters such as the rotation and calibration matrix, as well as their combination in the projection matrix, which are all important for methods used in this thesis.

The next section covered reconstruction which provided several methods to describe geometric relations between different images based on the camera parameters from the previous section. It also introduced homogeneous coordinates for both points and lines, which because of their scale dimension are very useful in reconstruction. The section geo-referencing described methods to transform a reconstruction in image-space to real-world coordinates. It also explained the transformation between WGS84 coordinates used by GPS, and RD-NAP coordinates used in the Netherlands. The final section provided methods to assess the quality of obtained results, where precision is the spread in error and accuracy is the combination of precision with a bias.

3

Methods for line reconstruction

This chapter provides the methods used to answer the research question and sub-questions. Like the sub-questions, this chapter is divided in 3 main parts:

- Line detection in original images
- Line matching and reconstruction
- Geo-referencing and validation

Following the flowchart of figure 3.1, each part describes how the input data is processed: Section 3.1: Line detection in original images starts with optimizing the images for line extraction, extracting the actual lines and filtering outliers. This also describes how Structure-from-Motion is used.

Next section 3.2 takes the filtered lines, and tries to match detected segments between images. First in pairs, and then using those pairs to find triplets. After removing suspected outliers, these matches are used to reconstruct a 3D model of the rails, again in two parts. First a linear method is used to generate a first estimate of the 3D position of each line. Then an iterative process is used to increase the quality of this estimate. Finally outliers again are removed from the data.

Geo-referencing and validation is covered in section 3.3 which starts with translating the filtered reconstruction from image coordinates to its RD/NAP real world location, which is the end product of this thesis. Then the part provides validation methods used to help answer the sub-questions by being able to give an quality estimate in line detection, reconstruction and geo-referencing.

3.1. Line detection in original images

The main idea of this thesis is to utilize very recognisable lines in images, which are not well covered by point based feature detection methods. Primarily used in image processing areas as line detection, a lot of research has been done on line detection algorithms and numerous methods exist with each their own benefits and problems.

In this thesis, the Line Segment Detector (LSD) as proposed by von Gioi is used, [7]. It is state-of-the-art, used by many projects and delivers sub-pixel accurate lines. Other advantages are the linear-time computation, the lack of parameters to be set, and build in false positive detection.

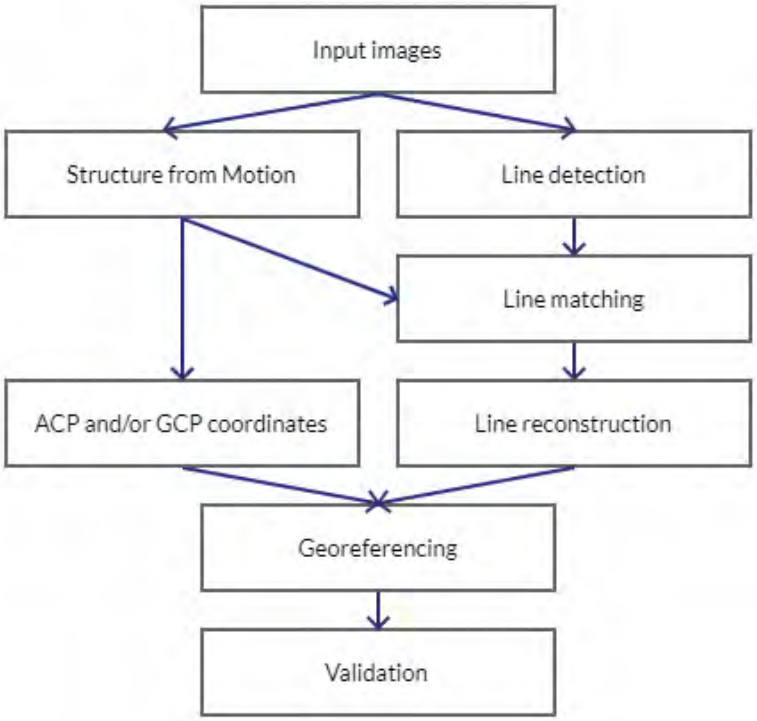


Figure 3.1: Flowchart of used methods. Input images are processed in both line detection and Structure from Motion, which are finally combined in a geo-referenced model

3.1.1. Pre processing

Before the actual line detection algorithm is run, the images are first processed to create an environment as easy as possible to detect lines in. First, the original images are undistorted. All cameras have imperfections in the lens and sensor, causing distortions in images. This is done with Structure-from-Motion (SfM) using the open source software Meshroom. Within Meshroom, a pipeline is created with functions starting with reading the images, then extracting SIFT features, matching these and finally reconstructing a sparse point-cloud model. The reconstruction is saved as JSON file which contains all camera orientations, internal parameters and matched SIFT points. The pipeline of Meshroom is finished by running a undistortion function which applies the found distortion parameters to retrieve the undistorted images.

After distortion removal the images are converted to a greyscale image because LSD requires a single channel image as input. This channel can be experimented with as there are many ways to scale RGB colours to a single greyscale. For this thesis, the build in method of OpenCV is used, [12] :

$$\text{Grey} = 0.299 \cdot \text{Red} + 0.587 \cdot \text{Green} + 0.114 \cdot \text{Blue} \tag{3.1}$$

Because of the way Meshroom processes images, their contrast is compressed while the image is saved as an EXR file which is made for high dynamic range multi channel raster data. To mitigate this, the image contrast is increased while converting the EXR to a JPG file. This is done by scaling the minimum and maximum found intensity to 0 and 255 respectively.

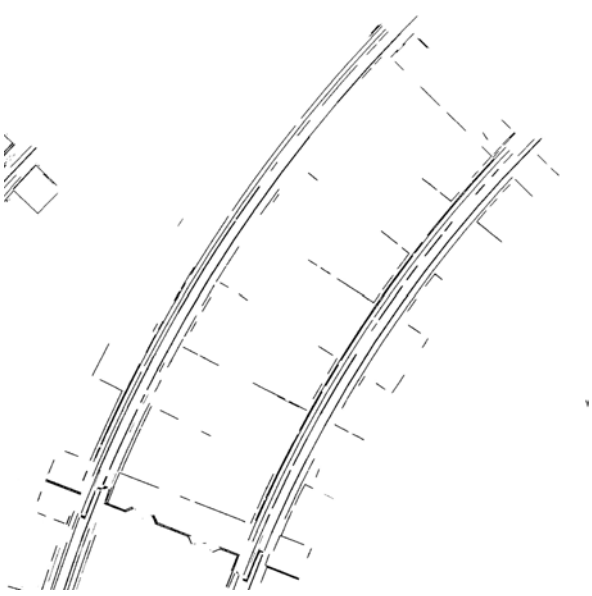
3.1.2. Line detection algorithm

With the image set prepared, lines are detected by implementing the LineSegmentDetector algorithm as proposed by von Gioi et al. [7]. This is based on region-growing, which iteratively determines an intensity gradient of a pixel, and then checks the surrounding pixels for a gradient in the same direction. When similar, the pixel is added to the region and the process is repeated.

Each resulting region is checked to have a rectangular shape and then is considered a candidate line segment. All candidates are then validated. The angles of pixel gradients are compared to the angle of the rectangle with a threshold. For an image with pixel values between 0-255 this threshold is set to about 2. The amount of aligned pixels is then tested in an a contrario approach which considers the probability of detection a pixel region with a similar amount of aligned pixels. The remaining line segments are returned and stored per image.

3.1.3. Post processing

Figure 3.2a shows an example output of the line detection algorithm. This shows a corner piece in which bends are displayed as series of small segments. The line detection algorithm detected rails as well as sleepers and some outliers. For the matching algorithms, longer segments are preferred to reduce matching time and non-rail outliers should be removed.



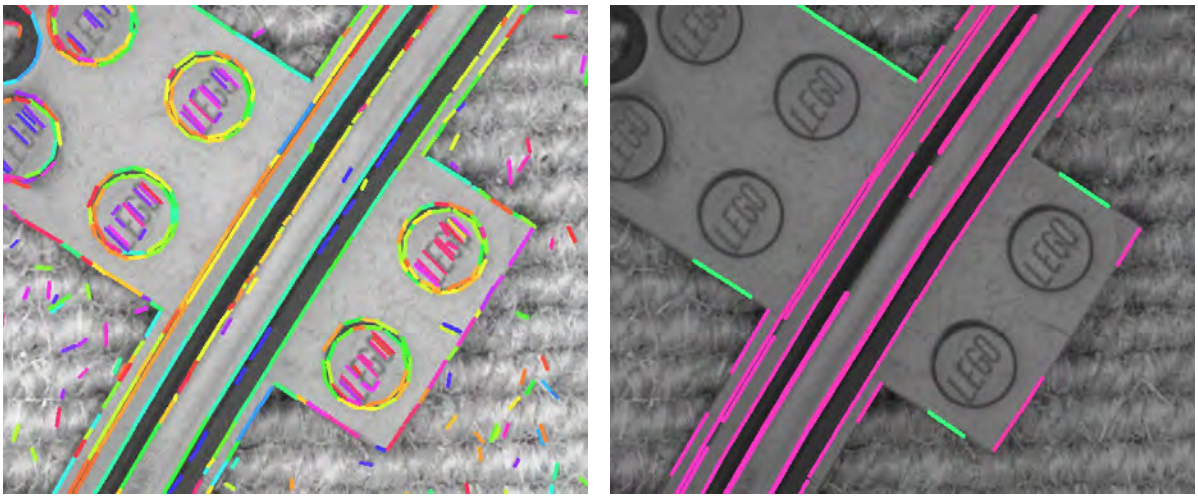
(a) LineSegmentDetector results of a Lego data-set image. The black line show the detected line segments which represent a Lego bend rail piece.

(b) The original image of the Lego data-set.

Figure 3.2: LineSegmentDetector results and the original image for the Lego data-set used as input.

First step is to remove all lines shorter than a set minimal length. The minimum value is chosen to be larger than most of the ballast rocks, removing these outliers. As all images are taken on roughly the same distance towards rails, the threshold is kept constant over all images. The second step is to merge collinear segments. This is done by computing the homogeneous line vector, explained in section 2.3.1. Homogeneous line vectors do not rely on endpoints, but are described by a direction and an offset. Therefore all segments with the same vector are collinear. To prevent two random segments which happen to be in the same direction from merging, a threshold has been set on the maximum distance between two segments. Figure 3.3b shows the processed result of a section of an image in the test dataset. In comparison with the original segments displayed in figure 3.3a most outliers have been removed and small segments have been merged in larger pieces. Downside is the disappearance

of some small segments on actual rail beams, which are filtered out.



(a) Line detection example on test data without post-processing. Many small segments are visible, for example covering the Lego dots. These should be removed before matching.

(b) Line detection example on test data after post-processing. Most non-rail outliers are removed by filtering, at the cost of some removed line segments on the rail itself.

Figure 3.3: Line detection example before and after post-processing.

3.2. Line matching and structure from motion

Reconstruction requires the same 3D line to be visible in multiple images. Therefore the line segments found in the previous section have to be matched to represent the same line. This is done by implementing a method proposed by Schmidt and Zisserman using geometric correspondences between the images, as found by the SfM process, to estimate possible matching segments and then comparing the surrounding pixels of each candidate to find the match with the highest correlation, [18].

3.2.1. Two view matching

In the two view matching process, pairs of images are compared both for and backward. The matching process is run both from image i to image j and from j to i. With a set of n images, this would result in $n(n-1)$ comparisons. To reduce comparisons a preselection is made based on point correspondences between images. These points are found in the sparse point cloud as a result of the SfM preprocessing. For each 3D point in the point cloud the images showing this point are known. With all points considered, a minimum count of correspondences restricts the amount of pairs.

With the filtered image pairs, the lines considered in each pair are filtered as well. For each line l_{im} in image i, the endpoints are projected on image j by the epipolar lines $\mathbf{l}_{eimk}$ as explained in section 2.3.2:

$$\mathbf{l}_{e,i,m,k} = \mathbf{F}_{i,j} p_{l_{i,m,k}} \tag{3.2}$$

Where $\mathbf{F}_{ij}$ is the fundamental matrix linking image i to j and $p_{l_{im}k}$ the $k = 0, l$ endpoints of line segment l_{im} with l the length of the line in pixels.

The area between epipolar lines $\mathbf{l}_{e,i,m,0}$ and $\mathbf{l}_{e,i,m,l}$ is called the epipolar beam. Every candidate match has to either intersect or lie within the beam. For each line satisfying this requirement, matching pixels are found by intersecting the epipolar line from each pixel k in image i with the candidate line l_{jm} in image j.

$$p_{l_{j,m,k}} = \mathbf{l}_{e,i,m,k} \mathbf{l}_{j,m} \tag{3.3}$$

Each intersection point $p_{l_{j,m,k}}$ is checked to lie between the endpoints $\mathbf{l}_{j,m}$ to make sure only overlapping

line segments are compared. For both points $p_{l_{i,m},k}$ and $p_{l_{j,m},k}$ a 15×15 pixel window is then obtained from images i and j. These windows W_i and W_j are then given a score based on their 2D correlation coefficient. W_{imn} and W_{jmn} are single pixels with coordinates (m, n) in the windows W_i and W_j. $\bar{W}_i$ and $\bar{W}_j$ are the average of all pixels within W_i and W_j:

$$\text{corr}(W_i, W_j) = \frac{\sum_m \sum_n (W_{imn} - \bar{W}_i)(W_{jmn} - \bar{W}_j)}{(\sum_m \sum_n (W_{imn} - \bar{W}_i)^2)(\sum_m \sum_n (W_{jmn} - \bar{W}_j)^2)} \tag{3.4}$$

This process is repeated for all pixels in the overlapping segments of $\mathbf{l}_{i,m}$ and $\mathbf{l}_{j,m}$. The final correlation score is determined by taking the mean of all windows. In two-view matching the best match is chosen by the line with the highest correlation score. In triple view matching, the top 10 two view scoring line are used as input in the algorithm.

3.2.2. Triple view matching

Triple view matching uses two view candidates and checks each candidate pair for matching in a third view. A selection on possible triplets is made to prevent three cameras to be collinear. This is done by dividing the cameras in strips parallel to the railway track. Triplets are limited to have cameras from at least two different strips.

