UTRECHT UNIVERSITY

Single Camera 3D Reconstruction of Railway Environments

Wouter Florijn

A thesis submitted in partial fulfillment for the
degree of Master of Science

in the

Faculty of Science
Department of Information and Computing Sciences

June 2017

Acknowledgements

First and foremost, I would like to thank my thesis supervisor Ronald Poppe for his support throughout the entire process. His feedback was indispensable for my work and helped me find solutions to problems that I encountered. Secondly, I would like to thank my supervisors at ProRail, Dirk Wouters and Jelle van Luipen. Their support and enthusiasm motivated me to work hard, even during the more difficult times. They also provided me with important insights and helped my find my way through the company. Furthermore, I would like to extend my gratitude to everyone I worked with at ProRail and RIGD-LOXIA, and in particular the innovation department at which I worked myself. Being surrounded by extremely nice, helpful and smart people made the good times great and the stressful times manageable. I also would like to thank the people at Inspectation for providing me with the data I required, and Linchao Bao for providing me with source code that was essential for my implementation. I would like to thank Aart Stuurman for sharing his knowledge about C++ (in exchange for knowledge about web development). Finally, I would like to thank all my friends and family for their support before, during and after my research.

Contents

Abbreviations

SFM	Structure From Motion
FPS	Frames Per Second
GPS	Global Positioning System
SLAM	Simultaneous Localization And Mapping
RANSAC	RANdom SAmple Consensus
MLESAC	Maximum Likelihood Estimation SAmple Consensus
NNF	Nearest Neighbour Field
PCL	Point Cloud Library
RD	Rijksdriehoek
CC	Catenary Construction

Chapter 1

Introduction

The research reported here aims to find the best way to reconstruct environments of railway tracks based on video footage recorded by cameras mounted on trains. More specifically, we investigate 3D reconstruction based on a single front-facing camera mounted at the front of a train. The research is conducted in collaboration with ProRail, the Dutch railway infrastructure manager. Combining various computer vision techniques, particularly those related to structure from motion (SFM), we aim to find a solution that generates high quality results in tractable time. However, a real-time solution is not required.

ProRail is interested in 3D reconstructions of railway environments for a number of reasons. First of all, these reconstructions could be used in simulations for new train drivers. Current simulations generally consist of simple 3D models created manually. Accurately modelling realistic railway environments is a difficult and time consuming task. Automatically generating these environments based on real-world data could therefore improve the quality of simulations and the speed at which they can be built. Another motivation is the simulation of architectural changes around the tracks. For example, the placement of new traffic signs could be tested in order to ensure that they are properly visible before they are actually placed. Additionally, measurements could be made in the virtual environment to easily estimate the amount of available space. Finally, 3D data could be combined with other data gathered around the railway tracks. By combining several types of data, detection systems could be implemented more robustly.

The resulting implementation could also be used in settings other than railway tracks. However, this is not the focus of this research.

In this chapter we will explain the context and goals of our research in detail. We will first describe our input data in Section 1.1. Next, we will describe our applications and requirements in Section 1.2. We will then formulate our research questions in Section 1.3. In Section 1.4 we will state our contributions. This chapter will be concluded with an outline of the remainder of this thesis in Section 1.5.

1.1 Data

Our research problem can be largely defined by the data that is available. Therefore, we will first describe the different sources of data and their corresponding properties. We will also describe the way we use each type of data in our research. Note that all positional data uses the Dutch *Rijksdriehoek* (RD) system as opposed to the commonly used latitude and longitude system. Conversion between both systems is straightforward. The RD system has the benefit of being a Cartesian coordinate system, allowing for easier displacement computations.

1.1.1 Reconstruction Data

First of all, we examine our source of input data. This data consists of video images gathered by measurement trains. An example video frame is shown in Figure 1.1. Measurement trains equipped with various sensors traverse The Netherlands on a regular basis. Of particular interest are the trains used for the project *Digitaal Schouwen (Digital Surveillance)*, developed by ProRail and several partners [1]. Each of these trains contains a front- and rear-facing camera and a GPS sensor used to determine position. The camera calibrations are known and the relative camera positions and orientations can be determined accurately up to a few centimeters and degrees respectively. The GPS data can be considered highly accurate for our purposes. Although it does contain noise, mostly in areas where GPS signals are weak (such as tunnels), we will exclude these cases, as our research is not aimed at improving sensory measurements. The rate at which pictures are taken varies depending on the supplier of the data. The data we

FIGURE 1.1: A frame from a Digital Surveillance video. In this case a front-facing camera was used.

use is recorded at a frequency of 1 frame per 50cm (independent of travel speed). The image dimensions are 1920 by 1080 pixels.

By examining this video data, we can make certain assumptions about the nature of the recorded video. First of all, we examine the motion of the train. The camera is attached to the train and therefore moves in the same way. Because the camera takes pictures every 50cm, as opposed to a set number of frames per second, the speed at which the train travels does not influence the resulting video data. Additionally, since a train cannot take sharp turns, the direction in which it is travelling changes only slowly.

Secondly, we look at the environment in which the video is recorded. The environment around railway tracks can be described as mostly static. Although small movements may occur in the periphery (e.g. pedestrians or cyclists on a neighboring road), we assume that the scene is static for our application. The landscape is flat, although the camera height might change slightly between frames (e.g. due to bumps in the tracks). This could cause issues, since the height is not included in the positional data. When another train passes, a large area of the view could become occluded. The motion of passing trains could also result in messy and inaccurate reconstructions. Although filtering these out is beyond the scope of our research, we do examine the possibly negative effects.

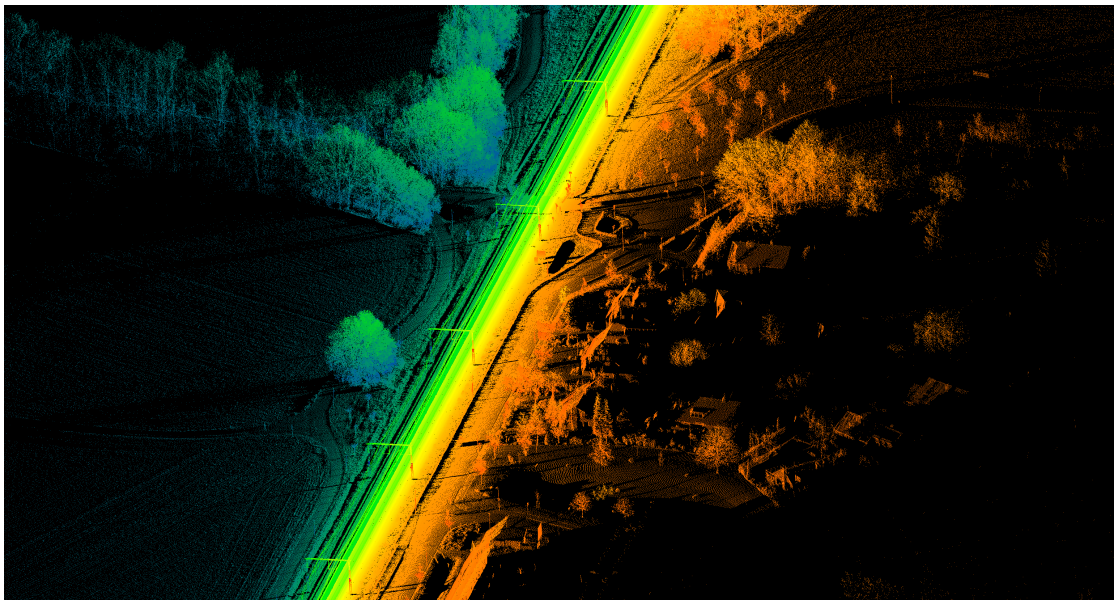


FIGURE 1.2: A section of laser scan point cloud data.

Related to the environment are the lighting conditions we take into account. In general, these conditions can vary largely, since video is recorded at different times during the day and under different weather conditions. Additionally, changes in lighting occur when passing through tunnels and train stations. For our research however, we assume lighting conditions to be good and remain constant.

1.1.2 Evaluation Data

In order to evaluate our results, we make use of existing point cloud data obtained through laser scans from measurement trains. Based on these scans, point clouds of the surrounding area of the train can be generated. This can be done up to a distance of a few meters in the direction perpendicular to the tracks. The position of points is determined using GPS and is stored as an absolute set of coordinates. An example is shown in Figure 1.2.

For our research, we consider this point cloud data to be the ground truth. We compare the results of our reconstruction to this data in order to obtain a distance measure that indicates the quality of the reconstruction.

1.2 Applications and Requirements

In this section, we will discuss the applications of our research within ProRail, including their requirements with respect to our evaluation criteria. As mentioned briefly before, the applications are simulation, architectural design and measurement. Since the requirements of architectural design and measurement are the same, we will group these two applications in one category called design.

The two applications have different requirements. For our research, we focus on the simulation application and therefore use its requirements. We will first describe each application individually. Next, we will outline the requirements and metrics used for our research, and explain how we evaluate these in Section 1.2.1.

- **Simulation.** For simulation, the main goal is to create a visually coherent environment. The environment should be realistic, but does not have to match the real world exactly. Unrealistic artifacts or holes in the reconstruction are more distracting to simulation users than inaccuracies in the exact location of objects. Important to note is that during simulation, the view is limited. This means that areas that fall outside of the view, such as the backs of signs, do not have to be taken into account.
- **Design.** For design, accuracy is more important than completeness. The result does not have to be visually pleasing, meaning small holes in the reconstruction are acceptable. However, colors are still required in order to identify objects. For design purposes, it is important that the environment can be freely navigated. Therefore, quality from any viewpoint and in any viewing direction has to be taken into account.

1.2.1 Requirements

To define our requirements, we use the following metrics:

- **Speed.** The number of frames that can be processed per unit of time.
- **Completeness.** The percentage of reconstructed area projected onto an image plane. This is relative to a viewpoint. Sky is excluded.