Triple view matching with a triplet of images i, j and k is done by projecting lines from image i and j to image k. A line $\mathbf{l}_{i,m}$ in image i with matching line $\mathbf{l}_{j,m}$ in image j can be projected in image k using the Trifocal Tensor $\mathbf{T}$ explained in section 2.3:

$$\mathbf{l}_{k,m} = \begin{bmatrix} \mathbf{l}_{i,m} & \mathbf{T}_1 & \mathbf{l}_{i,m} \\ \mathbf{l}_{i,m} & \mathbf{T}_1 & \mathbf{l}_{i,m} \\ \mathbf{l}_{i,m} & \mathbf{T}_1 & \mathbf{l}_{i,m} \end{bmatrix} \tag{3.5}$$

Any candidate line in image k should ideally be on $\mathbf{l}_{k,m}$. However, due to small errors in the Trifocal Tensor and detected lines it is possible lines do not exactly overlap. Therefore all lines similar to $\mathbf{l}_{k,m}$ are considered a candidate, where similarity is checked based on direction of each line and the orthogonal distance of their endpoint to $\mathbf{l}_{k,m}$. The threshold is set to be 5 degrees in direction. The threshold on endpoint distance is normalised over the length of the line segment and is set to 9 pixels per 100 pixels. From the filtered candidates a match is chosen based on the average correlation score found in two view matching. The highest average correlation is picked as match.

3.2.3. Reconstruction

Reconstruction is the actual lifting of the found and matched lines in 2D images to a 3D environment. This is achieved by implementing a triangulation method proposed by Bartoli and Sturm, [2]. The idea is to find a 3D Plücker line $\mathbf{L}$ which minimizes the reprojection error. This error is given by the orthogonal distance of two points to the Plücker line reprojected to a 2D line on the original photo. The two points can be any points on the 2D line, and are chosen to be the endpoints. The orthogonal distance of a point $\mathbf{q}$ to a line $\mathbf{l}$ is calculated by:

$$d_\perp^2(\mathbf{q}, \mathbf{l}) = \frac{(\mathbf{q}^T \mathbf{l})^2}{(l_1^2 + l_2^2)} \tag{3.6}$$

Where l_1 and l_2 are the first and second value of the vector $\mathbf{l}$.

The reprojected 2D line vector $\mathbf{l}$ is calculated via $\mathbf{l} = \mathbf{P}_t \mathbf{L}$ where $\mathbf{P}_t$ is a [3×6] Plücker projection matrix which is calculated from the camera projection matrix $\mathbf{P} = [\bar{P}|\mathbf{p}]$ via $\mathbf{P}_t = [\det(\bar{P})\bar{P}|[\mathbf{p}]_\times \bar{P}]$.

Using the reprojected 2D line, an error margin is calculated with the distance of both endpoints of the reprojected line to the actual line with equation 3.6. The total error is taken to be simply the sum of

distances of both endpoints:

$$\text{reprojection error} = \sum_{i=1}^{n} \left(\frac{(p_1^{iT} \mathbf{P}_t^i \mathbf{L})^2}{((\mathbf{P}_t \mathbf{L})_1)^2 + ((\mathbf{P}_t \mathbf{L})_2)^2} + \frac{(p_2^{iT} \mathbf{P}_t^i \mathbf{L})^2)}{((\mathbf{P}_t \mathbf{L})_1)^2 + ((\mathbf{P}_t \mathbf{L})_2)^2} \right) \tag{3.7}$$

The best fitting 3D Plücker line **L** is then found by minimising the total reprojection error over all images.

Initial estimate The initial estimate of L is calculated by a least squares method. Therefore the error function has to be linearised, which is done by introducing a bias $w^2 = l_1^2 + l_2^2 = ((\mathbf{P}_t \mathbf{L})_1)^2 + ((\mathbf{P}_t \mathbf{L})_2)^2$. The biased least squares reprojection error function is then written as:

$$\text{biased reprojection error} = \sum_{i=1}^{n} ((p_1^{iT} \mathbf{P}_t^i \mathbf{L})^2 + p_2^{iT} \mathbf{P}_t^i \mathbf{L})^2) \tag{3.8}$$

Where p_1 and p_2 are two points on the projection of **L** on image i, for example the endpoints found by the line detection algorithm. This function can be rewritten as $||A\mathbf{L}||$ where A is a $2nx6$ matrix for n available views:

$$A = \begin{bmatrix} p_1^{0T} \mathbf{P}_t^0 \\ p_2^{0T} \mathbf{P}_t^0 \\ \vdots \\ p_1^{nT} \mathbf{P}_t^n \\ p_2^{nT} \mathbf{P}_t^n \end{bmatrix} \tag{3.9}$$

Solving this system gives an estimate for **L**. As **L** is homogeneous, an extra constraint is set as $||\mathbf{L}||^2 = 1$. This additional constraint creates the possibility of solving the system using Singular Value Decomposition. By decomposing matrix $A = U\Sigma V^T$, the estimate of **L** is found in the singular vector (columns of V) corresponding to the minimal singular value in Σ.

As the Plücker constraint is not enforced in the estimation, the resulting estimate is not necessarily a set of orthogonal vectors. Therefore an additional process is required to find the Plücker Line closest to the estimate. This process is called the Plücker correction. A method is provided by Bartoli and Sturm, [2], but a faster and easier to implement option is found by Cardoso, Miraldo and Araujo, [3]. Instead of using two singular value decompositions, this method is more straightforward, dividing **L** in two 3x1 vectors a and b:

$$a_* = \frac{1}{1-\alpha^2}(a - \alpha b) \quad b_* = \frac{1}{1-\alpha^2}(b - \alpha a) \tag{3.10}$$

$$\alpha = \frac{2p}{q + \sqrt{q^2 - 4p^2}} \quad p = a^T b \quad q = ||a||^2 + ||b||^2 \tag{3.11}$$

The resulting Plücker vector $\mathbf{L} = (a^T | b^T)^T$ has orthogonal vectors, and is therefore compliant to the Plücker constraint.

Iterative process While the linear algorithm provides an estimate for a 3D line, it is very susceptible for noise due to the bias. Therefore an iterative process is introduced by Bartoli and Sturm, which uses the linear algorithm to create an initial estimate. The iterative process is created by setting an additional constraint as $\mathbf{L}_k^T \mathrm{G} \mathbf{L}_{k+1} = 0$ where $\mathrm{G} = \left(\begin{smallmatrix} 0 & I \\ I & 0 \end{smallmatrix}\right)$ is a 6x6 matrix, and k is the iteration. For $\mathbf{L}_k == \mathbf{L}_{k+1} = (a_k^T | b_k^T)^T$ this is rewritten so to enforce the Plücker orthogonality:

$$a_k^T b_k + b_k^T a_k = 2 a_k^T b_k = 0 \tag{3.12}$$

Together with the normalisation $||\mathbf{L}||^2 = 1$ this is solved by a linear least squares optimization, for which the method is described by Hartley and Zisserman, [8]. This method is based on finding the orthonormal basis which gives all vectors compliant to the constraint and is equal to the null-space of $\mathbf{L}_k^T \mathrm{G}$. The null-space is found by extending $\mathbf{L}_k^T \mathrm{G}$ with rows of zeros to form a 6x6 matrix. Singular value

decomposition gives a single singular value, with a corresponding vector in V. The null-space is given by the remaining columns on V, vectors corresponding to a singular value of 0.

With null-space $\bar{V}$ the new estimate is computed by $\mathbf{L}_{k+1} = \bar{V}\gamma$. The best fitting 5-vector γ is found by inserting $\mathbf{L}_{k+1}$ in the linear method described in the previous section:

$$||A\mathbf{L}_{k+1}|| = ||A\bar{V}\gamma|| \text{ with } ||\gamma||^2 = 1 \tag{3.13}$$

The new estimate $\mathbf{L}_{k+1}$ is used to reweigh A by dividing the bias:

$$w^2 = ((\mathbf{P}_t\mathbf{L}_{k+1})_1)^2 + ((\mathbf{P}_t\mathbf{L}_{k+1})_2)^2 \tag{3.14}$$

This process is repeated until convergence. The threshold is determined by the change in reprojection error and is set to 10^{-6}

Outlier removal Incorrectly matched line segments between cameras result in reconstructed lines which do not correspond with any real world object. To prevent clutter in the reconstructed lines, two methods are introduced to remove outliers.

The first method is based on the reconstruction method, where in the selected triplet the reprojection error to each line segment is minimized. As an extra check the 3D reconstructed line is reprojected on surrounding images with the assumption this image has a line segment corresponding to this 3D line. A line segment is determined to be corresponding based on a check in line direction and distance as explained in section 3.3.2. When a threshold is met of 5 degrees in angle and 9 pixels per 100 pixels segment length in distance, the line segment is considered corresponding. When a line does not find corresponding segments in two other images, it is considered an outlier and therefore removed.

The second method is based on the expected direction of reconstructed 3D lines. Based on the geometry of rail infrastructure, where the length of an area of interest is much higher than both the width and height, it is expected the camera locations of a photo dataset follow the same geometry. This assumption is used to do a principal-component-analysis on the camera locations in order to obtain the three major axes of the area of interest.

The first axis corresponds to the direction of the rail, the second is horizontal perpendicular and the third is the vertical axis. Each 3D line is decomposed in these three axes, and are represented as percentages when divided by its length. When a line has a vertical component higher than 10 % it is considered an outlier and removed.

3.3. Geo-referencing and validation

Geo-referencing is required to compare the digital measurements with real-world ground-truth. Therefore validation and geo-referencing have been combined in the third part of this thesis.

3.3.1. Geo-referencing

The reconstruction process delivers a set of lines in a 3D space, however the coordinates in this space don't have any real world relevance. In order to extract usable information from the lines, both the scale as well as the real world location is required. Therefore geo-referencing is applied to scale, rotate and translate the coordinates from image scale to real world coordinates referenced in the Dutch Rijksdriehoekstelsel with NAP height. The Helmert transformation is used to translate, rotate and scale the image coordinates to real world coordinates.

Helmert transformation The Helmert transformation, or 7-parameter transform, is a well known method to transform coordinates and is also often used while transforming between different reference systems. The coordinates are subsequently scaled, rotated and translated following equation 2.23. According to its name, 7 parameters have to be calculated. Therefore, 3 points with known 3D points are required as a minimum for parameter estimation. Per dataset there are multiple points with known positions, either in de form of accurately measured ground-control-points or camera locations measured via GPS. This is used in a weighted least squares optimization following a method of Sjöberg, [20].

The input for the optimization is a set of coordinate pairs $(\mathbf{x}_i, \mathbf{y}_i)$ for $i = 1, 2...n$, where $\mathbf{x}_i$ and $\mathbf{y}_i$ are m-dimensional (in this case $m = 3$) coordinates of the same point i in different systems.

3.3.2. Validation

Validation returns the answers to most of the research sub-questions, as well as the main research question as it gives quantitative results on the methods described above. Each domain has its own form of validation, either by comparing to a different method applied to the same data, or by manually providing a ground-truth.

Line detection Validation of line testing has to answer the following sub-questions:

- At what success rate can rails be detected in both orthogonal and oblique 2D images using line detection algorithms while limiting false positives
- What is the pixel accuracy of found rail lines in the images, validates with manual inspection?

The success rate is a way to describe the fraction of success among a dataset. In this case ”success” are all the lines needed to reconstruct a rail in 3D. A way to describe the success rate would be to divide the amount of line found in the images by the lines present in the ground-truth. However, the lines found are split up in many smaller segments of which multiple can be part of a ground truth line. Also lines found by the algorithm not necessarily correspond to a ground truth line, and may be a so called false positive.

To combat this, a method of masking is used which is visualised in figure 3.4. For testing images, ground-truth lines have been drawn as white lines in a binary image (Only consisting of white ”1” and black ”0”). After this, the lines found by the algorithm have been drawn in the same image, but this time in a black colour. Therefore any parts of the ground-truth lines which are found by the algorithm will be black, where missing parts will remain white. For visualisation purposes, this is first drawn in yellow in figure 3.4 To leave some room for pixel errors, the found lines drawn have a larger line-width called a buffer. Therefore lines just next to the ground-truth line will also count as a success.

The success rate is then simply determined by comparing the white pixels in the image after masking n_m with the image only containing the ground-truth lines $n_g t$. The amount of pixels found by the algorithm follow from $n_s = n_g t - n_m$ which gives the formula for a success rate:

$$\text{Succes rate} = \frac{n_s}{n_g t} \cdot 100\% \tag{3.15}$$

The line detection performance is checked on both data completeness and accuracy. To be able to do this, a ground truth image is selected, and required lines are drawn in manually. The line detection result of the same image is than compared. All detected lines are assigned to the closest ground truth line in both direction and distance.

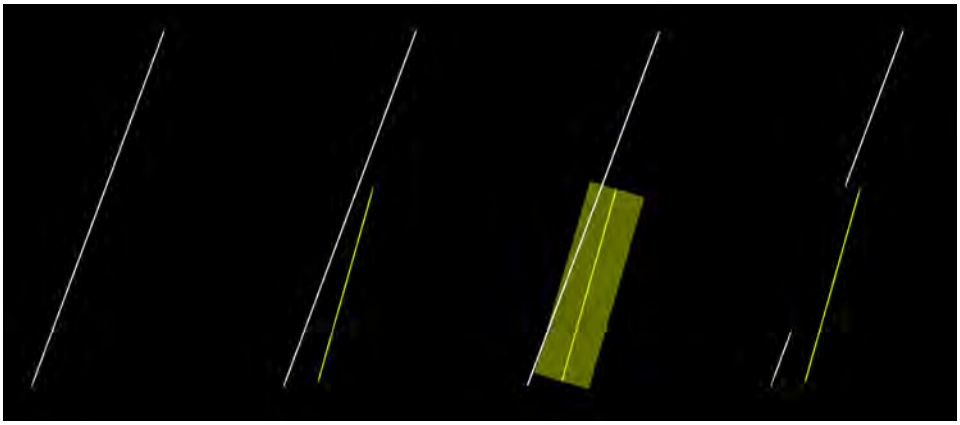


Figure 3.4: Masking method used for line detection validation, starting left with the ground-truth line in white. The segment found by LSD (yellow) is given a buffer (dark yellow). Overlap between buffer and ground-truth line is removed, after which the data-completeness is determined by comparing remaining white pixels with the original line.