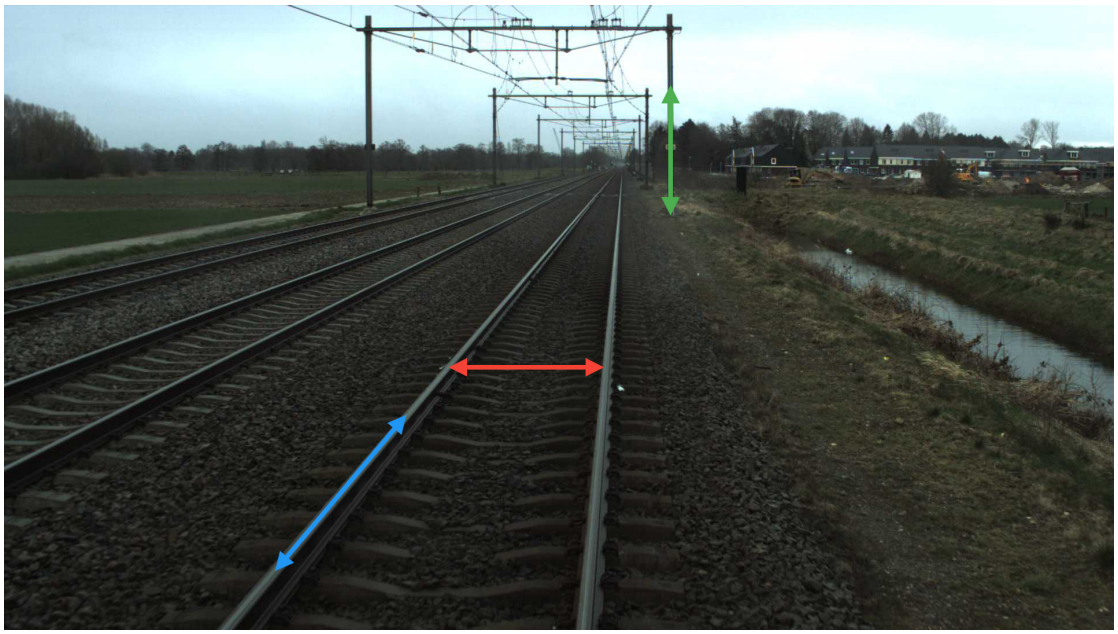


FIGURE 1.3: The directions in which we measure accuracy. Red: perpendicular accuracy. Green: vertical accuracy. Blue: parallel accuracy.

- **Accuracy.** The difference in location or size between the reconstructed models and the real world. Besides looking at the global accuracy, this measure can be split up into three different categories (See Figure 1.3):
  - **Parallel accuracy.** The accuracy in the direction of the movement. This direction is parallel to the railway tracks.
  - **Perpendicular accuracy.** The accuracy in the horizontal direction, perpendicular to the railway tracks.
  - **Vertical accuracy.** The accuracy in the up-down direction.

Though real-time processing is not required, our method has to be somewhat fast. Based on conversations with ProRail staff, we formalize this requirement by stating that one week worth of camera footage (approximately four hours) has to be processed in one week (168 hours). This implies a video to processing ratio of 1:42 on average. Based on an average speed of 100 kilometers per hour and a capture rate of 2 frames per meter, this leads to approximately 1.3 frames per second.

Completeness is most important for simulation purposes, where we aim for a 100% complete reconstruction. In order to fulfill this requirement, we will target 100% completeness by filling gaps using interpolation techniques. Therefore we will not evaluate

completeness in a quantitative manner, although it will partially become clear from our evaluation of accuracy. The potential negative effects of filling gaps on speed and accuracy should be taken into account, and will become apparent from other evaluations.

The parallel accuracy is less important than the perpendicular and vertical accuracy, as inaccuracies in the parallel direction have a smaller impact on safety. Therefore, errors up to 1m in the parallel direction are acceptable, while in the perpendicular and vertical direction errors up to 10cm are acceptable. In order to evaluate accuracy in each direction separately, we make use of certain known distances between landmarks based on measurements. To evaluate overall accuracy, we compare our results against 3D scan data, as mentioned before in Section 1.1.2. For the overall accuracy, we use a maximum average error of 10cm as our target.

Although it is important for our method to be usable in various environments, we focus on natural outdoor environments under daylight. These environments provide good lighting conditions. Other environments, such as tunnels and stations might violate some of our assumptions and therefore will not be considered when optimizing our algorithm. We do however examine the effects of these different conditions on the quality of the results.

As for the output of the algorithm, it is important that it can be visualized in a realistic manner. The output should therefore be a textured triangular mesh. Reconstructing the reflectance models of objects in the scene is not required. Color data is important and should be stored at each vertex. Although point clouds are an intermediate result, they are not suitable for the applications we focus on. They can however be used for other applications we will not consider here. Point clouds will be transformed into triangular meshes in the final steps of the algorithm.

1.3 Research Questions

Given the context and requirements of this research, we formulate our research question as follows:

Main research question. *How accurately can we create 3D reconstructions of railway environments based on a single front-facing camera mounted on a train, using structure from motion techniques?*

In order to answer this question, we select and implement a structure from motion algorithm based on our requirements and literature research. We test various parameter settings and strategies, particularly related to triangulation, surface reconstruction and post-processing to find the best results.

We will evaluate our results based on three criteria: accuracy, completeness and speed. In order to evaluate the accuracy of the results, we will compare them to point clouds based on 3D scans of the environment. We consider these point clouds to be the ground truth. Additionally, certain distances (such as the distance between rails) are known or can be measured and compared to the reconstructions. When necessary, we will use hole filling techniques to ensure 100% completeness. We can also gain information about the completeness of our results based on the accuracy evaluation. In order to evaluate the speed, we will run the candidate implementations on identical input data and the same machine. We formulate our variables and criteria in the following subquestions:

Subquestion 1. *How do different triangulation techniques perform in terms of speed and accuracy?*

Subquestion 2. *How do different surface reconstruction techniques perform in terms of speed and accuracy?*

Subquestion 3. *How do different filtering techniques perform in terms of speed and accuracy?*

These questions encapsulate the areas of our pipeline in which we will use various techniques. We aim to collect information about the effects of various parameters and methods on our final result, both in a quantitative and qualitative manner.

1.4 Contributions

Our main contribution is providing a method for 3D reconstruction of environments based on a single moving camera coupled with GPS data. Our method uses a fast and accurate optical flow technique combined with triangulation and surface reconstruction. We experiment with these techniques and their parameters to determine their influence on the quality and speed of the reconstruction. We carry out a quantitative evaluation and report the results of several variations based on quantitative data and qualitative

observations. We also evaluate potentially difficult cases that contain turns and passing trains qualitatively. We will conclude with recommendations for future research and the setup that is used to gather data.

1.5 Outline

The outline of this thesis is as follows. In Chapter 2 we will provide an overview of related work structured by topic. In Chapter 3 we will describe the setup of our research, including descriptions of the algorithms used. In Chapter 4 we will describe our evaluation methods. In Chapter 5 we will show our results based on evaluation. Finally, in Chapter 6 we will discuss our results with respect to our goals and answer the research questions. We will also talk about future work.

Chapter 2

Literature Review

The research topic of creating 3D models based on 2D images has generated a lot of interest in the past decades, mainly in the field of computer vision. An inherent problem with 2D images is their ambiguity. Since depth information is lost, the location of a point in the image can lie anywhere on a ray originating from the optical center of the camera. However, by combining information from multiple sources, much of this data can be recovered. Our research deals specifically with the use of multiple images from a single camera and motion data (GPS).

In the following sections we will address several topics that are closely related to our problem. In Section 2.1 we will introduce the general topic of structure from motion and describe the various steps that are required in subsections. In Section 2.2 we will address simultaneous localization and mapping, which shares many of the same issues with our research problem. Since we require distance measures between meshes for our evaluation, we will discuss this in Section 2.3.

2.1 Structure from Motion

Research in structure from motion (SFM), deals with several subproblems, which we will describe in this section. Therefore, complete SFM approaches combine various techniques. Most related work however, only deals with a specific subproblem. Application areas of SFM include robotics, biology, and geosciences.

Assumptions about the camera setup and calibration vary across different SFM techniques. The number of cameras and the movement of cameras can have a large impact on the requirements of the techniques. Furthermore, the image quality naturally influences the quality of the obtained results. Finally, knowledge of the intrinsic (calibration) and extrinsic (position) parameters of the camera(s) often leads to more accurate results if the parameters themselves are indeed accurate. This however implies a drawback of requiring calibration and localization of cameras, resulting is less dynamic applications.

SFM techniques usually provide a point cloud as output or intermediate result. Point clouds consist of a set of 3D points that represent the 3D structure of the scene. If there are no erroneous points, each point is located on the boundary or inside of an object in the scene. In our case, a boundary representation is sufficient, as this is enough to visualize the scene. Points can contain additional information such as color or texture of the volume they represent. Often, these point clouds are processed further, as there are many applications for which they are not suitable. We will further discuss this in Section 2.1.3.

Formal definitions of SFM vary across literature. We define the general SFM pipeline as follows:

1. **Point matching.** Find corresponding points in two or more views. This can be done sparsely (by only matching certain points, e.g. feature tracking), or densely (by matching all pixels, e.g. optical flow).

2. **Triangulation.** Given corresponding points from step 1, compute their 3D location based on their 2D location in different views. Camera calibration and motion data can be used in this step.

3. **Refinement.** In order to obtain the desired final model, a refinement step can be introduced. A common refinement process is outlier removal. Finally, the 3D data needs to be converted into the desired 3D structure (e.g. a mesh or depth map).

These steps will be described in detail in Sections 2.1.1–2.1.3. Since a number of techniques deal with the entire pipeline, we will discuss these in Section 2.1.4.

2.1.1 Point Matching

The issue of point matching, also known as the correspondence problem, is a key issue in many areas of computer vision. Over the years, many point matching techniques have been developed. Improvements have been made in terms of accuracy as well as speed. The choice of technique depends strongly on the requirements of the application and the assumptions that can be made. Roughly, point matching techniques can be classified in two different ways. Firstly, we can discriminate between dense matching and sparse matching techniques. Secondly, we can discriminate between narrow-baseline matching and wide-baseline matching techniques. We will first describe the properties of these categories.

Dense or sparse matching. Dense matching techniques compute correspondence for every pixel, while sparse matching techniques only do so for a number of pixels. Sparse matching techniques often select certain interest points that are detectable in a range of contexts, such as different scales, viewing angles or lighting conditions. These points are then represented using a descriptor and matched between frames based on the similarity of their descriptors. For dense matching, feature descriptors are often simplified in order to maintain efficiency.

Narrow- or wide-baseline matching. Narrow-baseline matching techniques operate under the assumption that the change in viewpoint between two frames is small. This is often the case when using video footage, as long as the camera does not move extremely fast. Narrow-baseline techniques can exploit this assumption by using relatively simple feature descriptors. Since rotation and scaling factors are generally small, the feature descriptor often does not have to be invariant to these types of transformations. Wide-baseline matching techniques can make fewer assumptions about the relation between views. These techniques apply to a more general case, where configurations can vary largely between two views. An example scenario would be a set of pictures of a building taken from different angles. Points can be translated, rotated or scaled in various ways. It is even possible for a number of points to only be present in one frame. Therefore, wide-baseline matching techniques have to be invariant to affine transformations as well as different lighting conditions.

We will now discuss two general point matching strategies. In Section 2.1.1.1 we will examine optical flow. Optical flow techniques are generally dense and narrow-baseline,

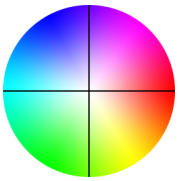


FIGURE 2.1: An encoding of optical flow used for visualization.

as they often deal with video and provide matches for all pixels. In Section 2.1.1.2 we look at feature tracking techniques, which are sparse and often wide-baseline.

2.1.1.1 Optical Flow

The first step in many SFM algorithms often consists of estimating the motion of points projected onto the image plane based on two or more frames of video. This information can be used to match pixels from different frames. This problem is referred to as the optical flow problem, and has seen extensive research ever since it was first addressed by Horn & Schunck [2] in 1981. The output of optical flow algorithms is a motion field that describes the motion for each pixel between two frames. This motion field is often visualized using hue to represent direction and saturation to represent velocity (see Figure 2.1). We will use this encoding throughout this thesis. Optical flow computation is essentially dense point matching, as its goal is to find the corresponding point in the next frame for each pixel in the current frame. Since the initial developments in optical flow research, a lot of progress has been made both in terms of speed and accuracy, as many new techniques have been proposed. Fortun et al. [3] provide an extensive survey of optical flow techniques, including recent developments. In recent years, quantitative evaluation of optical flow techniques has been made accessible through online benchmarks such as Middlebury [4, 5], MPI Sintel [6, 7] and KITTI [8, 9]. These benchmarks contain 127, 100 and 61 techniques respectively.

Many recent techniques [10–14] are still based on the variational method proposed by Horn and Schunck, but use various optimizations in order to achieve better results. These variational methods attempt to minimize an energy function consisting of a data term and a smoothness term. The data term is used to express some sort of constancy constraint (often image brightness or gradient) for a point at different times. However, the resulting equations are under-constrained and therefore can not be solved locally. This is known as the aperture problem. In order to solve this problem, the smoothness

term is introduced. The smoothness term expresses the way the optical flow field varies between neighbouring pixels, or over the entire image. Sun et al. [15] researched the effect of different energy functions and parameters in order to examine their contributions to the quality of the solution. An important element they found was the positive effect of median filtering of intermediate flow fields. This median filter makes methods more robust as it has the capability of removing flow outliers.

Many classical techniques have issues detecting large motions. This is often a result of the assumption of subpixel motion (velocities of at most one pixel per frame). This problem is often dealt with by using a multi-scale coarse-to-fine approach (e.g. [11, 15–17]). With this approach, an image scale pyramid (often a Gaussian pyramid) is constructed. Optical flow is first computed on the coarsest resolution. The computed flow is then upsampled and used as initialization for the next pyramid level. This process is repeated until the original resolution is reached. This approach can be used to effectively detect large motion, but often sacrifices motion details in areas with thin structures, as these structures are lost in lower resolutions. The use of sparse feature detection can lead to improvements [18] but requires extra computations. Alternatively, nearest neighbour fields (NNFs) can be used to detect large motion without sacrificing performance in detailed areas [19]. NNFs are computed over the entire image and therefore are not restricted to small motion. Approaches based on this technique rank high on the Middlebury benchmark, but often come at a high computational cost, requiring several minutes per frame, even when using more efficient approximate nearest neighbours implementations. This would make the reconstruction of large scenes an immensely time-consuming task. Many recent techniques [20, 21] employ the PatchMatch algorithm [22] as a faster approximation for NNFs. These techniques can achieve performance near the state-of-the-art, with a runtime of only a few seconds per frame. Given our requirements, these techniques appear very promising.

2.1.1.2 Feature Tracking

Feature tracking is a naturally sparse technique. It relies on keypoint detection and matching. Detected points are often corner points [23] or regions that differ from their surroundings [24, 25]. Other locations such as edges or uniform regions are difficult to match between images because of ambiguity. Detected points are described using

a feature vector. Feature descriptors differ in terms of their robustness to different types of transformations. SIFT [26] is a widely-used scale- and 2D rotation-invariant descriptor. A more recent detector and descriptor is SURF [27]. SURF is scale- and rotation-invariant, and can be computed faster than most descriptors. Many different detectors and descriptors exist and can be selected depending on the application. An extensive survey was written by Tuytelaars et al. [28].

After extracting interest points and their descriptors, points are matched between images based on the similarity of their descriptors. In some cases, the new location of interest points can be roughly predicted based on their estimated 3D locations and the camera pose estimation [29, 30]. The similarity measure depends on the descriptor used. Often, the cross-correlation or sum of squared differences is used. If descriptors can be computed efficiently, points can be compared to all pixels (optionally inside a predicted region) instead of only previously detected interest points [31].

Given the lower number of points available (generally fewer than one thousand points), SFM methods based on feature tracking generally provide less detailed results compared to dense methods. As a result, small geometric structures can be lost. In order to visualize the results, meshes can be generated based on a few points.

2.1.2 Triangulation

In our context, triangulation refers to the computation of the 3D location of a point based on its positions in two or more camera images. Assuming the camera intrinsic and extrinsic parameters are known, this problem is trivial to solve in theory. One can simply compute the two rays through the optical centres of the cameras and the point and find their intersection. In practice however, this is usually not possible due to noise, errors in localisation or calibration and finite precision.

Triangulation is possible with uncalibrated cameras in unknown configurations, based on a number of corresponding points in both views. An algorithm to compute the 3D structure of a scene, up to an unknown scale factor, has been described by Longuet-Higgins [32]. This problem is solvable using eight points with the exception of a few degenerate cases. When the intrinsic parameters of the cameras are known, a solution can be found using only five points [33]. When certain additional information is available,

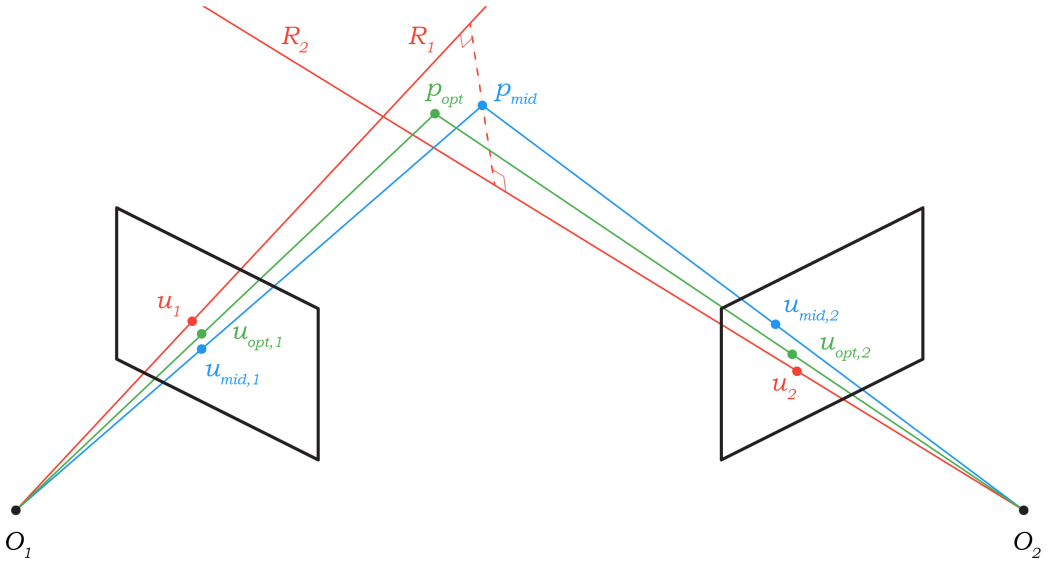


FIGURE 2.2: Schematic example of a case where the midpoint method yields suboptimal results in terms of 2D reprojection error. O_1 and O_2 are the camera optical centers. The camera image planes are depicted by quads. u_1 and u_2 are the projections of a 3D point on respective image planes. R_1 and R_2 are the camera rays through these points. p_{mid} is the midpoint of the common perpendicular of these two rays (midpoint method). $u_{mid,1}$ and $u_{mid,2}$ are the projections of p_{mid} onto the image planes. As observed, the midpoint method is suboptimal, since a point p_{opt} exists which has smaller reprojection errors with respect to u_1 and u_2.

such as a common direction between two views, less points are required [34]. Although interesting for other applications, we will not discuss uncalibrated camera setups in further detail, as it is beyond the scope of our research.

The geometric relations between two cameras are described by epipolar geometry. The projection of the optical center of one camera onto the image plane of another is called an epipole. If a point X is visible for both cameras, a line through the optical center of one camera and the point can be drawn. The projection of this line onto the image plane of the other camera is called an epipolar line.

In the case where the camera intrinsic and extrinsic parameters are known, we can use these to compute a ray through the optical centre of the camera and the projection of the point onto the camera's image plane, based on its pixel location. If we do this for every view the point appears in, we obtain a number of rays. We then assume the 3D point we aim to reconstruct lies on the intersection of these rays. Since the probability of these rays intersecting exactly in one point is low due to aforementioned factors, we must find

a way to estimate the location of the point. An intuitive way would be to simply take the midpoint of the line segment perpendicular to the two rays (the midpoint method [35]). Although this might be an acceptable method in some cases, it is not valid in the projective case and provides relatively poor results [36, 37]. An example of a case where the midpoint method is suboptimal can be seen in Figure 2.2.

Hartley & Sturn [37] provide an overview of several triangulation methods and propose a new noniterative method which they call the polynomial method. This method minimizes the squared 2D reprojection error of the reconstructed 3D point and the two image points. It does so by parametrizing the epipolar lines using a single parameter t. The error function is then defined as a function of t. This function has up to 3 local minima and 3 local maxima. The global minimum is found by evaluating the function at the roots of its derivative. Under the assumption that the image points are perturbed by Gaussian noise, this method is optimal. The methods were evaluated using artificial configurations of points and camera positions with added Gaussian noise. Evaluation based on real images was also provided. The authors conclude that their method provides the best results in terms of 2D and 3D errors. Minimizing the absolute values of distances instead of the squared distances provides slightly better results. A disadvantage is the higher time complexity of these methods compared to others. The midpoint method is fast but doesn't provide good results. Other methods (Linear-Eigen, Linear-LS, Iterative-Eigen and Iterative-LS) provide varying results in terms of transformational invariance, time complexity and error measures. A choice should be made between these methods based on the requirements of the application.

2.1.3 Refinement

As a final step in the pipeline, it is often required to refine the obtained 3D information (usually in the form of a point cloud). This refinement step often consists of filtering outliers, merging multiple reconstructions or converting to the required 3D representation. A simple but often quite effective approach to outlier removal, is simly removing all points with less than a certain number of neighbours within a certain radius. A slightly more advanced method is described by Rusu et al. [38]. This method first computes the mean distance from each point to its k nearest neighbours. It then calculates the mean $\overline{d}$ and standard deviation σ of these distances over all points. Points with average distance

larger than $\overline{d} + ak$, where a is some parameter (commonly equal to 1.0), are filtered out. Rusu et al. [38, 39] also describe an adaptation of the Moving Least Squares (MLS) algorithm they call Robust Moving Least Squares (RMLS). This algorithm smooths the surface and can perform normal estimation. It is also very useful in merging overlapping layers that resulted from merging multiple point clouds.