Data completeness Data completeness is the percentage of the ground-truth lines which are retrieved by the line detection algorithm. Low completeness percentages indicate it will be impossible to reconstruct the complete railway. The data completeness is calculated by a buffer method. Each ground-truth line is defined to have a 1 pixel with and are drawn as a white line on a black image. All lines assigned to the ground-truth line are drawn on the same image in black. The width of these lines is given by twice the desired buffer size, which is chosen to be 2 pixels in this case. The parts of the ground-truth line which are not covered by a detected segment remain as white pixels in the image. The data-completeness is then given by:

$$\text{completeness} = \left(1 - \frac{n_{after}}{n_{before}}\right) * 100\%, \tag{3.16}$$

where n is the amount of white pixels found before and after drawing the line segments.

Accuracy The accuracy of the line detection is given by the error in both direction and distance between the detected segment and ground-truth line. A line is represented by various detected segments of different lengths, therefore the length is not taken into account as an error, and the distance is normalized over the length of a segment.

The difference in angle is simple calculated by subtracting the angles of both lines, which are calculated using the endpoints (x_1, y_1) and (x_1, y_1) :

$$\phi = \tan\left(\frac{x_2 - x_1}{y_2 - y_1}\right) \tag{3.17}$$

The error is then given by the absolute difference between the lines. Because in this case opposing line directions are considered equal, the value chosen is the minimum error to either 0 or π.

The error in distance is computed by taking the orthogonal distance of both endpoints of a segment to the ground-truth line. This is calculated by:

$$d_1^2(\mathbf{q}, \mathbf{l}) = \frac{(\mathbf{q}^T \mathbf{l})^2}{l_1^2 + l_2^2}, \tag{3.18}$$

where $\mathbf{q}$ is a 2D point given in homogeneous coordinates and $\mathbf{l} = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix}$ is the homogeneous line vector of the ground-truth line. The sum of distances is then divided by the segment length to normalize the error.

Geo-referencing Validation of geo-referencing has to answer the following sub-questions:

- Which accuracy and precision in the RD/NAP coordinate-system can be found on the GCPs comparing traditional SfM and the proposed method?
- Which accuracy, precision and data completeness in the RD/NAP coordinatesystem can be found on the rails comparing 3D rail detection in the traditional SfM point cloud and the proposed method?

Accuracy and precision are terms both explained in section 2.5. In short accuracy is the error margin between the measured data and the ground-truth, or the mean error. Precision is the spread of errors in the measured data, or the standard deviation of the error.

The first sub-question determines the accuracy and precision of the used transformation algorithm. Ground control points are accurately measured with GNSS measurements, and are considered ground-truth. These points are also visible in the traditional photogrammetry approach as a collection of points, and within the line based reconstruction where they are visualized as cross point between two lines drawn over the ground-control points.

The second question determines the accuracy and precision of the reconstruction method by comparing the reconstructed lines with the position of rails in the ground-truth. Laser-scans of the area of interest in this case represent the ground-truth. Using the algorithm of Sassen the position of the rails is obtained per 0.5 m, [16]. These points are then compared to the reconstructed lines, where the error is expressed by equation 3.6.

3.4. Tools

The following tools will be used during this research, subjective to change or extension. Listed software will be used on both a Dell Latitude with an Intel i7-8750H processor, 16 GB RAM, and a Nvidia MX 130 graphics card, and a HP Zbook G5 Studio x360 with an Intel i7-8750H processor, 16 GB RAM, and a Nvidia Quadro P1000 graphics card.

AliceVision An open source SfM-MVS software written in C++, highly customizable and command line support.

Meshroom A Python based GUI written for AliceVision

Python Scripting language, using libraries: Numpy, Json, Itertools, pyLSD.

OpenCV An open source computer vision library for both C++ and Python, has multiple options for line extraction, camera calibration and some other photogrammetry scripts

CloudCompare Software to view, compare and edit point-clouds Used for comparison with existing point clouds

4

Results

This chapter explains the results of the methods described in the previous chapter. Section 4.1 introduces the used data-sets together with some key properties. The following sections provide results divided per part:

4.2 Line detection in original images

4.3 Line matching and reconstruction

4.4 Geo-referencing and validation

The parts are split in subsections for each data-set and end with a short conclusion. Section 4.5 contains a deeper dive considering two cases: one where the methods work as intended, and one where they do not. Finally section 4.6 provides recommendations on issues found in section 4.5.

4.1. Data-sets

Two data-sets are used in this thesis. The main dataset is obtained using a DJI MATRICE 600 Pro unmanned aerial vehicle (UAV) with a Phase One iXU1000 camera capturing photos in a 11608 x 8708 pixel resolution. It is called the Stroe data-set. With the 80 millimetre focal length lens, medium format sensor and a 30 meter flight height, these images have a ground sampling distance of around 1.5 millimetre. For testing purposes a small scale data-set is created using a Canon 70d digital camera and is called the Lego data-set. It contains 32 pictures of a Lego railway track, with photo dimensions of 5472 x 3648 pixels. With a camera to ground distance of about 1 meter, a 1.6x crop sensor and a 50 mm lens, this results in a ground sampling distance of about 0.5 mm. Geo-referencing this dataset is not possible, however a 3D model was used to estimate relative accuracy.

Stroe A region of interest of around 350 m was captured in a set of 762 images with the camera angled towards the track. 12 Ground Control Points are available with 2.4 mm accuracy XYZ positions.

Lego A set of 32 images were captured with the camera angled towards the track. No reference points are available.

A Terrestrial laser scan point cloud is also available for the Stroe dataset, it is geo-referenced with the

same control points as used for the photogrammetry set. Because of its high accuracy, this is used as ground truth.

4.2. Line detection in original images

This part will answer the following sub-questions:

- At what success rate can rails be detected in both orthogonal and oblique 2D images using line detection algorithms while limiting false positives?
- What is the pixel accuracy of found rail lines in the images, validated with manual inspection?

4.2.1. Lego data-set

The line detection algorithm, including pre and post processing was run on all 32 images of the Lego data-set. Figures 4.1a and 4.1b show the results of the line detection algorithm on the Lego data-set. Figure 4.1a shows a lot of smaller segments spread over the whole image. Figure 4.1b shows that the post-processing does remove the smaller segments not belonging to rail infrastructure. It also merges parts of the lines to form longer segments, although not all of them. Curved rail parts are represented by small line segments, which brings the possibility to reconstructed curves as well as straight lines. On the downside, the algorithm also removed some of the found line segments which should be kept.

The results of line detection are validated using manually drawn ground-truth lines, as described in section 3.3.2. As the post-processing example in figure 4.1b shows, 61 of 106 found segments correspond with the actual rails. The other 45 lines are found on the sleepers. However, ground truth is specified in 4 lines per rail as explained in section 2.1. This causes many lines to be obsolete in success rate and pixel accuracy testing.

For the Lego dataset, a test is run on 4 images with ground truth lines drawn in. The line detection algorithm is run on these images with changing parameters:

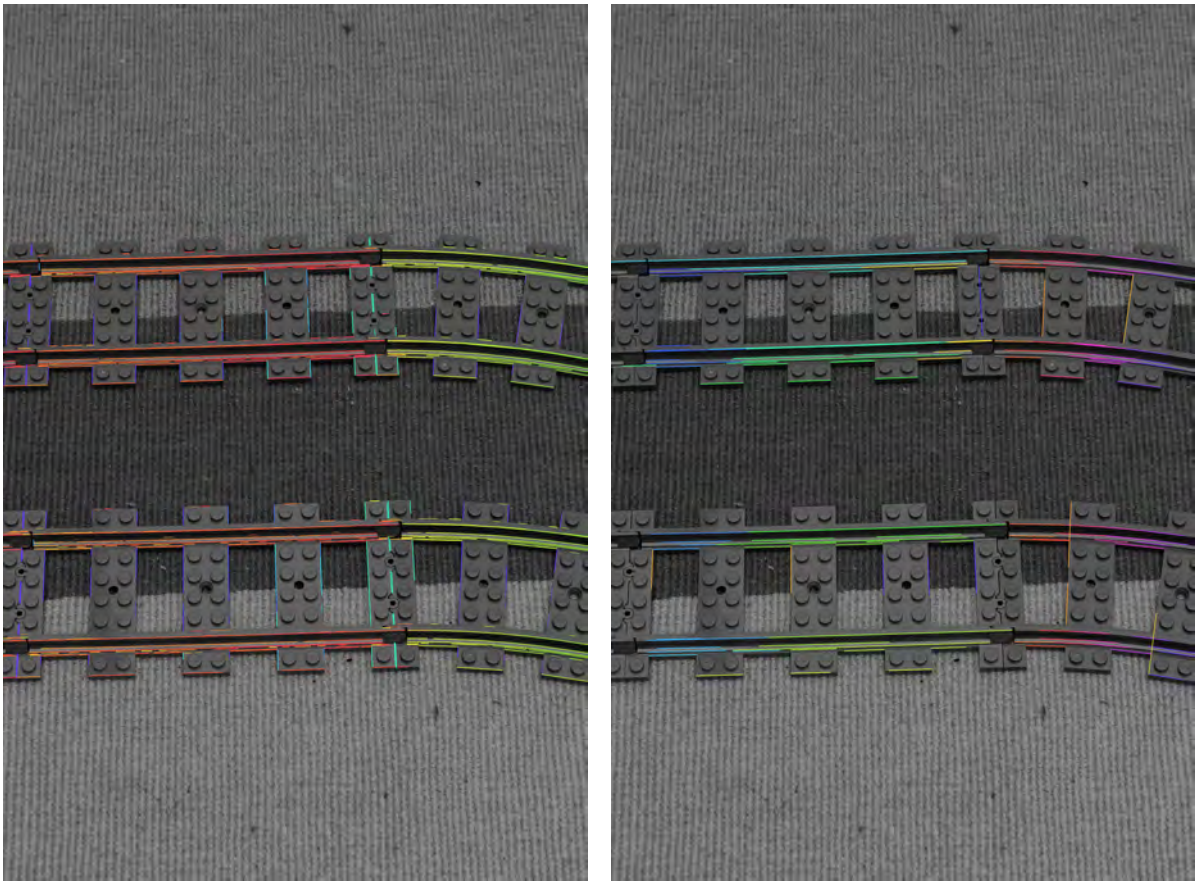
Minimal segment length - from 50 to 200 pixels in steps of 50.

Maximal segment length - from 50 to 200 pixels in steps of 50.

For all sets of parameters, the error in angle and distance between found segments and ground truth lines is measured, as well as data completeness. Figures 4.2a, 4.2b and 4.3, show the results of these measurements as a function of both the minimal line and maximal gap length.

Figures 4.2a and 4.2b show a decreasing error in both angle and distance with longer minimal length and maximal gap length, which is expected as merged segments smooth errors over a longer length. The minimal angle error was found with a segment length of 200 pixels, where the error rounds to 0.09 degrees. Results for a segment length of 150 pixels with a gap length of 200 pixels have the same error. The minimal distance error was lowest for the same parameter set, where it equalled an average error of 1.4 pixels, normalised to a 100 pixel length segment.

Data completeness in figure 4.3 shows a different story, where the results decreased for increasing parameter values. Stricter filters result in more segments to be discarded from the data, therefore resulting in areas without any segments. The best result was found for a segment length of 50 pixels with a maximal gap of 200 pixels, where 45 % of the lines were detected. The results show the chosen parameters for minimal segment length and maximal gap length are a compromise between data completeness on one side and line accuracy in the form of angle and distance error on the other



(a) Line detection on Lego data without post-processing, this result contains 12844 line segments of which 2856 correspond to the actual rails.

(b) Line detection on Lego data after post-processing, this result contains 106 line segments of which 61 correspond to the actual rails

Figure 4.1: Line detection on an example image from the Lego data-set both before (a) and after (b) post-processing. Lines are coloured to show segments with similar directions.

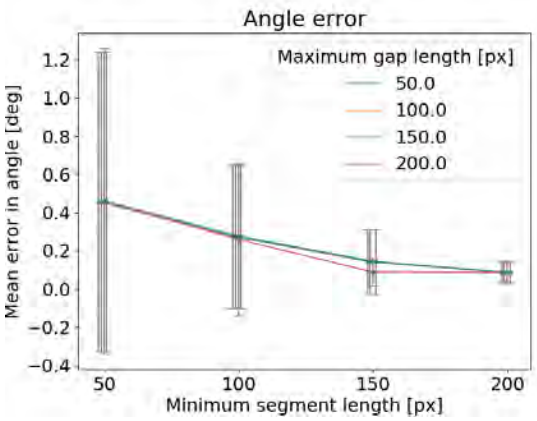
side. Therefore the chosen parameters are 100 pixels for both parameters as they removed a lot of the outliers and limit the decrease of data completeness.

4.2.2. Stroe dataset

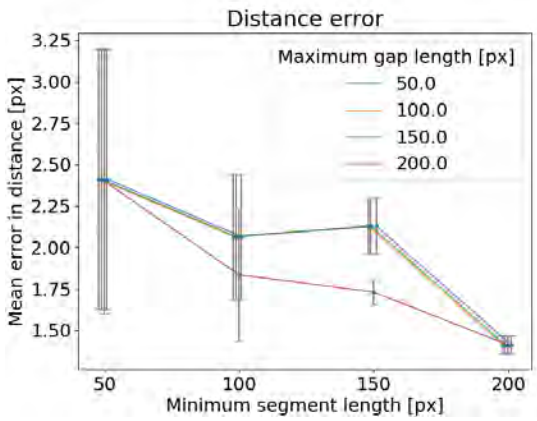
The Stroe dataset contained 762 photos with a resolution of 11608 x 8708 pixels. The quality of the lines depends on the chosen parameters because the parameters in the line-detection post-processing script are specified in pixels. Therefore a test run was performed on four images of the dataset with different parameter values.

The first requirement is to choose a pre-merging minimal length value which is higher than the size of ballast stones in the photo. The post-merging length filter is found to have little to no use on the filtering of lines. With the false positives filtered out by the pre-merging parameter, the additional filter only removed true positives.