A very well-known technique for model fitting and outlier filtering is RANSAC [40]. The RANSAC approach is to represent a model using the minimum amount of required data, which is randomly selected. Next it finds the set of data points that are close enough to this model (inliers). If this set is large enough, the model is refined based on this dataset, otherwise a new model is chosen randomly. RANSAC is an iterative algorithm that finds reasonable results only with a certain probability. This probability increases with the maximum number of iterations chosen. Over the years, various RANSAC-related techniques have been developed, many of which still closely resemble the original technique. An example of this is MLESAC [41]. MLESAC employs the same strategy as RANSAC, but maximizes the likelihood as opposed to the cardinality of the set of inliers.

Conversion from point clouds to meshes is often referred to as surface reconstruction. Various algorithms exist to achieve this. Marching Cubes [42] is a famous algorithm originally designed for the conversion of discrete scalar fields to triangular meshes. Thus, it would require a conversion from point clouds to discrete scalar fields to be useful in our case. Hoppe et al. [43] proposed an algorithm that computes a signed distance function from a point cloud. the zero set of this function represents the surface to be reconstructed. To generate the mesh, a variation of the Marching Cubes algorithm is used. Poisson surface reconstruction [44] is based on a similar principle. It computes an indicator function using an octree. The function equals 1 inside of the mesh and 0 outside of the mesh. This method does however require the normals of the points to be known. The Point Cloud Library (PCL) by Rusu & Cousins [45] provides great tools and implements greedy projection triangulation [46], Marching Cubes [43, 47], Grid Projection [48] and Poisson [44] surface reconstruction algorithms. It also includes normal estimation required by these methods.

2.1.4 Complete SFM Techniques

The work of Pollefeys et al. [49] is very similar to ours in spirit. Their work is aimed at creating 3D reconstructions of urban scenes in real-time. With examples they show reconstruction of multiple streets based on video from a camera mounted on a vehicle. GPS and inertia measurements can be used in their system to determine location and are combined with camera pose estimation using the Kalman Filter [50]. Similar to this research, the environment is assumed static. Furthermore, the camera calibration is assumed to be known. In their work, an extended version of the KLT tracker is used that works in real-time and estimates camera gain simultaneously. This gain estimation allows the tracker to remain accurate even when lighting conditions change, as is common in urban environments. The technique constructs and merges depth maps and creates a triangular mesh using plane sweeping. In this process, gaps are filled and the resulting mesh is smoothed. the The authors mention a common problem with stereo reconstruction, where parts of the background (such as the sky) often end up attached to foreground surfaces. This problem is circumvented by creating a color model for the sky and filtering out pixels that fit this model.

Another method targeting real-time reconstruction is described by Newcombe & Davison [51]. This method however, focuses on indoor environments and a handheld camera. Their approach is to estimate the camera pose in real-time based on a coarse mesh of the environment. Next, variational optical flow is used to generate dense pixel correspondences and highly accurate depth maps. A three pixel median filter is applied to the depth maps to remove outliers. Depth maps are then fused to recreate the entire scene. This fusion step is straightforward due to high quality of the depth maps and consists of simply removing overlapping regions and connecting gaps.

2.2 Simultaneous Localization and Mapping

SFM is closely related to the Simultaneous Localization and Mapping (SLAM) problem. SLAM is concerned with the problem of building a map of the environment based on sensor data, while simultaneously determining the position of an agent (e.g. a robot) navigating the environment. SLAM is a very general topic and is not limited to a specific type of sensor. Traditionally, work in this area has focused mainly on laser scanners as

sensors. More recent work however has also examined the use of cameras. In gerenal, SLAM often uses landmarks detected in some fashion. SLAM techniques generally maintain a state vector, containing information about the agent and landmarks. This information is then updated and refined using the following loop:

1. **Predict.** The next state is predicted based on a model of agent's motion and control system.
2. **Measure.** The next state is measured based on sensor input.
3. **Update.** The predicted and measured states are combined into a new state vector.

For this research or similar ones, certain principles used in SLAM could be used in order to make up for inaccuracies in input data such as GPS. In our case however, GPS data was accuracte enough without modification. These principles are applicable to SFM techniques where motion data is inaccurate or not accessible. Techniques like this can be considered part of a gray area between SLAM and SFM. Many SLAM techniques [29, 52–54] employ the Kalman Filter. The Kalman Filter is used to estimate the next state of a linear system by combining observations and predictions, instead of using either of these. This way, more accurate estimations can be made. As an example, Dissanayake et al. [52] use the Kalman Filter to estimate the location of a vehicle and a number of landmarks.

Work related to our research uses SLAM based on a single moving camera. Davison [54] provides a technique for a single moving camera in an unknown scene. The work of Davison et al. [29] builds upon this approach. The proposed techniques assume constant velocity and constant angular velocity for their statistical model, and use the Extended Kalman Filter to estimate successive states. Both these works focus heavily on real-time performance, as this is required in many applications in robotics. This results in only building a sparse map of features (around 100 features can be tracked at 30 FPS). Additionally, probabilistic approaches are used in various aspects of the methods in order to increase efficiency. A very interesting part of their approach is their prediction of feature locations in successive images based on the estimated camera position, narrowing down the search region and therefore reducing computational costs. This idea is based on the principles of SLAM, but not necessarily part of the SLAM pipeline. In order

to determine 3D locations of points, as well as camera parameters, bundle adjustment is often used. This technique is not very suitable for real-time applications however, as multiple frames are required to obtain accurate results. Therefore, the mentioned techniques only maintain a single state. This state contains information used for the predict step and is updated after every frame. This is done in such a way that past information is not explicitly stored, but not entirely discarded either.

2.3 Point Cloud and Mesh Distance Measures

In order to evaluate the accuracy of our methods, we will compute the distance between the reconstructed data and measured 3D data. A commonly used distance measure is the Hausdorff distance. Aspert et al. [55] provide a method to compute the distance between two triangular meshes based on the symmetrical Hausdorff distance. This distance d_s is computed between surfaces S and S' as follows:

$$d_s(S, S') = \max[d(S, S'), d(S', S)] \tag{2.1}$$

This symmetrical distance is more accurate than the distance in either direction, as these two distances could differ largely. The distance between two meshes can be computed in a similar fashion, by taking the maximum of the backward and forward distances. The distance from a single point to a mesh can be straightforwardly computed by taking the minimum of the distances between the point and all the triangles in the mesh. In order to compute the distance between a triangle and a mesh however, sampling of the triangle is used, since there is an infinite amount of points on a triangle. Aspert et al. also provide an implementation of their method. In the implementation, space partitioning in the form of a uniform grid is used in order to reduce the amount of required distance evaluations.

For our evaluation, we compute the distance between reconstructed meshes and a ground truth point cloud. We do not convert the ground truth point cloud to a mesh, since the surface reconstruction algorithm would affect the resulting mesh in this case. In this scenario, we cannot use the distance measure by Aspert et al., since it is designed to compute mesh to mesh distances. Therefore, we use a slightly different approach

by computing the point cloud to mesh distance (forward distance) [56] and the mesh to point cloud distance (backward distance). For the latter, we use the vertices of the mesh as a point cloud and use the local modelling tool provided by CloudCompare [57] in order to achieve more accurate results.

Chapter 3

Methods

In this chapter, we will describe our approach in detail. We provide an overview of our method in Section 3.1. In Sections 3.2–3.4 we will discuss the steps used in our method. In Section 3.5 we will discuss our baseline algorithm. In Section 3.6 we will show the potentially interesting areas in which we can vary upon this algorithm. In Section 3.7 we will describe the way we implement the required techniques.

3.1 Method overview

For our research, we implement the SFM pipeline as described in Section 2.1. This pipeline suffices because camera calibration and motion are known. Therefore, we do not need to use SLAM techniques. Since the SFM pipeline is very general, we select specific techniques for each of the subproblems based on our domain. Our approach is to define a baseline algorithm and vary one parameter or technique at a time, much like Sun et al. [15].

In contrast to Pollefeys et al. [49] and Newcombe & Davison [51], real-time performance is not required. Therefore, we can sacrifice speed for accuracy when selecting techniques. Newcombe & Davison are also concerned with camera pose estimation, which is not required in our case. For these reasons, we choose not to replicate or implement the frameworks outlined in these works.

The input is processed on a frame-by-frame basis. Since our goal is to create detailed reconstructions, we expect a dense point matching technique to deliver the best results.

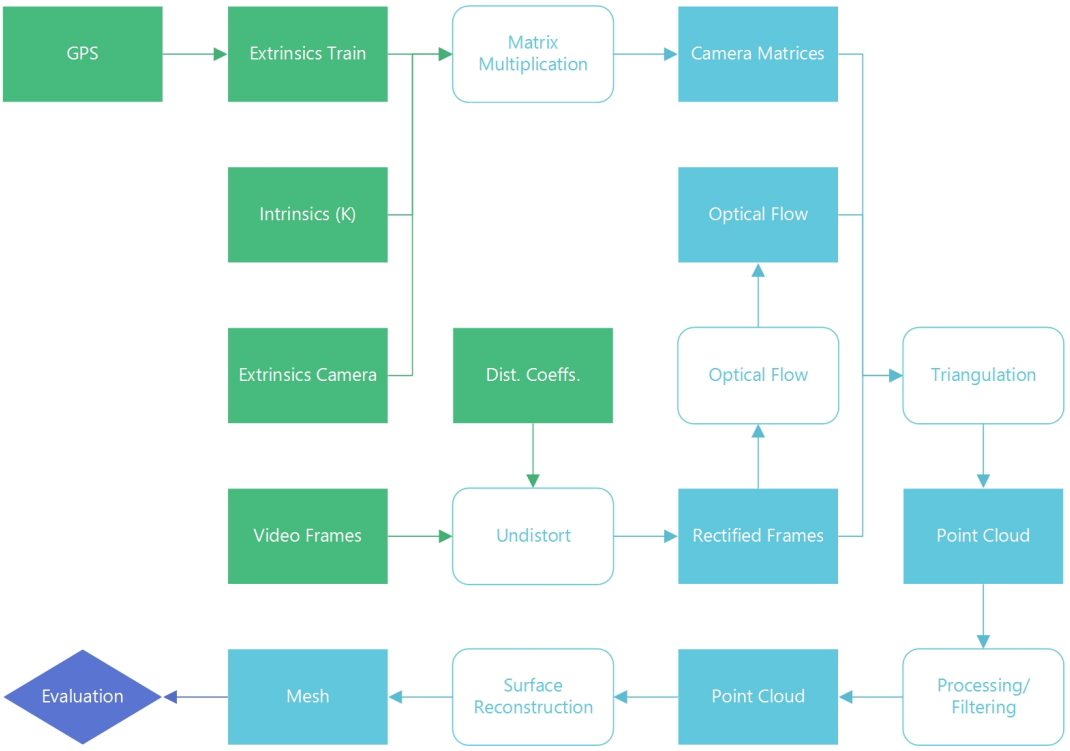


FIGURE 3.1: The reconstruction pipeline. Input data is marked green.

Furthermore, looking at our input data, it is clear that it is narrow-baseline, as the measurement train typically only moves half a meter between frames and image points move approximately 100 pixels at most. These observations point towards the use of optical flow techniques for point matching, as these are naturally dense and narrow-baseline. For each frame, we compute the optical flow with respect to the next frame. Based on the optical flow, each pixel in the current frame can be matched to a point in the next frame.

We compute the location of the 3D point corresponding to a pair of image points using triangulation based on GPS data and camera parameters. Because we use a dense matching technique (optical flow), we can generate a lot of 3D points. We expect this to yield more detailed results while taking more time than sparse matching techniques. Since we do not require a real-time implementation and we select a relatively fast optical flow technique, this is not expected to impose problems. After processing several frames in a video sequence, we end up with a point cloud representation of the 3D environment. This point cloud is converted to a triangular mesh such that we can visualize the results. In this conversion process, gaps are filled and outliers are removed automatically by most techniques. A visualization of the full reconstruction pipeline can be found in Figure 3.1.

FIGURE 3.2: An example image from the KITTI data set. Image courtesy of Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago.

3.2 Optical flow

In order to select a specific technique from the large amount of optical from techniques published, we look into the various benchmarks discussed in Section 2.1.1.1 (Middlebury, MPI Sintel and KITTI [4–7, 9, 58]). Comparing the different optical flow techniques is not necessarily straightforward, as benchmarks differ in terms of their metrics and input. The speed is especially difficult to evaluate as different machines have been used to run the implementations, and in some cases speed is not documented. We can however use the times mentioned in the Middlebury and KITTI benchmarks as a rough guideline. From these times it becomes apparent that most of the highest performing techniques are not suitable, having a running time of several minutes per frame. We require a running time of at most a few seconds in order to meet our requirements. Another difficulty is the fact that some techniques are very recent and have not yet been well documented.

By looking at the input images used by all three benchmarks, we conclude that the KITTI benchmark data is most comparable to our research. This benchmark uses images taken by a front-facing camera from a driving car (see Figure 3.2). Important to note is that this benchmark uses stereo camera data. Techniques that make use of both images are marked as such and will be excluded for our research since we only use a single camera.

From the available data, we select one technique to be the best fit for our research. This appears to be *Edge Preserving Patch Match without Hierarchical Matching (EPPM w/o HM)* introduced by Bao et al. [20]. Although *ComponentFusion* and *SVFilterOh* perform better on the Middlebury benchmark and also fit our speed criteria, a description

of these techniques has not yet been made publicly available. We therefore select *EPPM w/o HM*. This technique is based on patch matching and uses clever approximation schemes to achieve fast performance. Another advantage of this technique is that it is strongly edge-preserving. This is important for separating objects that are located at different distances. In the following subsections, we will describe the steps used by this technique.

3.2.1 Patch Matching

The first step in the algorithm is to compute an NNF estimation using an adapted version of the PatchMatch algorithm. Assuming we have two patches centered around location $\boldsymbol{a}$ and $\boldsymbol{b}$ respectively, the matching cost between the two patches is defined by Bao et al. as:

$$d(\boldsymbol{a}, \boldsymbol{b}) = \frac{1}{W} \sum_{\Delta(\Delta_x, \Delta_y): |\Delta_x| \leq r, |\Delta_y| \leq r} w(\boldsymbol{a}, \boldsymbol{b}, \Delta) C(\boldsymbol{a}, \boldsymbol{b}, \Delta) \tag{3.1}$$

Here, W is a normalization factor, r is the patch radius, w is a weighting function that prioritizes pixels closer to the patch center and pixels that are similar to the center pixel, and C computes the cost between two pixels robustly.

Given the complexity of PatchMatch and the required patch size, computing this formula exactly takes too much time. Therefore, an approximation is used. Instead of using all pixels inside a patch to compute the cost, only the n pixels most similar to the center pixel are used. In the original work, a patch size of 35×35 pixels is used. $n = 50$ is found to provide the best trade-off between quality and efficiency. This is obviously a much smaller number than $35 \cdot 35 = 1225$. Computing the n most similar pixels in the patch for each pixel is still too slow however. Therefore, a strategy similar to PatchMatch itself is used. First, for each pixel, n random pixels are selected from its patch and stored in a vector. Next, the image is scanned from top-left to bottom-right. For each pixel, the vectors of neighbouring pixels are merged into its vector based on similarity. Finally, another scan is performed in reverse direction in the same manner. This approach is stated to be about 6 times faster than selecting the n most similar pixels exactly [20].

Afterwards, the original PatchMatch algorithm is applied based on the matching cost in Equation 3.1 and only using the pixels selected in the previous step. The algorithm uses random initialization, as is customary for PatchMatch.

3.2.2 Hierarchical Matching

In order to efficiently perform patch matching on large images, images are downsampled first. This is typically done twice using a scaling factor of 0.5. The NNF is computed on the coarsest scale and upsampled to a higher resolution. Local patch matching is then performed to refine the NNF. As with other coarse-to-fine schemes, this approach has the risk of not detecting motions for small structures correctly. This risk is present if we downsample to the point where these structures are not or barely visible. In our case, most small structures are located far from the camera in world space. Many of these structures do not have to be reconstructed at this point in time, as the train will pass them at some point. For objects situated far away from the railway tracks, a high level of detail is not essential for our purposes. Therefore, we examine various scaled images to ensure no essential detail is lost.

3.2.3 Handling Occlusions and Outliers

In order to handle occlusions, the NNFs in both directions are computed between two frames. The NNFs are compared and inconsistent pixels are fixed based on nearby pixels. Pixels that can not be fixed are treated as outliers. To handle outliers, a weighted median filter is applied to the computed NNF. Afterwards, a second consistency check is applied.

3.2.4 Subpixel Refinement

Since our input data is very narrow baseline, optical flow values are typically only a few pixels. With such small motions, slight errors in optical flow can lead to larger errors later on. Therefore, subpixel accuracy is required. In order to compute the optical flow up to subpixel accuracy, the matching cost of a patch in its surroundings is assumed to follow a paraboloid surface:

$$d = f(x, y) = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6] \cdot [x^2, y^2, xy, x, y, 1]^\top \tag{3.2}$$

Where $\theta_1 ... \theta_6$ are the unknowns and (x, y) are the pixel locations of m surrounding points in a square grid. By substituting these points into the equation, a linear system is obtained which can be solved. The location (x^*, y^*) of the patch up to subpixel accuracy can be computed using:

$$x^* = \frac{2\theta_2\theta_4 - \theta_3\theta 5}{\theta_3^2 - 4\theta_1\theta_2} \qquad \text{and} \qquad y^* = \frac{2\theta_1\theta_5 - \theta_3\theta 4}{\theta_3^2 - 4\theta_1\theta_2} \tag{3.3}$$

To increase accuracy even further, the cost can be computed on an upsampled version of the image instead. This is a computationally inexpensive step that simply requires interpolation of pixel values and does not add any significant overhead in terms of time [20].

3.3 Triangulation

Triangulation can be seen as the step from 2D to 3D, and therefore is a very important step in the pipeline. It has already been discussed in Section 2.1.2 and appears to be a largely solved problem. Therefore we consider the methods described by Hartley & Sturm [37]. From their evaluation it appears the polynomial method provides the best results but is also computationally more expensive than many other methods. Additionally, we use the simpler midpoint method to examine potential trade-offs in terms of speed and accuracy.

3.4 Surface Reconstruction

In order to make the results more visually appealing and usable for simulation purposes, we wish to generate a 3D mesh based on the reconstructed point cloud. As discussed in Section 2.1.3, many different surface reconstruction techniques exist. Implementations of some of the most popular ones [43, 44, 46–48] are available through PCL. For our baseline algorithm, we choose greedy projection triangulation [46], as this method appears to be

the most naive, but fast. This method is relatively fast due to its greedy approach. It attempts to create a surface using all the points in their original locations. This could lead to holes in the reconstruction or very apparent noise. From the literature, it appears that the algorithms make different assumptions about the input data and desired output. It is not intuitively clear which algorithm will perform best in our case. Therefore, we evaluate these different methods to determine the one most fit.

3.5 Baseline Algorithm

As a baseline algorithm, we combine the techniques described above into a single pipeline, using the chosen settings. For parameters, we select values based on literature or small pre-evaluations (described in Section 4.2). The values we use for the various parameters are shown in Appendix A. Our baseline algorithm can be described by the following steps:

1. Compute the optical flow using the technique by Bao et al. [20] We use the settings mentioned in the original paper with some adaptations.
2. Remove pixels with low optical flow or high brightness in order to filter out the sky and distant regions. Thresholds are set based on pre-evaluation (Section 4.2).
3. Remove edge regions in order to filter out inaccurate optical flow values. Borders are set based on pre-evaluation (Section 4.2).
4. Triangulate matching points between frames using the midpoint method.
5. Construct a triangular mesh from the remaining points using greedy projection triangulation. Fill gaps by connecting points around the gaps.

We expect this algorithm to provide decent results, as well as a baseline against which we can compare the other variations in order to determine their effect on the results based on our criteria.

3.6 Variations

Based on our baseline algorithm, we generate and evaluate several variations. For these variations, we choose parts of the algorithm that appear interesting given our input data and goals. Below, we discuss each area in which we apply variation and include the different settings used. We name each variation so we can reference them easily. The names of these variations are shown in parentheses.

3.6.1 Triangulation

For triangulation, we simply evaluate two methods, the midpoint method (included in the baseline algorithm) and the polynomial **(t-poly)** method. We aim to find the trade-off between complexity and performance between these two methods in order to determine whether or not it is worthwhile to use the more complex polynomial method.

3.6.2 Post-Processing

Errors in optical flow can lead to outliers in the 3D results. Since we have a lot of points to work with, removing some of them should not cause any issues. We test various post-processing steps aimed at removing outliers and filling gaps. For our variations, we will consider the following post-processing steps:

- **Statistical outlier removal (p-stat).** Outliers are removed using the statistical method described by Rusu et al. [38].
- **Radius-based outlier removal (p-radius).** Points are filtered based on the number of neighbours within a certain radius.
- **Robust Moving Least Squares (p-rmls).** Apply the RMLS technique by Rusu et al. [38, 39].