The maximum gap parameter is less trivial. It is a trade-off between creating longer lines and the chance of merging lines which are not connected in the real world which is shown in figure 4.4. Longer line segments are both easier and faster to match and reconstruct. Lego results show gap errors are also less prone when a higher minimal segment length is chosen as errors mainly arise from shorter false positives which happen to line up. Therefore in the Stroe data set, only the minimal line length was tested.



(a) Angle error of line segments. Mean angle difference in degrees between the ground-truth lines and line segments after post-processing using changing values for minimal segment length and maximal gap length between collinear segments.



(b) Distance error of line segments. Mean distance of endpoints of found line segment to ground-truth line, normalized over segment length, after post-processing using changing values for minimal segment length and maximal gap length between collinear segments.

Figure 4.2: Quality of detected line segments in both angle error and distance error as a function of post-processing parameters.

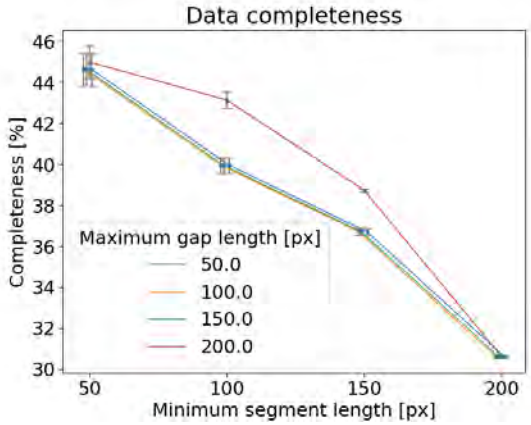


Figure 4.3: Data completeness of line segments. Percentage of ground-truth lines covered by found line segments after post-processing using changing values for minimum segment length and maximum gap length between collinear segments.

The results of line detection were validated with manually drawn ground-truth lines described in section 3.3.2. Figures 4.5a and 4.5b show the ground truth and found lines by the algorithm respectively. Results from the Lego data-set found most detected lines correspond with actual rails shown in the post-processing example in figure 4.1b. However, the lines required to determine the location of rails as specified in section 2.1 are not always completely covered.

Ground truth lines have been drawn on 4 selected images of the Stroe dataset. After which the validation on both accuracy and completeness has been determined for ranges of parameters. The results are shown in figures 4.6a, 4.6b and 4.7. These figures show respectively accuracy, split between angle and distance error, and line completeness as a function of minimal length.

Similar behaviour as in the Lego data set can be seen in the Stroe figures as well. The error in angle shows a decrease with increasing minimal line length, and minimizes at 0.08 degrees for a segment length of 200 pixels. The distance error was minimal for a segment length of 200 pixels as well, but was with 1.5 pixels slightly higher compared to the Lego data set.

Data completeness was found to be lower with 38.4 % for the most optimal parameter values and dropped more steep for increasing values as well. Just as the Lego data-set, the drop can be explained

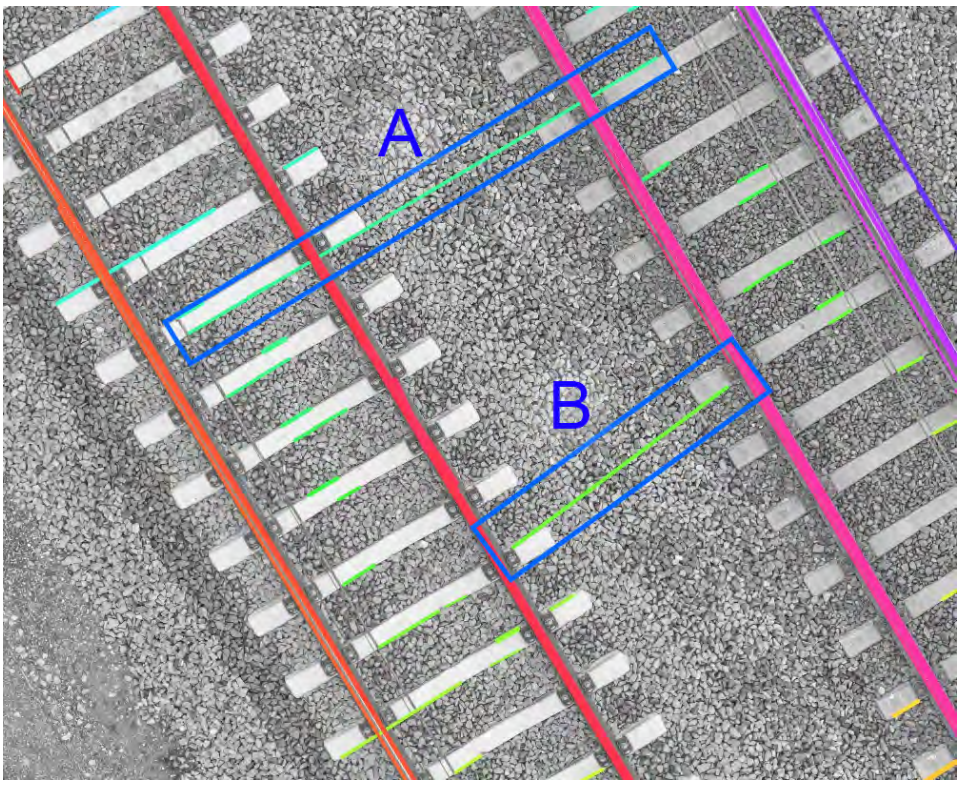


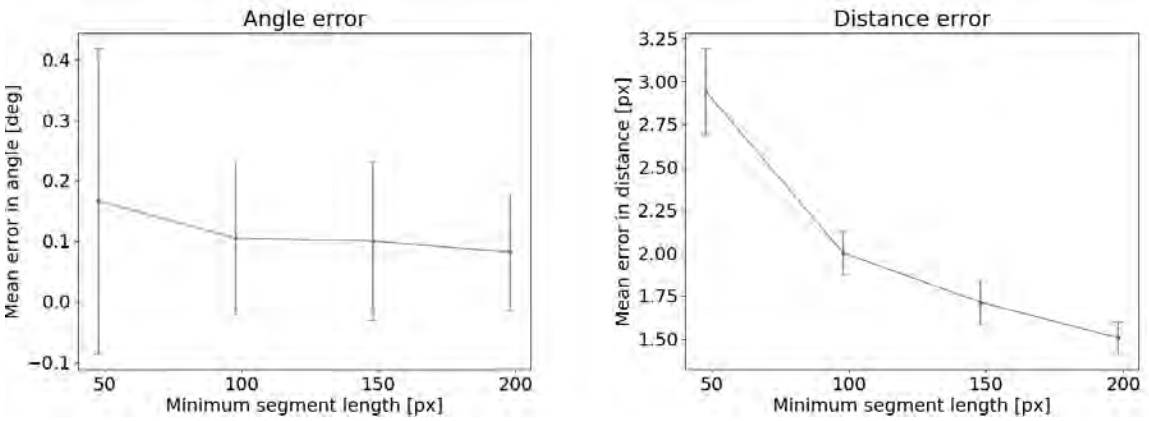
Figure 4.4: Example of merged lines with no real world connection. A high maximum merge gap setting causes collinear lines to be merged while not having a real world connection. In this case line segments between separate sleepers are merged shown in the blue borders A and B.

(a) Example photo from the Stroe dataset with in green the manually drawn ground truth lines, each rail has 4 lines which represent the rail dimensions.

(b) Example photo from the Stroe dataset with in green the manually ground truth lines and in blue the detected line segments. As there are some visible green lines, not all lines are completely covered by the line detection algorithm.

Figure 4.5: Line detection results from an example photo of the Stroe dataset with manually drawn ground-truth lines (a) and automatically detected line segments (b).

by stricter filter settings filtering out segments which should be kept in. The lower initial completeness compared to the Lego data-set can probably be explained by edges of the rail not having enough contrast to the surrounding rail parts. Especially rail top edges sometimes have almost equal colour to the rail bottom below and were therefore not picked up by the line detection algorithm.



(a) Angle error of line segments. Mean angle difference in degrees between the ground-truth lines and line segments after post-processing using changing values for minimum segment length between collinear segments.

(b) Distance error of line segments. Mean distance of endpoints of found line segment to ground-truth line, normalized over segment length, after post-processing using changing values for minimum segment length between collinear segments.

Figure 4.6: Quality of detected line segments expressed in both angle error and distance error as a function of post-processing parameters.

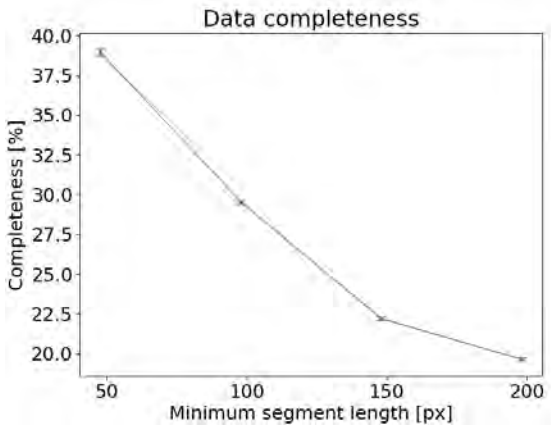


Figure 4.7: Data completeness of line segments. Percentage of ground-truth lines covered by found line segments after post-processing using changing values for minimum segment length between collinear segments.

4.2.3. Conclusion

This section showed the results of line detection in the original images, which aimed to detect line segments corresponding to rails. Results using the LSD algorithm showed small differences between the Lego set and the actual dataset. In the Lego set data-completeness decreased from 45 to 31 % for increasing minimal segment length. Orientation and position errors decreased from 0.4 to 0.1 degrees and 2.4 to 1.4 pixels respectively for increasing segment length.

The Stroe dataset showed data completeness with a drop from 38 to 20 % for increasing minimal segment length. Orientation and position errors are almost equal with a decrease from 0.2 to 0.1 degrees and 2.9 to 1.5 pixels respectively. Limiting false positives, some filtering is required. Therefore a trade-off has to be made between line accuracy and data completeness. For both datasets, this is

chosen to be a minimal line segment length of 100 pixels.

4.3. Line matching and Structure from Motion

With line segments detected in the images, matches need to be made between different images in order to be able to construct a 3D reconstruction. This part will answer the following sub-questions:

- What features can be used to match railway parts such as rails and sleepers between images?
- Can constraints found in specifications of rail superstructures be used to improve the SfM model?

4.3.1. Lego dataset

With a threshold of 1000 SIFT key-points in common, the algorithm found 123 pairs to consider out of the maximum possible 496. Figure 4.8 shows matching results between two images. The colours of the matching lines correspond to their correlation score, where green equals a perfect correlation and red negative correlation. The figures show the lower scoring lines (more red) generally are false matches.

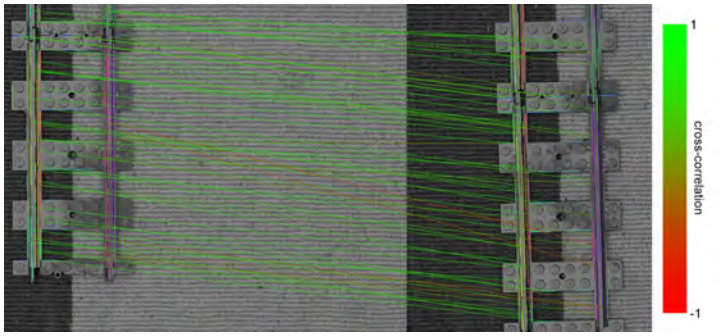


Figure 4.8: Line matching on Lego data. Matches are chosen based on a cross correlation score. The lines link the best matches between the left and right image. A higher correlation score is shown as a more green line, lower scores are more red.

Triple view matching takes the 10 most related candidates found by images pair matching, and tries to find the best fitting option for three images by checking for geometric matches using the fundamental matrix. After this, the resulting matches are reconstructed in a 3D model of the Lego data set, which are shown in figure 4.9. The mismatches result in a messy cloud, which obstructs any well matched lines.

As the matching so far is not accurate enough, reconstruction based on these matches is not accurate either. However the method has been tested on manually drawn and matched lines. The result of this are shown in figure 4.10, with a dense point cloud reconstruction shown as context. The general shape of the reconstruction is quite good, but some large variance is visible in the sleeper lines as well. As the lines are hand drawn, on a resized image, it is expected most (if not all) of this error is due to 2D line errors.

4.3.2. Stroe dataset

Compared to the Lego data, the set of Stroe contains a lot more images with each having a higher resolution as well. Line detection results showed to have similar accuracy compared with Lego images, however a slightly smaller data completeness.

An advantage of the images in the Stroe dataset is availability of real world imperfections in the railway

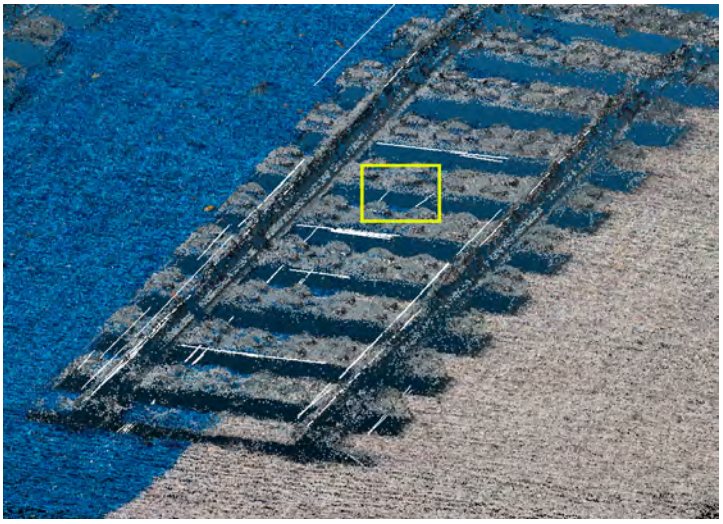


Figure 4.9: Automatic reconstruction result. The general shape of the rail, shown as a photogrammetric point-cloud, is visibly reconstructed by the white lines. Also some lines are reconstructed beneath the actual scene shown in the yellow border, this is because of mismatches.

construction. Unlike plastic Legos coming from a perfect mold, rails rust unevenly, do not sit on top of perfectly spaced sleepers, and are surrounded by a random noise induced by ballast. Therefore the matching process is expected to find pairs of lines more accurately compared to the Lego set.