The statistical and radius-based filters rely on parameters that will require tweaking. As long as we don't remove too many points, these methods are expected to remove noise and therefore improve accuracy. By merging multiple independently reconstructed point clouds, large areas could overlap. This leads to ambiguity, slight variations in height

and noise. To reduce these undesired results, we use the RMLS method by Rusu et al. [38, 39]. This method is expected to reduce the number of points and create smoother surfaces in areas with or without overlap.

3.6.3 Surface Reconstruction

For surface reconstruction, we test each of the methods available through PCL. These methods are greedy projection triangulation [46] (included in the baseline algorithm), Marching Cubes [43, 47] **(s-mc)**, Grid Projection [48] **(s-grid)** and Poisson surface reconstruction [44] **(s-poisson)**, and have been discussed in Section 2.1.3. For the s-grid and s-poisson variations, we also apply the statistical outlier removal to the point cloud before surface reconstruction. This is necessary, as these implementations are not able to deal with the amount of noise present in the unprocessed point cloud. As a result, these two variations can not be directly compared to the others without taking into account the outlier removal step.

3.7 Implementation

In this section we will walk through our pipeline and briefly describe our implementation for each step.

For the optical flow (EPPM [20]), we use the implementation [59] kindly provided by author of the original paper Linchao Bao. We make minor modifications in order to link this project to our pipeline.

We have two variations for our triangulation algorithm. The first is the midpoint method, which we implement ourselves. The second is the polynomial method by Hartley & Sturm. For this method, we use code provided by Roy Shil [60].

For surface reconstruction and refinement, we use the Point Cloud Library (PCL) [45]. PCL is a C++ library for point clouds and 3D geometry processing. This makes it a very useful tool for our implementation. PCL contains many operations, including point cloud processing (statistical, radius-based and RMLS) and surface reconstruction. We use this library to perform various operations on point clouds and meshes.

For many intermediary or debugging steps, such as I/O, visualization and computations, we make use of the OpenCV and PCL libraries and their dependencies.

Chapter 4

Evaluation

In this chapter, we will describe our evaluation methodology. In Section 4.1 we will describe the data we use for our evaluation. We will describe our pre-evaluation and explain our process of selecting parameter values and implementation details in Section 4.2. In Section 4.3 we will describe our quantitative evaluation process for each metric. In Section 4.4 we discuss how we evaluate performance for special cases highlighted in Section 1.1.1 (turns and passing trains), as well as some additional evaluations.

4.1 Data

Our evaluation data consists of a set of video data combined with GPS data and camera parameters. This set consists of a segment of 100 meters (202 frames) of a train run near Holten (evaluation segment). Point clouds generated from laser scans are available for this train run as well. An example frame is shown in Figure 4.1. We select this segment because it contains various types of objects, such as a building and trees. Additionally, the distances between the railroad and the objects vary. We use different sets of data for testing and evaluation to prevent over-fitting. In order to define a baseline algorithm and find appropriate values for parameters, we use data recorded near Groningen.

We use the baseline algorithm as described in Section 3.5 and create different variations in order to construct different 3D models of the environment. We then evaluate all variations based on our criteria. It is again important to note that we use the statistical outlier removal for the s-grid and s-poisson variations.

FIGURE 4.1: An example frame of the train run we use to conduct our evaluation.

4.2 Pre-Evaluation

Due to the large amount of parameters available and lack of evaluation data, running complete evaluations for every single parameter instance would be infeasible. Choices for these parameters are occasionally clear-cut based on literature as well as generated results. However, for most parameters, we stray from the original literature and run small tests to determine their values based on observation and our evaluation criteria. In Section 4.2.1, we describe our tests with the optical flow and address some additional steps we took in order to deal with some important issues. Additionally, we outline some issues with the input data and solutions in Section 4.2.2. In Section 4.2.3, we will explain our thought process for the settings for remaining parameters.

4.2.1 Optical Flow Parameters and Filters

Based on observation of several outputs of the optical flow computation, the largest optical flow magnitudes typically lie between 70 and 80 pixels and occur around the edges of the image. These regions are generally closest to the camera as the camera faces in the direction the train is moving and the tracks are devoid of obstacles. This indicates that pixels less than 70 pixels from the edge of the image often don't have a match in the next frame. On one hand, regions near the camera are expected to generate

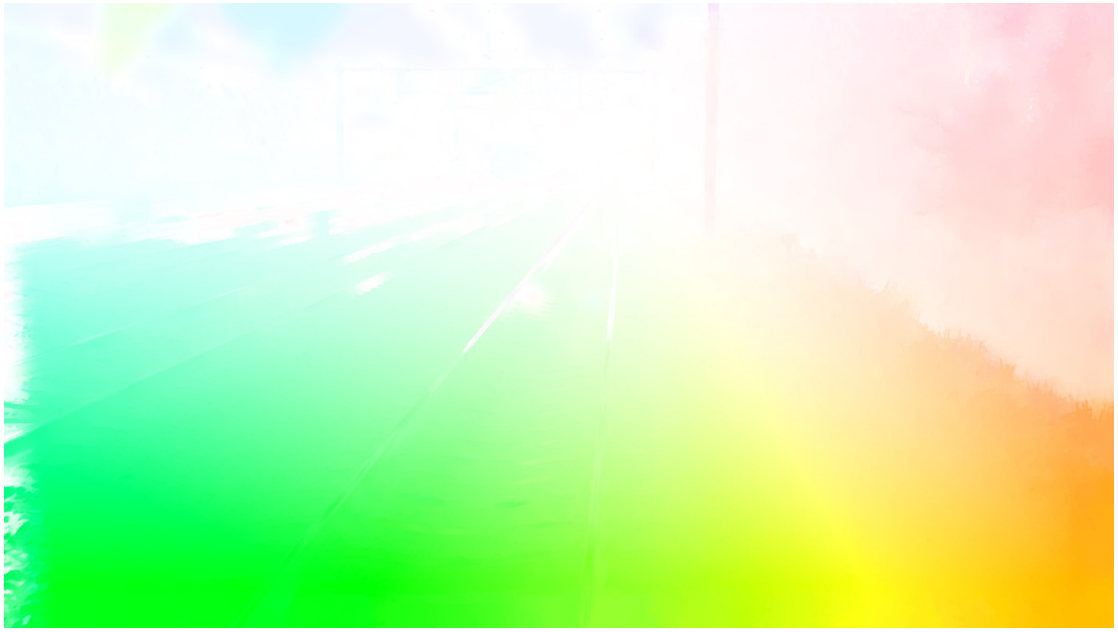


FIGURE 4.2: A visualization of optical flow where errors occur near the left edge of the image. The entire bottom left quadrant of the image consists of ground and tracks. Therefore, the optical flow in this area should not change much between pixels. This is obviously not the case near the left edge. Errors like this are not always present on all edges, but can occur due to the random nature of the optical flow technique.

better results since small errors have little effect if the optical flow values are large. On the other hand, the optical flow for these pixels is mostly based on surrounding pixels and can contain large errors, as for example shown in Figure 4.2. These errors are easily observable in the optical flow, as well as the resulting 3D reconstruction. Therefore, we crop the image by 70 pixels after computing the optical flow in order to filter out these errors.

In order to filter out the sky and distant regions where the optical flow is less accurate, we perform two steps. First, we do not consider pixels with an optical flow vector smaller than 2 pixels. For points with low optical flow values, small errors can cause large problems after triangulation. Often, these points end up floating in the air, or even behind the actual camera position. One potential downside of this filter is that objects that are never close to the camera will not be reconstructed. Second, we remove any pixels with a color above a certain brightness threshold. This threshold is dependent on factors such as weather conditions and can be changed for different recordings. In our case we use a maximum of 50% brightness. Combined, these two filter steps appear to remove all sky and very distant regions, while leaving other areas mostly untouched (see Figure 4.3). Small parts of the tracks and other objects that do get removed should

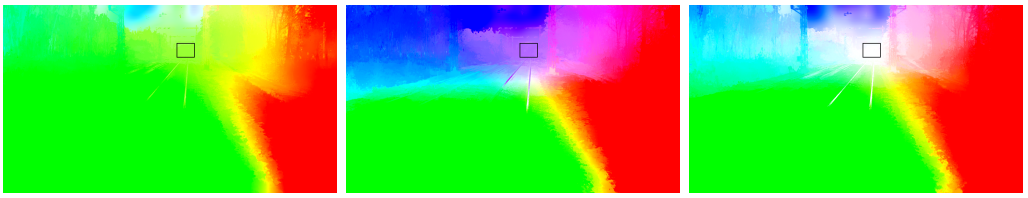
FIGURE 4.3: A visualization of the pixels that are filtered out by the optical flow threshold (yellow), brightness threshold (red) or both (orange).

not cause many issues, since these areas can be easily filled in the surface reconstruction step. For environments with brighter colors (i.e. containing white buildings), a more advanced color model would be desirable. However, we consider this beyond the scope of our research.

For the refinement step in the optical flow computation, particularly the weighted median filter, we choose parameters to generate a relatively large amount of blurring. Without this, the optical flow shows regions of outliers, particularly on certain areas of railway tracks. This is likely due to little texture and repeating patterns that cause ambiguity. Therefore, we choose a median filter radius of 24 pixels, and higher values for σ_s and σ_r ($\sigma_s = 0.75r = 5.25$ and $\sigma_r = 2.5$) than stated in the original paper. This way, the median filter is slightly less edge-preserving and weighs pixels further from the patch center higher than usual. Overall, this creates a larger amount of blur which eliminates more outliers, while still preserving significant depth edges.

4.2.2 Input Data Issues

From some initial runs with our evaluation data, it became clear that it contained some inaccuracies with respect to the GPS data. Firstly, the camera sometimes moves vertically, even though no height information is stored. Secondly, the position of the



(A) Upward camera movement. (B) Downward camera movement. (C) Minimal vertical camera movement.

FIGURE 4.4: The optical flow for several frames showing the different types of vertical camera movement. The rectangles mark the windows that are used to filter frames based on optical flow values. The difference in optical flow in these windows is clearly visible for the three cases. Note that these visualizations use higher saturation for clarity.

train is tracked accurately but can contain local errors. These issues were rarely present in the data we used to construct our pipeline and test our algorithms and mostly became apparent by examining the evaluation data. Therefore, we did not deal with them before. They do however cause large problems for the reconstruction. Therefore we use some simple techniques to correct them filter them out.