Figure 4.11 shows some matching results of the Stroe dataset. As with the Lego data-set results shown in figure 4.8 line with a high cross correlation score are shown in green, and a low score is shown in red. It shows that the differences brought by a real world example do not bring the expected improvement on matching accuracy as many lines are not matched correctly, and there are lines mismatched with high cross correlation scores.

One explanation of this might be found in the change in angle between matched views. Because of the slight rotation between both images, compared pixel windows show lower correlation than expected on matching lines. Therefore no obvious match is found and a list of semi correlating matches is returned.

These mismatches also influence the results of the triple view matching, and therefore the reconstruction process. Figure 4.12a shows the result of reconstruction, together with the photogrammetry point cloud. All the white line should correspond to part of a rail, but most of the lines do not. Some of the reconstructed lines shown in white are reconstructed parallel to the rail but on the wrong location, as visible beneath the point cloud. Others do not seem to resemble any coherence with the scene and point in random directions, for example the group of vertical lines.

Because of the amount of incorrectly reconstructed lines, post-processing is applied to remove outliers. Section 3.2.3 introduced two methods to detect outliers and removed lines when a threshold is met. Figure 4.12b shows reconstruction after reprojection and filtering. Although the result is less cluttered, it is more visible there are very few lines corresponding to actual rails shown in the photogrammetry point cloud.

4.3.3. Conclusion

Matching turned out to be the hardest part in reconstruction of both data sets. Because of slight defects in the rails, and a random pattern in the ballast, it was expected line matching using pixel values would yield adequate results. This was also found in smaller test images. However, for the large data it does not work, probably due to camera rotation. Therefore full size reconstruction using this method so far did not yield the expected results. In order to have geo-referenced results, separate parts of the

Figure 4.10: Reconstruction of manually matched Lego data. The white reconstructed lines resemble the rail from the photogrammetric point-cloud, two lines per rail show the top and two lines the bottom of the rail.

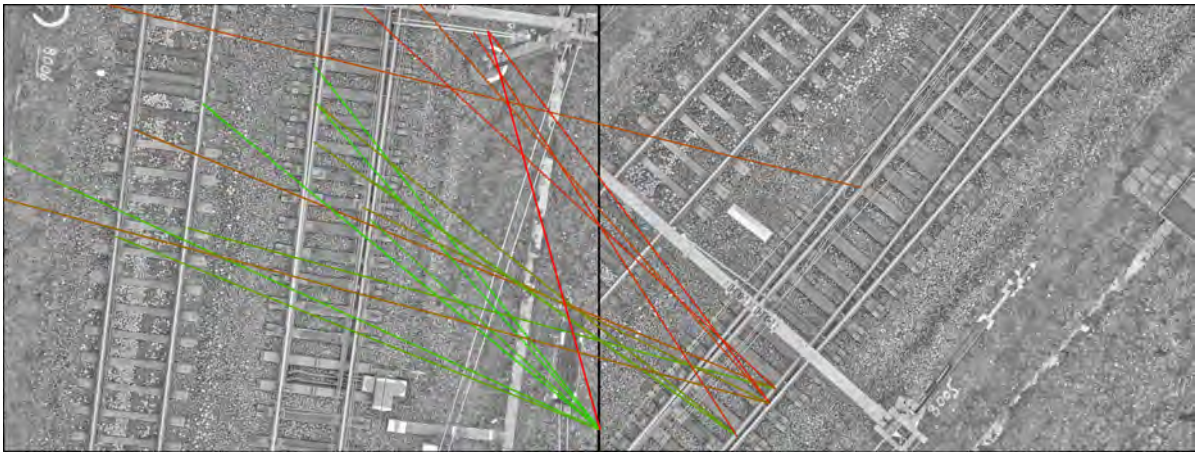


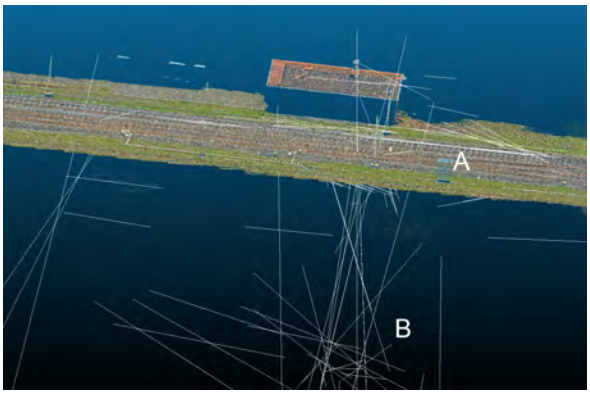
Figure 4.11: Stroe matching example. Matches are chosen based on a cross correlation score, the lines link the best matches between the left and right image. A higher correlation score is shown as a more green line, lower scores are more red.

datasets are inspected. Reconstruction did not benefit of rail superstructure specifications due to failed reconstruction as mentioned before. In order to aid reconstruction with known specifications, each line segment has to be reconstructed accurate enough to estimate its location in the superstructure. Because of failed reconstruction, no specification based constraints have been applied.

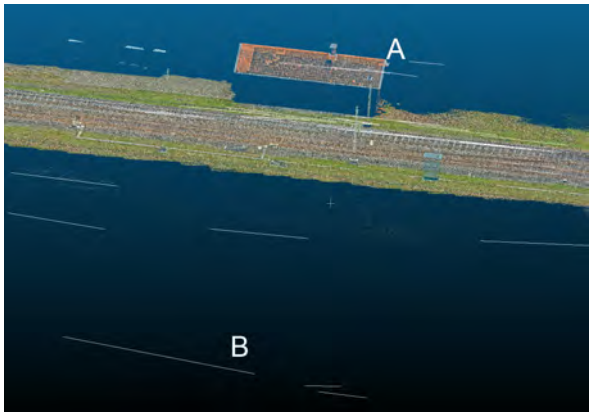
4.4. Geo-referencing

This part will answer the following sub-questions:

- Which accuracy and precision in the RD/NAP coordinate-system can be found on the GCPs comparing traditional SfM and the proposed method?
- Which accuracy, precision and data completeness in the RD/NAP coordinatesystem can be found on the rails comparing 3D rail detection in the traditional SfM point cloud and the proposed method?



(a) Reconstruction of Stroe data-set without post-processing. Next to the building some lines are shown representing rails (A), but most of the lines are reconstructed in a seemingly random direction and location (B).



(b) Reconstruction of Stroe data-set with post-processing. Filtering removed the vertical outliers, resulting in a more clear reconstruction. However quite some lines are floating above (A) or below (B) the scene as visible in the photogrammetric point-cloud.

Figure 4.12: Reconstruction results of the Stroe data-set before (a) and after (b) post-processing.

4.4.1. Lego dataset

Because the Lego dataset does not have a geo-referenced real world model and any control points either, it is not possible to answer the sub-questions concerning the RD/NAP coordinate system. Therefore the reconstruction is compared with a 3D model of a Lego rail piece. Both models are registered by point-picking with CloudCompare, and the matched pair is shown in figure 4.13.

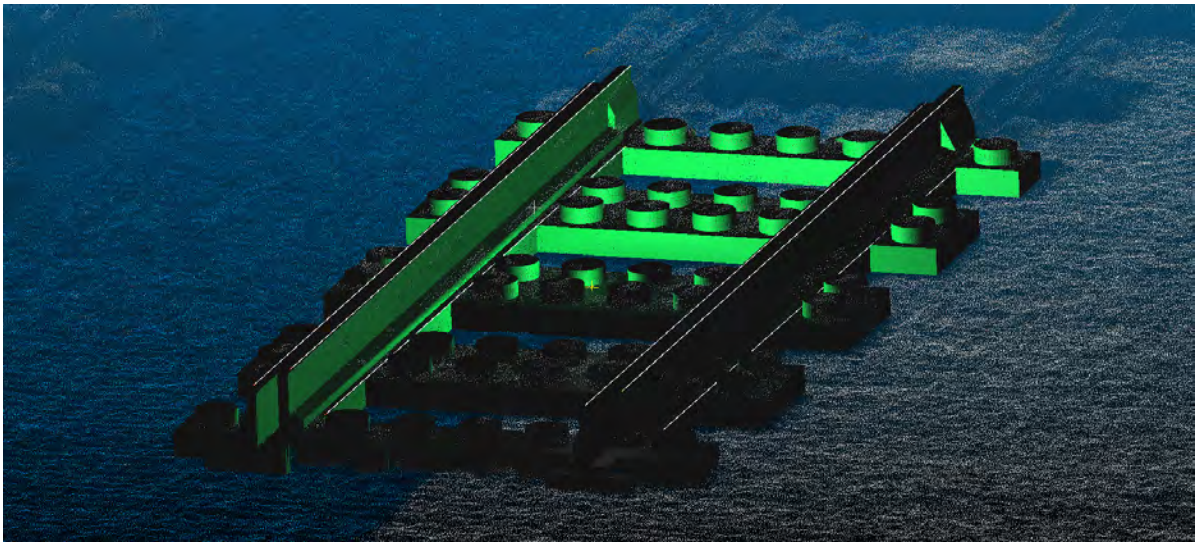


Figure 4.13: Reconstruction of Lego data together with ground-truth model. As a background the photogrammetric point-cloud is visible. In green the 3D model of the Lego rail which is aligned with the point-cloud. The white lines following manual reconstruction follow the model precisely.

For each line as shown in figure 4.13 the orthogonal distance to its ground truth line is determined. Over 8 lines, this results in a mean reconstruction error of just 0.56 mm, with a standard deviation of 0.22 mm. To give it a sense of scale: the Lego piece is 125 mm long, and the top of the rail is 3.2 mm wide.

4.4.2. Stroe dataset

The Stroe dataset does have ground control points, and is therefore geo-referenced according to PPK GPS measurements on these points. Figure 4.14 shows the available ground control points on the photogrammetry point cloud. Using Helmert transformation, the rotation, translation and scaling parameters can be determined as explained in section 2.4.

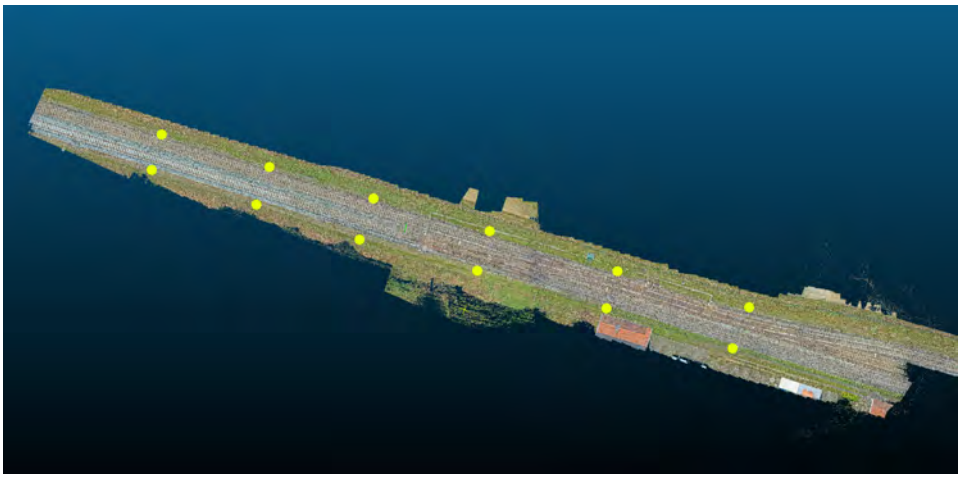


Figure 4.14: The photogrammetric point-cloud of the Stroe area. In yellow the locations of accurately measured Ground Control Points

Given the results of figure 4.12b, accuracy and precision of reconstructed lines is difficult to measure. Comparing all lines to their actual location would result in an average error in the order of meters. The best result is shown in figure 4.15, over a small part there are enough images with similar viewing angles, where photos are taken without camera rotation. The algorithm is able to reconstruct some of the lines using this set of images. Unfortunately, this part of the Stroe area is not examined during previous research and therefore there is no laser scan or traditional photogrammetric point cloud available.

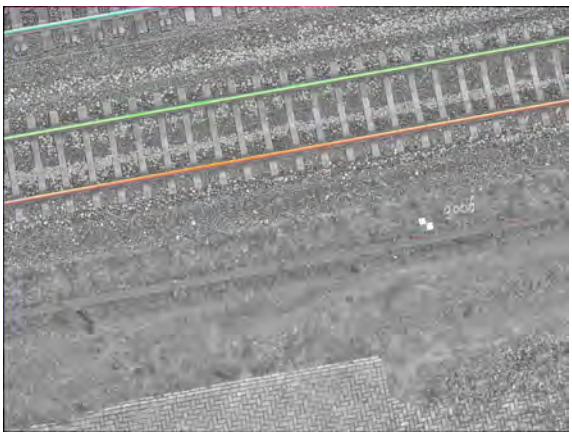
Figure 4.15: Best results for reconstruction found in the Stroe data-set. Reconstructed lines shown in yellow. Although most lines are in the general direction and location of actual rails, some lines have a vertical component which does not exist in reality (highlighted by the blue border).

4.4.3. Conclusion

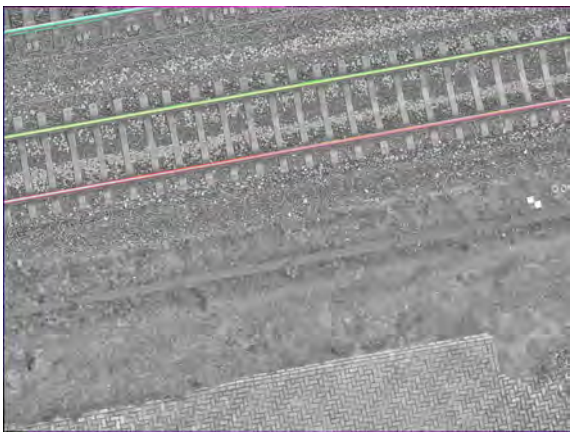
Due to matching problems, a geo-referencing result cannot be given. It is impossible to link the incoherent reconstructed lines to their ground-truth twin. Therefore it is also not possible at this moment to answer the sub-questions regarding accuracy, precision and data completeness.