Issues caused by inaccurate positional and rotational data for the camera are sometimes too large to correct using our pipeline. Fortunately, the amount of points we have is sufficient to filter out a large portion. Therefore, we eliminate frames where the camera is moving vertically by examining the optical flow. We take a window within the frame that corresponds to the area around the vanishing point of the tracks. Since points in this area are mostly very distant in this area, the average flow for this window should be close to zero. When vertical camera movement occurs, this is clearly reflected by much larger flow in this window. Examples are shown in Figure 4.4. We find that discarding frames with average flow larger than 1 pixel in the window $1000 \leq x \leq 1100 \wedge 220 \leq y \leq 300$ removes all frames with undesired vertical motion. To filter out errors in GPS data, we make use of the knowledge that pictures are taken approximately every 50cm. By examining some incorrect reconstructions, we find that most errors occur because the displacement between two frames is too small. We therefore discard frames that correspond to a displacement of less than 30cm or more than 60cm. Together, these frame filters remove 74% of all frames. The remaining frames still contain much overlap and therefore appear sufficient to create a complete reconstruction.

After triangulation, the ground level of point clouds reconstructed from different frames can vary as a result of small variations in the height of the camera. To correct this,

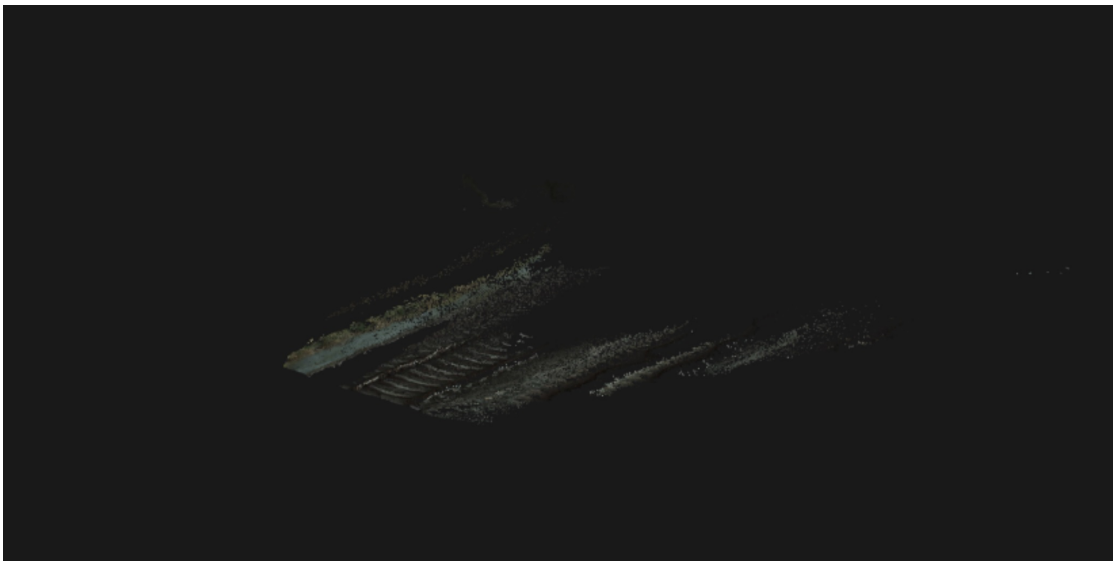


FIGURE 4.5: A result of our simple ground plane detection. Though not all ground points are selected, importantly, the selected points are all part of the ground.

we note that in all point clouds, most points are part of the ground. Additionally, these ground points lie more or less on a horizontal plane (xz-plane). Therefore, the point with median y value must be a ground point. We find this median $\tilde{y}$ and perform windowing on the y value of all points. For points within this window, we again take the median ($\tilde{y}_2$). Finally, we translate the point cloud in y direction by $-\widetilde{y}$, such that the ground plane approximately corresponds with $y = 0$. We find that a window of $(\widetilde{y} - 0.35) \leq y \leq (\widetilde{y} + 0.15)$ generally selects a large portion of ground points (see Figure 4.5). We found that for point clouds with varying different ground levels, the resulting translation decreases the height difference. Our simple approach runs very quickly and aligns point clouds to a global reference plane $y = 0$. Because we use a global reference, as opposed to aligning point clouds locally, the ground level does not drift.

4.2.3 Parameter Settings

Having discussed our parameter settings for the optical flow computation, as well as a number of methods we use to filter inconsistent data and improve the results, we will now go through the parameters we have not discussed yet and motivate the settings we use.

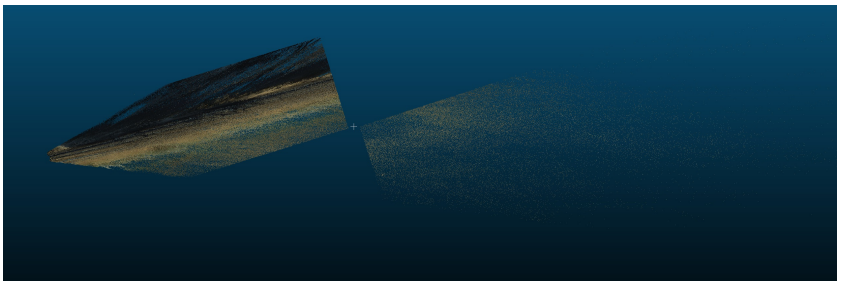


FIGURE 4.6: The sections we used to determine point cloud density. We considered the segment on the left to contain little noise, while the segment on the right contains almost only noise.

For the statistical outlier removal step, we used a σ multiplier of 1.0 and a neighbour count of 30. This means that any point with an average distance to its 30 nearest neighbours of more than the average over all points plus one standard deviation is filtered out. Assuming a normal distribution, this will remove approximately 15.9% of all points. This seems like a reasonable number, given the large amount of points we have. A neighbour count of 30 gives us a good representation of distance, while also maintaining good performance.

For the radius outlier removal, we examined some segments of the point cloud resulting from the baseline algorithm. Looking at one area of the point cloud that contained a lot of noise, we found that points in this area had at most 10 neighbours within a radius of 0.25m. with an average of around 1.5. Taking a segment with little noise, we found that points in this area had around 4.5 neighbours within a radius of 0.25m. Therefore, we selected a minimum number of neighbours of 3.0, based on the average of these values. The segments we used are shown in Figure 4.6.

For RMLS, we examined the mesh resulting from the baseline algorithm to determine the area that should give a good approximation of the surface around a point. We found that a radius of 0.5m around a point would generally give a clear representation of its surrounding surface. Therefore, we used 0.5m as the search radius.

With Greedy Projection Triangulation, we want to achieve high completeness in the final mesh. Therefore, we use a high search radius and μ, and allow for a lot of variation in surface and corner angles. This forces us to use a high number of nearest neighbours, as the implementation generates warnings and poor results (containing a lot of gaps or faces that cross each other) with low numbers. At 2000, warnings are still generated,

but no significant issues seem to be present in the final result. Increasing this number also increases the amount of time required. Therefore, we limited it at 2000.

For the Marching Cubes algorithm, we set the grid resolution to 200 in each dimension. According to observations about the level of detail in the point cloud and its total size, this should be enough to maintain accuracy.

Using the same approach, we determined the resolution for Grid Projection (0.001). Using a padding size of 3 allows for small holes to be filled. A maximum binary search level seemed to provide a good balance between accuracy and performance.

For Poisson surface reconstruction, we performed qualitative evaluations using a point cloud based on the area around Groningen. We found that a depth of 10 was enough to preserve details in areas with small structures, while a minimum depth of 6 ensured that smooth surfaces did not consist of large numbers of triangles while only a few are necessary. The effect of different degree, point weight and samples per node parameters was not very clear based on experiments. Therefore, we used recommendations from the PCL documentation for these parameters. The scale mostly impacted the size of certain structures in the resulting mesh. A value of 1.1 provided good results. Increasing the value would generate large and inflated objects, while decreasing the value would reduce the size of thin structures.

4.3 Quantitative Evaluation

We evaluate the speed and accuracy of the different variations quantitatively. The speed can simply be evaluated by running the implementations on the same machine under the same conditions and recording the running times.

We evaluate all variations of the algorithm. This allows us to compare the quality of different techniques, and compose new variations based on the results. We then evaluate these new variations in the same manner as the initial ones. A visualization of the full evaluation pipeline can be found in Figure 4.8.

In Section 4.3.1 we will describe our ground truth point cloud data. We will then describe our global evaluation and axis evaluation in Section 4.3.2 and Section 4.3.3 respectively.

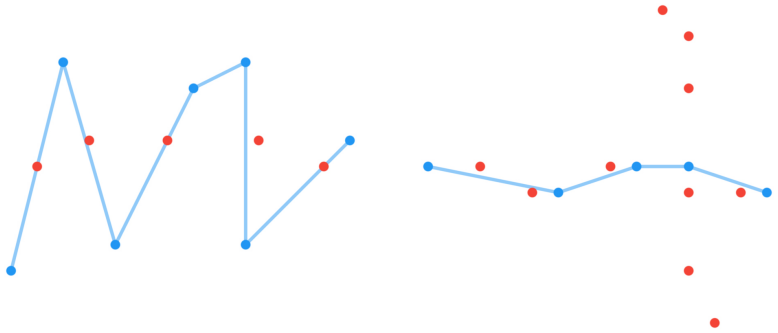


FIGURE 4.7: 2D example cases where distance measures in one direction are not sufficient. The point cloud is shown in red, while the mesh is shown in blue. Left: The forward distance fails. In this case, all the points lie near or on the surface of the mesh. However, the mesh matches the points very poorly. Right: The backward distance fails. The mesh corresponds very well with part of the point cloud, but a lot of points are not represented in the mesh.

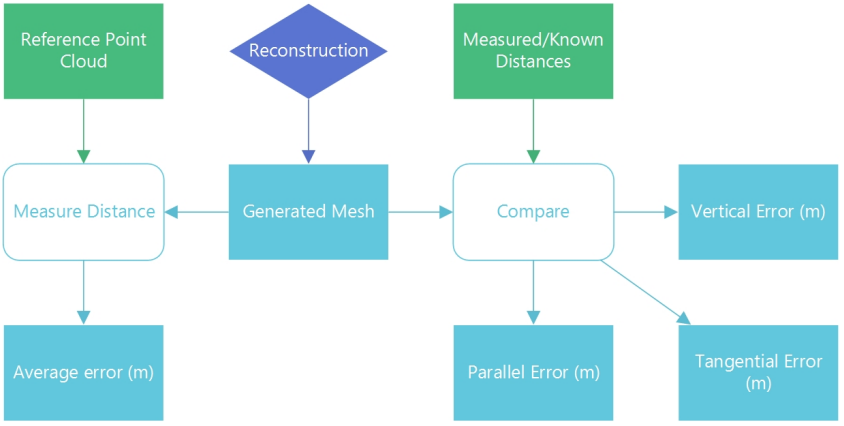


FIGURE 4.8: The evaluation pipeline. Input data is marked green.

4.3.1 Ground Truth Data

For our evaluation segment, a point cloud generated from laser scans are available. We use this point cloud as ground truth data in our quantitative evaluation. Since this point cloud covers a much larger railway section and uses some different coordinate conventions, we have to process it such that it is aligned with our evaluation segment and covers the same segment.