4.5. Case study on Stroe data-set

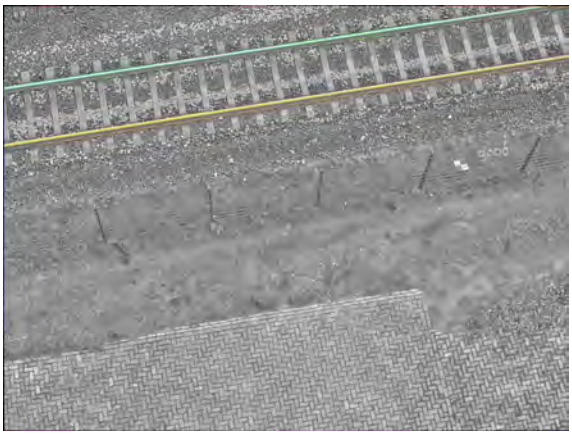
To get more insight in why automatic reconstruction fails, a case study has been performed following a line segment in the Stroe data-set. The case study is split in two parts, one to show a working example and the other one with a failing reconstruction. This section uses the same structure as the whole chapter, excluding geo-referencing. Line detection in original images introduces the used set of images, and the line segments found on these images. A single line is than chosen to be followed and in Line matching and reconstruction both a working and failing example are discussed.



(a) Image 1 with 33 detected line segments



(b) Image 2 with 35 detected line segments



(c) Image 3 with 28 detected line segments

(d) Image 4 with 87 detected line segments

Figure 4.16: Photos of Stroe data-set used in case study with the detected line segments drawn. Image 1 is the base image. Image 4 has a notably higher count of line segments which are probably explained by the presence of an extra rail-line and part of a switch visible in the lower bottom.

4.5.1. Line detection in original images

Four images have been selected which all cover the same area. All images are shown in figures 4.16a, 4.16b, 4.16c and 4.16d. For all images, the line detection algorithm is run with a minimum line

segment length of 100 px and a maximum merge gap length of 150 px. For this case study, one line segment $\mathbf{l}$ is chosen to be followed which is shown in image 4.17. The image containing this segment is considered the base image. All other images have a segment $\mathbf{l}'$ corresponding to $\mathbf{l}$ as well as other detected segments:

Image 1 33 line segments.

Image 2 35 line segments.

Image 3 28 line segments.

Image 4 87 line segments.

The large difference in the amount of detected segments between image 4 and the other images can be explained by the additional rail present in this image. The rails in this image are expected to double the amount of found segments. In addition, due to the presence of additional rails because of a switch, this number is slightly increased as the switch introduces a third rail.

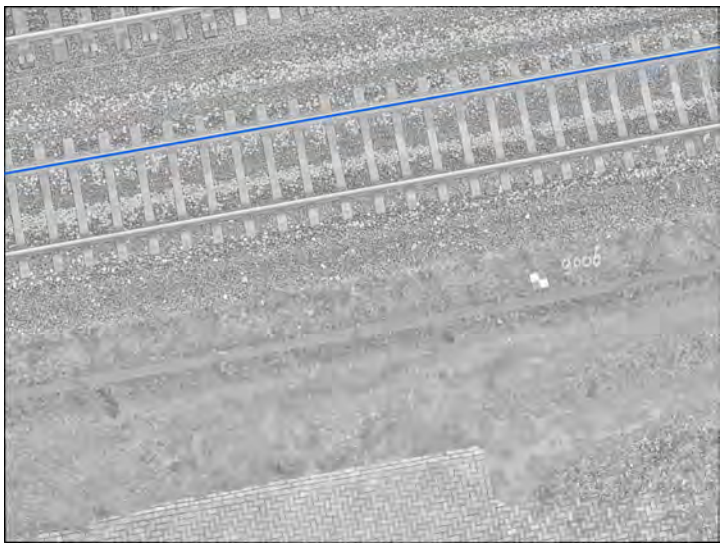


Figure 4.17: Line segment $\mathbf{l}$ in image 1 shown in blue. This line segment has been chosen to be the subject for this case study.

4.5.2. Line matching and reconstruction

With all line segments found, this section is focussed on matching $\mathbf{l}$ in image 1 with $\mathbf{barl}$ in image 2, 3 and 4 as well as matching $\mathbf{l}'$ to $\mathbf{l}'$ between images 2, 3 and 4. After matching between pairs, triplets are formed to check for the best matching candidates which are then reconstructed in 3D.

Pair matching Pair matching is performed on all pairs of images. For pairs containing image 1 this is only done for line segment $\mathbf{l}$, but for other pairs all segments are considered because of necessary triplet matching candidates as explained in section 3.2. This means for images 2, 3, 4 there are a maximum of 10 possible candidate matches to segment $\mathbf{l}$ with a correlation score based on a pixel window comparison. This yields the following results:

Image 2 3 candidates with a maximum correlation score of 0.98 and an average score of 0.36.

Image 3 10 candidates with a maximum correlation score of 0.96 and an average score of 0.77.

Image 4 10 candidates with a maximum correlation score of 0.81 and an average score of 0.63.

The results differ for each pair. Both image 2 and 3 have candidates with a high correlation score, where the score of image 4 is slightly lower. Image 2 has a clear favourite candidate, with the other two candidates having a much lower score. Image 3 however, has both more candidates and the other candidates scores are closer to the maximum as well.

A possible reason for this is the geometric restriction used in the matching algorithm. As explained in section 3.2, epipolar lines $\mathbf{l}_e$ are calculated from both endpoints of a segment in image i and then used as an area restriction in image j by limiting possible candidates to segments between both epipolar lines. Figure 4.18 shows the epipolar lines of the endpoints of line segment $\mathbf{l}$ on image 2 in blue. It shows the so-called epipolar beam is really small, severely limiting the possible candidates.

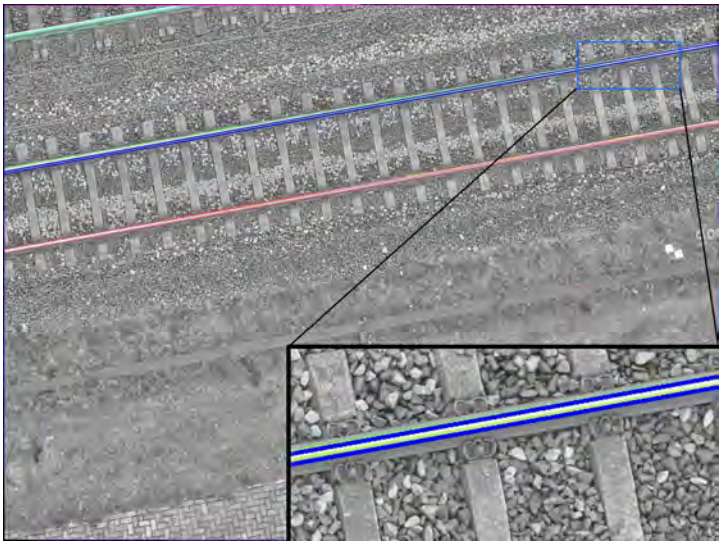


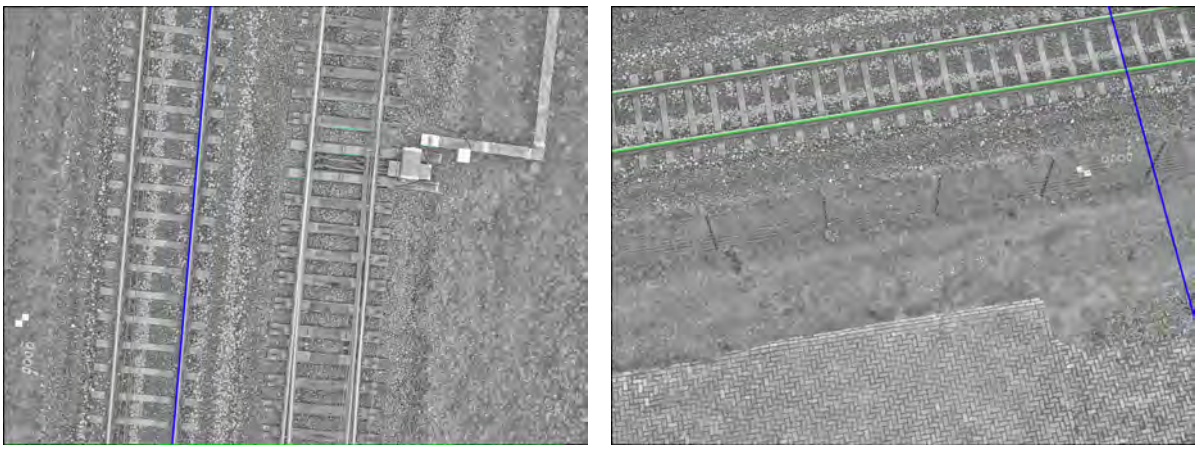
Figure 4.18: Epipolar lines from both endpoints of line segment $\mathbf{l}$ in image 1 shown as blue lines on image 2. Only line segments between both epipolar lines are considered possible matching candidates. The segment in blue is enlarged shown in the bottom right corner, showing the narrow space between both epipolar lines thus limiting the possible candidates.

The shape of the epipolar beam, explained in section 3.2, depends on both the segment length and the relative motion between both images. When the motion between the first and second image is primarily in the direction parallel to the line segment, as is the case in figure 4.18, the resulting epipolar beam is very narrow and almost parallel with the line segment as well. On the other hand, motion perpendicular to the line segment results in a beam-width equal to the length of the line segment itself. With line segment $\mathbf{l}$ stretching from the left border to the right border of image 1, this means almost all line segments in a perpendicular translated image are considered possible candidates. This is the case in image 3 and 4.

The lower maximum score of candidate matching segments in image 4 is possibly caused by the camera rotation between image 1 and 4. The 15x15 pixel windows considered for calculating the correlation score are taken oriented to the x and y axis of an image and do not consider the angle of the camera with respect to the line segment. This means a rotated image will not have a high correlation score for the matching line segment which will probably cause mismatches.

The maximum score candidates of both image 2 and 3 are the line segments $\mathbf{l}$ corresponding to $\mathbf{l}$, with a correlation score of 0.98 and 0.96 respectively. For image 4 this is not the case due to rotation, the maximum score candidate correspond corresponds with a line segment representing a sleeper with a correlation score of 0.81.

Triplet matching With the pair matches completed, triplets have to be formed in order to be able to reconstruct line segment $\mathbf{l}$. Image 1 is again taken as the base image, and two separate triplets are considered; image 1, 2 and 3 and image 1, 2 and 4. Triplet matching considers all candidate matches from the second image of a triplet and uses the combination of each candidate and line segment $\mathbf{l}$ together with the trifocal tensor explained in section 2.3.2 to determine the projection of the matched line on the third image as shown in blue in image 4.19a. With this projected line, possible matches are



(a) Projection of line $\mathbf{l}$ on image 4 as a result of pair matching between image 1 and 2. The correct match between image 1 and 2 results in a projected line corresponding to $\mathbf{l}$ in image 4.

(b) Projection of line $\mathbf{l}$ on image 3 as a result of pair matching between image 1 and 2. Due to a mismatch between image 1 and 2, the projection in image 3 does not correspond with $\mathbf{l}$.

Figure 4.19: Successful (a) and failed (b) triplet matching results.

found by comparing line segments considering both direction and distance to the line. When multiple candidates are found, the triplet of candidates with the highest correlation score will be picked. Here the correlation score is taken to be the average correlation score of these segments in all pairs. For an incorrect match between the first and second image, the projected line will likely not correspond to any candidate in the third image as shown in figure 4.19b. In the case a segment will correspond, correlation is expected to be lower than a correct matching triplet.

The candidates are restricted to segments already considered a possible match in pair matching. Therefore the possible candidates are an overlapping subset of the candidates between the first and third image, and the second and third image. This poses a problem for the second triplet, as $\mathbf{l}$ in image 4 is not found as a possible candidate in image pair 1-4, it is not considered in triplet matching. Therefore, although the correct projection as shown in figure 4.19a, the second triplet does not yield a correct match due to an absence of suitable candidates in image 4.

Reconstruction Both triplets result in a single line segment to be reconstructed. As seen in the previous section, this is one correct match for the first triplet, and a mismatch for the second. Reconstruction will take the 3 segments per triplet and find the best fitting 3D line corresponding to these segments based on the reprojection error.

Figure 4.20 shows the resulting reconstructed lines, in green the correctly matched line from triplet 1 and in red the incorrectly matched line from triplet 2. Although both triplets use the same segments from image 1 and 2, the mismatch in the third image causes a line to be reconstructed both in the wrong direction and location, which is therefore useless.

4.5.3. Relation between rotation and correlation

The case study results show camera rotation between images decreases pair matching correlation scores and cause mismatches. To quantify this, a set of pixel windows from multiple images have been

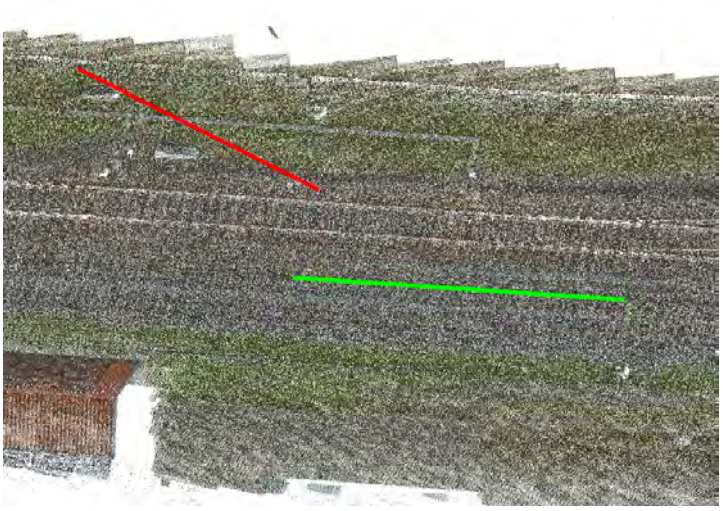


Figure 4.20: Reconstruction of line $\mathbf{l}$ for both a correct triplet match shown in green, and an incorrect triplet match shown in red. Background is the SfM result of the Stroe data-set

rotated after which the correlation between the original and rotated window is determined. Doing this over a range of angles gives an insight in the decrease of correlation over angles and provides an recommended maximum angle of rotation between images.

Figure 4.21 shows the found relation between rotation and average correlation following rotating the set of windows between -180 and 180 degrees. The maximum correlation of 1.0 is logically found for zero rotation. Between -2.5 and 2.5 degrees the correlation is higher than 0.99. Larger rotation angles show an increasing drop in correlation between the windows. For an absolute rotation of 8.1 degrees a cross-correlation of 0.90 is found and the correlation drops below 0.80 for a rotation of over 13.2 degrees.

As mismatches occurred between image 1 and 4 which had a correlation score of 0.81, correlation between correctly matched line segments would ideally be higher than 0.90. Therefore a maximum rotation angle between images of 8.1 degrees would be advised.

4.5.4. Conclusion

This case study described step by step the reconstruction process of a line segment in the Stroe dataset. It showed the importance of matching for reconstruction accuracy and some of the flaws in the used matching methods. The main issue found in matching is camera rotation between images. In two images with little rotation the correct matching line segment was matched with a correlation score of 0.98 and 0.96. For a large rotation a mismatch occured where the line segments were matched with a correlation score of 0.81. When matching is successful, the reconstruction method shows an accurate line is reconstructed in 3D. Rotating several test segments provided an estimation for the relation between camera rotation and correlation. It found an absolute rotation of 8.1 degrees decreases the correlation to 0.90 for perfectly matching pixel windows.

4.6. Recommendations on solving matching problems

The conclusion that matching problems caused by camera rotations prevented adequate reconstruction indicates following research should focus on solving that problem. Therefore this section provides two recommendations on methods to improve matching performance. The first is image alignment in section 4.6.1, which rotates images according to an estimated angle of the railway tracks. The second

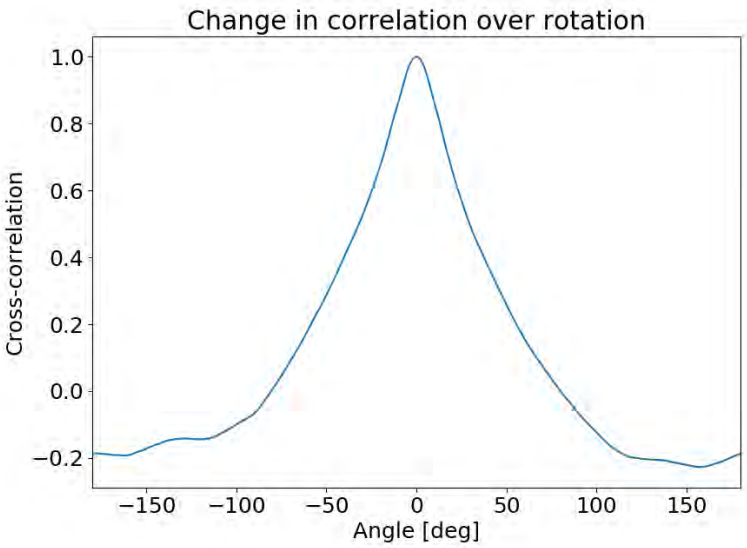


Figure 4.21: Average correlation found between a pixel window and its copy rotated over an angle between -180 and 180 degrees.

is arrangement matching in section 4.6.2. This method uses pairs of line segments for more robust matching as well as an line descriptor which prevents rotation issues.

4.6.1. Image alignment

The first option is to rotate images in order to align the railway tracks. When the tracks are aligned, the pixel windows used for matching should be aligned as well and therefore correlation between matching line segments improves. An expected downside of this method is the loss in image detail due to interpolation required for image rotation.

As the image is rotated, the camera parameters need to be rotated accordingly to correct for this. Without correcting the rotation matrix, the geometric relations between cameras as used in matching would not be aligned and therefore finding matching segments would become impossible. The correction is done by applying a rotation to the intrinsic rotation matrix R_c. The camera is looking in the negative-z direction, a rotation of the sensor plane is therefore a rotation around the z-axis of the camera. For a given rotation θ the corrected intrinsic rotation matrix R'_c follows from:

$$R'_c = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} R_c \tag{4.1}$$

The images as well as the lines on the image are rotated by multiplying the upper left 2x2 part of the equation with the pixel coordinates.

To estimate the angle θ different options exist. One could use a selection of matched SIFT key-points following from the Structure-from-Motion algorithm to find a rotation relative to a fitted plane through those points. Another option is to first detect line segments in the images and use the assumption that the majority of those line segments do belong to railway tracks as the results show in section 4.2. Figure 4.22 shows a boxplot of the distribution of angles found in the images used for the Stroe case study. The red line shows the median value of the line segments in each image. This median is used as an estimate for the angle of the railway tracks. The box around the median show the angles corresponding to 50% of the line segments in each image.

The angle estimate is around -0.15 rad or -10 degrees for the first three images and -1.5 rad or -90 degrees for the fourth image which is as expected as image four is about 90 degrees to the other images. This also raises a problem with this method as image four is rotated over 1.5 rad in a counter-clockwise direction instead of just under 1.5 rad in a clockwise direction. Rotating the image according to the found angle flips the image upside down which does not help the matching algorithm. Pixel window matching would yield a negative correlation score for a flipped image where the algorithm looks for high positive correlation.

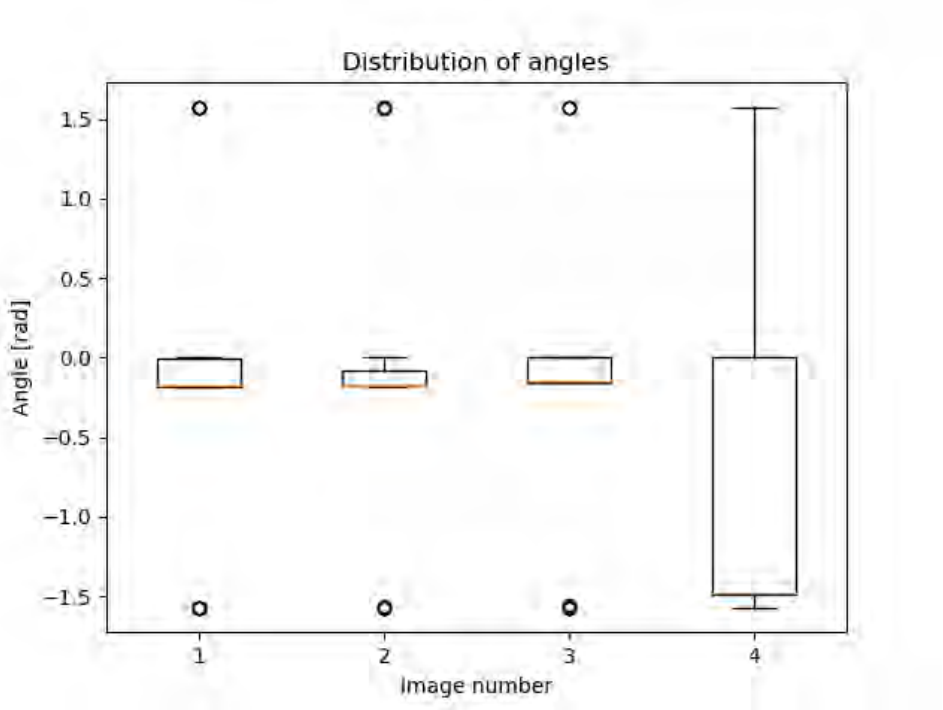


Figure 4.22: Boxplot distributions of the angles of detected line segments as found in the four images used for the Stroe case study. Red line shows median value.

With aligned images and camera parameters rotated accordingly, matching is performed using the same method from section 3.2. Figure 4.23 shows a comparison between a rotated and original pixel window of the same part. The middle square shows the difference between both windows where no difference is shown in black. A completely black square would indicate a correlation score of 1, indicating a match. Some loss of correlation due to rotation is expected using this method. Rotating an image causes points from a regular pixel grid to end up between multiple grid points. In order to be able to save the image all these points have to be converted to pixels within the grid again, which is done by interpolating these points to corresponding pixels. This interpolation causes some loss of detail, which decreases the correlation score.

A test on the 4 case study images from section 4.5 show a slight increase in correlation scores for lines which were correctly matched without this method as well. Image 1 and 2 have a relative rotation of just 0.1 degrees, correcting for this resulted in a correlation improvement from 0.981 to 0.983 or a 0.1% increase for the highest scoring segment. Image 1 and 3 have a relative rotation of 1.5 degrees, and the correction resulted in an increase from 0.958 to 0.988 or 3.1%. The determined relative rotation between image 1 and 4 is 74.8 degrees, as visible in figure 4.22. As the image was found to be flipped this was however corrected to 105.2 degrees. In the case study the maximum correlation score found was 0.81 although for an incorrect match where two line were matched.

The case study images show a promising increase in matching performance after aligning the images, although flipped images possibly causes issues. Using a different method to find the angle of the railway tracks might solve this, as for example a set of matching SIFT key-points is insensitive for this. There is some loss of correlation to be expected due to interpolation, but the increasing correlation for little

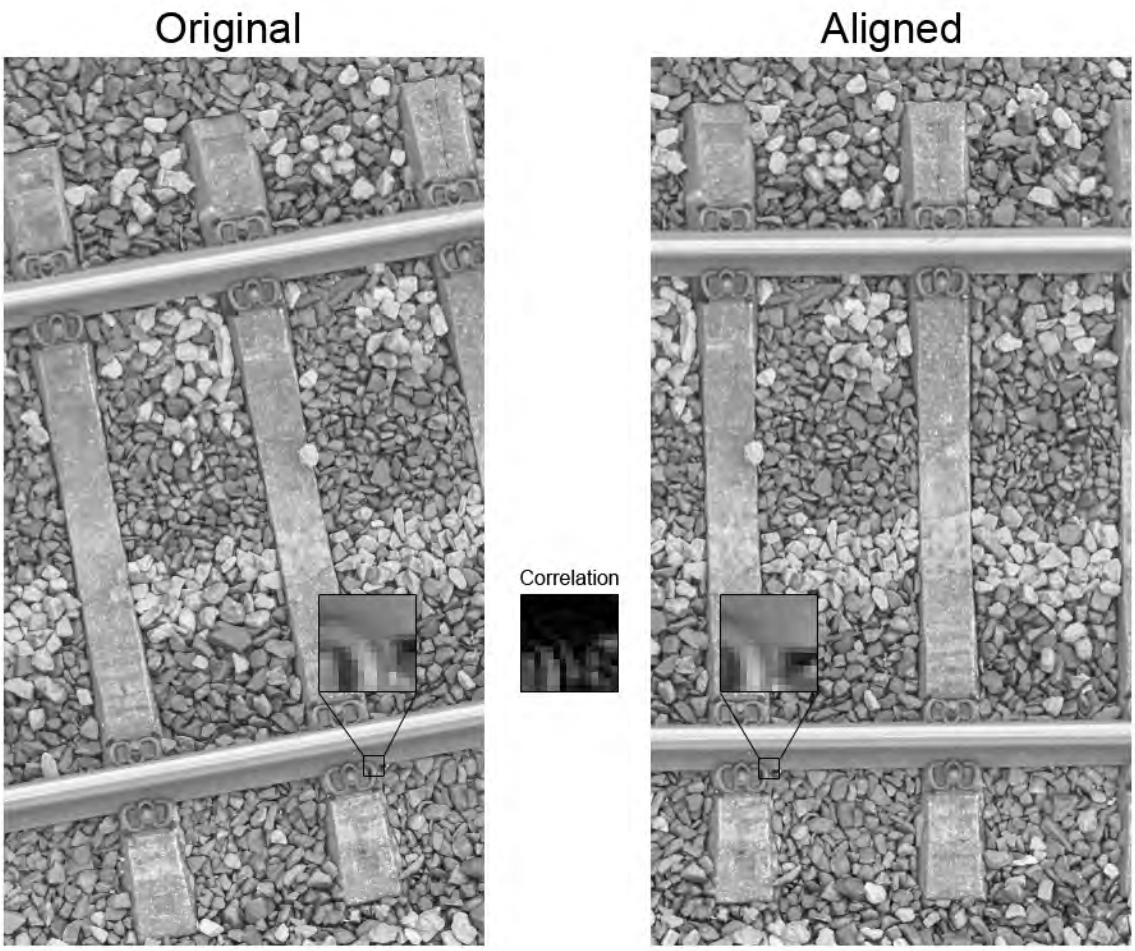


Figure 4.23: Difference between pixel windows of an original and aligned image. The middle square shows the correlation between both squares on the side, dark areas are have a higher correlation because they are more alike.

rotation image pairs are positive. Additional research is required to find it's effect on full scale matching performance.

4.6.2. Arrangement matching

Arrangement matching utilizes a combination of non-parallel line segments to form a group where features from all segments are used to get a unique descriptor. Preferably close to orthogonal segments are combined in a set as they form intersections which are used as an anchor. One example using this method is Wang et al. [26]. Their method forms pairs of segments where possibilities are filtered based on the angle and the distance between end points of both lines. This prevents far away and parallel line segments from becoming a pair. The intersection point of pair segments is used as an extra geometric constraint by projecting it on the matching image. Figure 4.24 shows the pair to be matched on the left, with the intersection marked in red. The epipolar line of this intersection is shown on the right on top of a candidate pair of segments. The intersection of this candidate pair should match the intersection of the original pair. Therefore only candidate pairs with an intersection close to the epipolar line are considered a possible match.

To combat rotation errors, Wang et al. proposes a 5 band Line Band Descriptor (LBD). After determining overlapping segments between the original and candidate pair, a rectangular region parallel to each segment is transformed by affine transformation to remove camera rotation. The LBD is then calculated

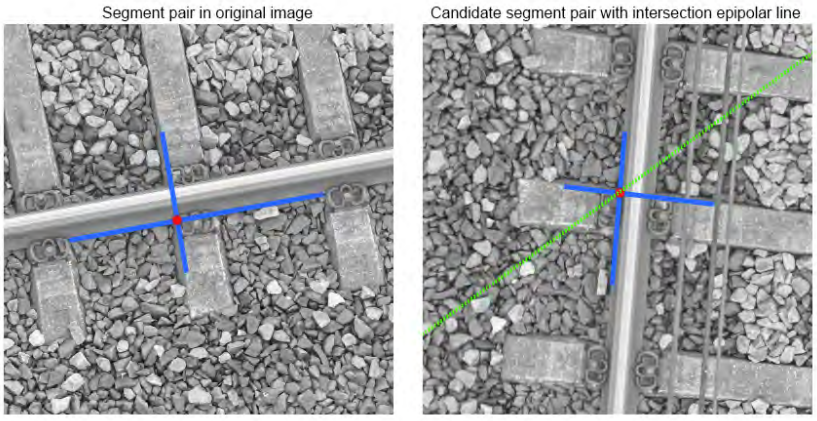


Figure 4.24: Segment pair matching. Original pair on the left with the intersection marked. Dotted line in the candidate image is the epipolar line of the intersection. The intersection of the candidate segment pair shown should be on this epipolar line.

by dividing the region in 5 equally sized bands parallel to the segment, and estimating the pixel gradient over the band. The 5 gradients of these bands and the mean and standard deviation of the intensity values within these bands form the Line Band Descriptor which is used to determine if a candidate following the geometric constraints is a possible match.

The LBD should mitigate the rotation errors from this thesis' method, while the intersection relation between pairs of segments should help prevent flipping mistakes as found in the image alignment method. A possible downside of this method is the downsampling of pixel details to three features in 5 bands. Further research will have to show whether the descriptor is able to be distinguishable for repetitive environments as railway tracks.

4.6.3. Conclusion

Two methods have been provided to improve results of this research, which suffers from line segment mismatching due to camera rotation between images. One method tries to solve the rotation itself, by aligning all images before performing the matching process. Tests show increased performance on the case study images from section 4.5 however there is some ambiguity between rotating an image counter-clockwise or clockwise which causes flipped images.

The second method uses a more strict geometric constraint by matching intersecting line segment pairs. Using epipolar lines as constraints for both the line segments and the intersection, possible mismatches are limited. It also solves the rotation issue by using a line based descriptor, describing a line segment in 15 parameters spread over 5 bands. This means some detail found in pixel window correlation is lost, which leaves the question if 15 parameters will be a unique enough descriptor to match railway track line segments.

The more constrained method of matching pairs of line segments, arrangement matching, prefers further research as it both solves the camera rotation issues as well as introduces more strict geometric constraints which limits possible mismatches. When the line descriptor proves to be insufficient, a more detailed descriptor can be implemented by combining both methods.

5

Conclusion and recommendations

This thesis has been done to find an improvement for rail infrastructure monitoring where rails are reconstructed in a geo-referenced point cloud in order to measure their accurate real-world location. Photogrammetry is a method to create such a point cloud, but it has difficulties reconstructing rails. On photos obtained by UAV carried cameras rails appear as long small beams with a homogeneous rusty texture and a reflective rolling band on top. In this scale they do not have enough features for automatic feature extraction methods used for photogrammetry. Their line like nature brought the idea to use line based features for reconstruction instead of points used in traditional photogrammetry methods. This research brought a method starting with extracting line features from images, matchings those features and reconstructing the lines in 3D. The next sections provide the conclusions of that research divided over Line detection in original images, Line matching and reconstruction and Geo-referencing and validation. Each sections also includes recommendations on further research. The chapter ends with an answer on the main research question.

5.1. Line detection in original images

In order to find the rails in photos, line detection was applied as rail to the human eye is a line-like feature instead of a point-like feature. The goal of this part was to test whether an automated algorithm would think the same. To test this, an algorithm has been found in the form of the Line Segment Detector by von Gioi et al. Raw results showed many found segments found were small and did not correspond to either rails or sleepers. The algorithm found 12844 segments on an image of the Lego dataset where 2856 corresponded to rails.

To mitigate this, post-processing in the form of segment length filtering and segment merging was applied. On the same image this reduced the amount of segments to 106 and 61 corresponding to rails. A range of parameters has been tested to find optimal performance. This resulted in a data-completeness for the Lego data-set between 31% and 45%, and a slightly lower data-completeness for the Stroe data-set between 20% and 38%. Line quality was similar for both data-sets, measured in both accuracy in angle and accuracy in distance. For the Lego data-set the angle error lies between 0.1 and 0.4 degrees, with an distance error between 1.4 and 2.4 pixels. For the Stroe data-set, the angle error was found to be between 0.1 and 0.2 degrees, with an distance error between 1.5 and 2.9 pixels.

The error in angle and distance is considered a good result. Compared with the ground sampling distance it translates to 1.2 millimetre for the Lego dataset and 4.5 millimetre for the Stroe dataset. The data-completeness needs improvement.The ProRail requirement of a rail position measurement

every 2 meters as described in section 2.1 leaves some room for missing parts of the railway tracks but this requires at least measurements on the same cross section of the track. The results show the data-completeness is not capable to do that.

To improve data-completeness new line detection algorithms can be researched. The Line Segment Detector is well known and well praised, but newer algorithms are available. Another option is to extend the current workflow. The results showed most line segments after detection correspond with rails. Gaps between segments can be filled by methods more smart than the maximal gap distance used in this research. Here one could for example use the region growing idea already implemented in LSD.

5.2. Line matching and reconstruction

Line matching is required to find multiple views with the same line segment in order to estimate its 3D position. Pixel based matching was used between detected segments in image pairs. Aided by a geometric constraint it takes a window of pixels around the line segment and a matching window from the other image. It then calculates the cross-correlation between both windows and selects the 10 candidates with the highest average score.

Results from both data-sets showed many mismatches limited usability of matched segments for reconstruction. Therefore case-study was conducted which showed rotation was a cause of mismatches due to low correlation scores. It found correct matches with a score of 0.96 and 0.98. For a rotated image an incorrect match was found with a correlation score of 0.81. Rotating windows over a range of angles showed correlation is constant for lower rotation angles. A correlation score higher than 0.99 was found for an absolute rotation smaller than 2.5 degrees. Using this method it is advised to keep camera rotation lower than 8.1 degrees as the correlation than drops to 0.90.

The matching performance translates in an incoherent reconstruction and therefore unusable result. The manually matched Lego dataset and the case-study showed reconstruction to be promising for correctly matched line segments, although this is not quantified.

Two recommendations on matching improvement are already provided in section 4.6. One method aligns images before matching, correcting for camera rotation. The method showed an increasing correlation score for test images, but it suffered from ambiguity between clockwise and counter-clockwise rotation. Wang et al. [26] provides a second method which matched pairs of intersecting segments instead of separate segments. The intersection provides an additional geometric constraint for matching. Camera rotation issues are solved by a Line Band Descriptor which is aligned.

5.3. Geo-referencing and validation

Geo-referencing links the reconstruction with real-world coordinates and is therefore of vital importance to be able to get accurate measurements usable for railway monitoring. Due to matching problems, a geo-referencing result cannot be given. It was impossible to link the incoherent reconstructed lines to their ground-truth twin. Therefore it was also not possible to answer the sub-questions regarding accuracy, precision and data completeness of the geo-referenced results.

The results of the manually matched Lego dataset show precise reconstruction is possible for correctly matched line segments. It found a mean distance error of 0.56 millimetre. Translating the size of a Lego railway track to real railway infrastructure this corresponds with an error of 12.2 millimetre.

5.4. Answering the research question

This research started with the question *What contribution can line recognition on the input images bring to the structure from motion based process to extract the geo-referenced 3D position of railway tracks?*. Some of sub-questions included to help answer the main question were left unanswered due to matching and reconstruction problems. Therefore the contribution of the used method is limited. Results showed the matching method was susceptible to camera rotation between images. An angle of over 8.1 degrees causes a decrease in correlation to 0.90, even for perfect matching windows. This causes mismatches which led to failing reconstruction.

However manually matched segments in the Lego dataset showed the reconstruction algorithm is promising. The precise line segments were able to reconstruct parts of the railway track which were not visible in the traditional photogrammetry result. It is therefore concluded the contribution of line recognition deserves more research, for which recommendations are given to improve matching results and line detection data-completeness.

Bibliography

[1] C. Baillard, C. Schmid, A. Zisserman, and A. Fitzgibbon. Automatic line matching and 3d reconstruction of buildings from multiple views. page 13, 2011.

[2] A. Bartoli and P. Sturm. Structure-from-motion using lines: Representation, triangulation and bundle adjustment. *Computer Vision and Image Understanding*, 100:416–441, 2005. URL https://hal.archives-ouvertes.fr/hal-00092589.

[3] J. Cardoso, P. Miraldo, and H. Araujo. Plücker correction problem: Analysis and improvements in efficiency. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, page 2795–2800. IEEE, Dec 2016. ISBN 978-1-5090-4847-2. doi: 10.1109/ICPR.2016.7900059. URL http://ieeexplore.ieee.org/document/7900059/.

[4] J. L. Carrivick, M. W. Smith, and D. J. Quincey. *Structure from Motion in the Geosciences*. John Wiley & Sons, Incorporated, 2016. ISBN 978-1-118-89582-5. URL http://ebookcentral.proquest.com/lib/delft/detail.action?docID=4595447.

[5] C. Esveld. *Geometrisch en constructief ontwerp van wegen en spoorwegen: Deel D. Constructief ontwerp van spoorwegen*. 2005. URL https://denhaagtekijk.nl/randstadrail-Ri59N/Spoorweg%20ontwerp.pdf.

[6] W. Förstner and B. P. Wrobel. *Photogrammetric Computer Vision*, volume 11 of *Geometry and Computing*. Springer International Publishing, 2016. ISBN 978-3-319-11549-8 978-3-319-11550-4. doi: 10.1007/978-3-319-11550-4. URL http://link.springer.com/10.1007/978-3-319-11550-4.

[7] R. Grompone von Gioi, J. Jakubowicz, J. Morel, and G. Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):722–732, Apr 2010. ISSN 1939-3539. doi: 10.1109/TPAMI.2008.300.

[8] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2004. ISBN 978-0-511-18618-9. URL http://qut.eblib.com.au/patron/FullRecord.aspx?p=256634.

[9] M. Hofer. *Building with Lines: Efficient 3D Scene Abstraction for the Built Environment*. PhD thesis, Graz University of Technology, 2016.

[10] A. Hosseininaveh, R. Yazdan, A. Karami, M. Moradi, and F. Ghorbani. A LOW-COST AND PORTABLE SYSTEM FOR 3d RECONSTRUCTION OF TEXTURE-LESS OBJECTS. volume XL-1-W5, pages 327–332, 2015. doi: 10.5194/isprsarchives-XL-1-W5-327-2015.

[11] I. Nurutdinova and A. Fitzgibbon. Towards pointless structure from motion: 3d reconstruction and camera parameters from general 3d curves. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2363–2371, 2015. doi: 10.1109/ICCV.2015.272. ISSN: 2380-7504.

[12] OpenCV. Opencv: Color conversions, (accessed March 21 , 2020). URL https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html#color_convert_rgb_gray.

[13] ProRail. Jaarverslag 2018, 2018. URL https://jaarverslagprorail.nl/FbContent.ashx/pub_1000/downloads/v191212105237/ProRail_Jaarverslag_2018.pdf.

[14] ProRail. Netverklaring 2020, November 2019. URL https://www.prorail.nl/sites/default/files/netverklaring_2020_versie_1.3a.pdf.

[15] C. Ressl. *Geometry, Constraints and Computation of the Trifocal Tensor*. PhD thesis, Technische Universität Wien, 2003.

[16] T. C. F. Sassen. The influence of drone flightpath on photogrammetric model quality. Master's thesis, Delft University of Technology, 2019.

[17] G. Schindler, P. Krishnamurthy, and F. Dellaert. Line-based structure from motion for urban environments. In *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, pages 846–853. IEEE, 2006. ISBN 978-0-7695-2825-0. doi: 10.1109/3DPVT.2006.90. URL http://ieeexplore.ieee.org/document/4155810/.

[18] C. Schmid and A. Zisserman. Automatic line matching across views. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, page 666–671. IEEE Comput. Soc, 1997. ISBN 978-0-8186-7822-6. doi: 10.1109/CVPR.1997.609397. URL http://ieeexplore.ieee.org/document/609397/.

[19] R. Singh, D. P. Chapman, and K. B. Atkinson. Digital photogrammetry for automatic close range measurement of textureless and featureless objects. *The Photogrammetric Record*, 15 (89):691–702, 1997. ISSN 1477-9730. doi: 10.1111/0031-868X.00078. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/0031-868X.00078.

[20] L. E. Sjöberg. Closed-form and iterative weighted least squares solutions of helmert transformation parameters. *Journal of Geodetic Science*, 3(1):7–11, Mar 2013. ISSN 2081-9943. doi: 10.2478/jogs-2013-0002.

[21] S. A. K. Tareen and Z. Saleem. A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK. In *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pages 1–10, 2018. doi: 10.1109/ICOMET.2018.8346440. ISSN: null.

[22] P.J.G. Teunissen and O. Montenbruck, editors. *Springer Handbook of Global Navigation Satellite Systems*. Springer International Publishing, 2017. doi: 10.1007/978-3-319-42928-1. URL https://doi.org/10.1007/978-3-319-42928-1.

[23] C.C.J.M Tiberius. *Primer on Mathematical Geodesy*. Faculty of Civil Engineering and Geosciences, Delft, The Netherlands, 2015.

[24] H. van der Marel. *Reference Systems for Surveying and Mapping*. Faculty of Civil Engineering and Geosciences, Delft, The Netherlands, 2016.

[25] A. Verma and M. Bourke. A method based on structure-from-motion photogrammetry to generate sub-millimetre-resolution digital elevation models for investigating rock breakdown features. *Earth Surface Dynamics*, 7:45–66, 2019. doi: 10.5194/esurf-7-45-2019.

[26] J. X. Wang, W. X. Wang, C. Y. Wang, H. Zhu, W. Y. He, and S. Y. Liu. Line segment matching algorithm based on feature grouping and lbd descriptor. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIII-B2-2020:103–109, 2020. doi: 10.5194/isprs-archives-xliii-b2-2020-103-2020.

[27] M. J. Westoby, J. Brasington, N. F. Glasser, M. J. Hambrey, and J. M. Reynolds. 'structure-from-motion' photogrammetry: A low-cost, effective tool for geoscience applications. *Geomorphology*, 179:300–314, 2012. ISSN 0169-555X. doi: 10.1016/j.geomorph.2012.08.021. URL http://www.sciencedirect.com/science/article/pii/S0169555X12004217.