To align the ground truth point cloud with our reconstructions, we use a set of affine transformations and reflections. The composite of these transformations converts from the coordinate system of the ground truth data to the coordinate system of the reconstructions. Next, we align the ground to the $y = 0$ plane, as described in Section 4.2.2.

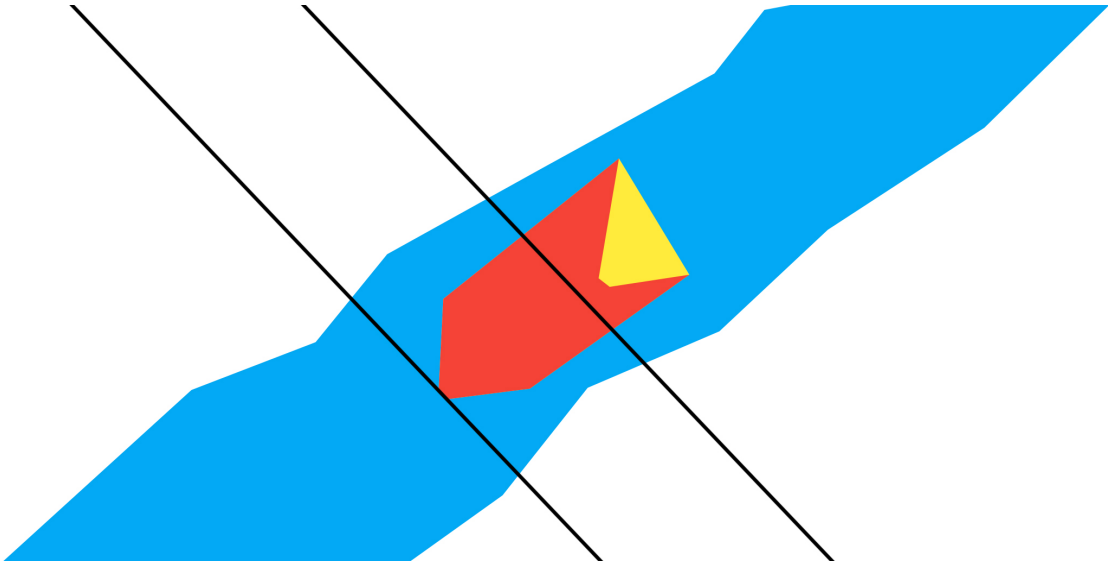


FIGURE 4.9: A schematic top-down view of the laser scan point cloud (blue) and reconstructed point cloud (red). The union of the red and yellow areas represents the convex hull of the reconstructed point cloud. The black lines indicate what the result of straight cuts would look like. It is clearly shown that with straight cuts, the symmetric difference of the ground truth and reconstruction would be large. The yellow area is in the ground truth, but not in the reconstruction. Since this area lies within the field of view of the camera, we still take it into account although it will likely increase the forward distance in our evaluation.

Since we wish to only evaluate regions that can realistically be reconstructed based on our input data, we can not simply make straight cuts at the first and last camera positions. This concept is shown in Figure 4.9. Instead, we have to take the field of view (frustum) of the camera into account and limit the maximum distance from the camera. This is not a straightforward process. We use the reconstructed intermediate point cloud from our baseline algorithm as a reference. We first apply the statistical filter described in Section 3.6.2 to the reconstructed point cloud in order to filter gross outliers. Next, we compute the convex hull of the filtered point cloud on the XZ-plane (ground plane). Finally, we remove all points from the laser scan point cloud that lie outside this convex hull based on their X and Z coordinates. This ensures that the ground truth point cloud matches the area of our reconstructions well such that we can generate meaningful evaluation results.

4.3.2 Global Accuracy Evaluation

We evaluate the global accuracy of the reconstructed mesh using 3D scans of the same area. Specifically, we compute the distances and standard deviations between a point

cloud generated from laser scans and the mesh generated from our reconstruction in order to determine the global accuracy. We compute both the forward (point cloud to mesh) and backward (mesh to point cloud) distances, since computing the distance in only one direction could paint a very biased picture, as shown in Figure 4.7. The forward distance is a better representation of completeness and can only be positively influenced by increasing the coverage of the generated mesh. The backward distance is a better representation of accuracy and can only be positively influenced by increasing the density of the reference point cloud. This approach is similar to the Hausdorff distance used by Aspert et al. [55]. However, we choose to report both the forward and backward distances separately as they carry additional information.

For point cloud C and mesh M, the forward distance is given by

$$d_f(C, M) = \sum_{\boldsymbol{p} \in C} \frac{d(\boldsymbol{p}, M)}{n} \tag{4.1}$$

where $\boldsymbol{p}$ is a point in the point cloud. The point to mesh distance $d(\boldsymbol{p}, M)$ is then given by

$$d(\boldsymbol{p}, M) = \min \{d(\boldsymbol{p}, t) | t \in M\} \tag{4.2}$$

where $d(\boldsymbol{p}, t)$ represents the shortest distance between point $\boldsymbol{p}$ and triangle t.

For the backward distance, we sample points on the mesh in order to obtain a uniform density. This is necessary as vertex density can vary across a mesh, depending on the surface reconstruction technique used and the local densities of the original point cloud. This would mean that dense regions would have a larger impact on the global average, which is not desired. Therefore, we randomly sample mesh M with 500 points per square meter of surface area [61]. This results in a point cloud C_M. The backward distance is then computed in similar fashion to the forward distance:

$$d_b(M, C) = \sum_{\boldsymbol{p} \in C_M} \frac{d(\boldsymbol{p}, C)}{n} \tag{4.3}$$

For each point $\boldsymbol{p} \in C_M$, we compute the shortest distance to the ground truth point cloud C_{gt} using

$$d(\boldsymbol{p}, C_{gt}) = \min \{d(\boldsymbol{p}, \boldsymbol{p_{gt}}) | \boldsymbol{p_{gt}} \in C_{gt}\} \tag{4.4}$$

To increase accuracy of the evaluation, C_{gt} is locally modeled using Delaunay triangulation [62] (in this paragraph triangulation refers to subdivision of a planar object into triangles) on a least-squares plane [57]. This is done by taking the nearest neighbours $\boldsymbol{p_{nn}}$ in C_{gt} of currently examined point $\boldsymbol{p}$ in C_M. A plane is fitted to all points in C_{gt} within a radius 0.5 of $\boldsymbol{p_{nn}}$. Based on the density of our point cloud, this radius is enough to ensure enough points are included. These points are then projected onto the plane. Delaunay triangulation is applied to the projected points. The resulting triangulation is then translated back to the original points, resulting in local mesh M_l. $d(\boldsymbol{p}, \boldsymbol{p_{nn}})$ is then computed as the shortest distance between $\boldsymbol{p}$ and M_l.

In order to decide which variation performs best, we use the following formula for the global distance:

$$d_{global} = \sqrt{\frac{d_f^2 + d_b^2}{2}} \tag{4.5}$$

Here we use the squared distances in order to punish bad scores in either the forward or backward distances more severely. We take the average, as both distances are equally important and we do allow trade-offs between the distances as long as the gain is high enough (hence we do not take the maximum of both distances). Finally, we take the square root to make the final outcome comparable to the actual distances in meters.

The distances and standard deviations can be easily computed and visualized using CloudCompare software [56, 57].

4.3.3 Axis Accuracy Evaluation

In order to determine accuracy in each main direction (parallel, perpendicular and vertical), we use a selection of known distances based on the rails and catenary constructions (CCs) and measure the errors compared to these distances based on random samples.

Description	Direction	Distance (cm)	Count
Distance between rails	Perpendicular	14.35	3
Distance from track to top of CC	Vertical	812.00	2
Distance between consecutive CCs	Parallel	6490.00	1

TABLE 4.1: The measurements used to evaluate accuracy in each direction separately.

FIGURE 4.10: An example frame of the train run we use to perform our evaluation for turns and passing trains.

These errors can be expressed in parallel, perpendicular and vertical components based on the selected measurements. The measurements we use are shown in Table 4.1, as well as the amount of times they occur in the evaluation data. We will also include the ground truth point cloud in this evaluation in order to validate its accuracy.

4.4 Additional Evaluation

Since we also wish to investigate the effect of turns and passing trains on the reconstruction qualitatively, we use another segment of 100 meters around Almelo that contains both (see Figure 4.10). We only reconstruct this segment using the best variation found from our main evaluation. Furthermore, we do not evaluate this reconstruction quantitatively, as no ground truth data is available. Instead, we discuss the observed results and the effects of turns and passing trains. Because the motion of the train is different from our initial evaluation, the frame filtering based on optical flow (Section 4.2.2) does

not work as intended and removes all frames. To be able to generate a reconstruction, we set the average flow threshold to 2.0 instead of 1.0. This way, a sufficient amount of frames (17%) is preserved.

Chapter 5

Results

In this chapter, we will show our results in terms of speed and accuracy based on a segment of 100m of railway (202 frames) near Holten. We show results for each of our variations described in Chapter 4. Images of the results of our quantitative evaluation can be seen in Appendix B. This includes visualizations of the errors and colored meshes where available (all variations that use Greedy Projection Triangulation). Results from our additional evaluation with turns and passing trains are shown in Appendix C. The running times of the variations are shown in Section 5.1. The global accuracy and axis accuracy are shown in Section 5.2 and Section 5.3 respectively.

5.1 Running Times

Variation	Opt. flow	Triang.	Refinement	Surf. Rec.	Total	FPS
baseline	2442	91	0	3062	5594	0.04
t-poly	2442	94	0	3013	5549	0.04
p-stat	2442	91	17	3222	5772	0.04
p-radius	2442	91	54	2852	5425	0.04
p-rmls	2442	91	1411	1982	5925	0.03
s-mc	2442	91	0	80	2613	0.08
s-grid	2442	91	17	4063	6612	0.03
s-poisson	2442	91	17	44	2594	0.08
additional	2438	80	0	4890	7408	0.03

TABLE 5.1: The running times for each step and the total running times for each variation. All times are in seconds.

To evaluate our speed requirement of 1.3 frames per second, we measured the running time of each variation. The results are shown in Table 5.1. A breakdown of the time taken by each step of the pipeline is shown separately. It should be noted that these running times can vary slightly between runs of the same implementation. Therefore, we ran each variation four times and took the average time for each step. Running times for each step were quite consistent, and therefore the averages are a good indication of the differences between variations. As can be seen in the results, none of the variations met the speed requirement.

A few things can immediately be noted. Surface reconstructions for s-mc and s-poisson are much faster than the other methods. Although it should be noted that without the statistical outlier removal step, Poisson surface reconstruction does not produce results. p-radius performs faster than the other post-processing variations due to its faster surface reconstruction. p-rmls has faster surface reconstruction than other variations that include greedy projection reconstruction, but uses a lot of extra time in the refinement step.

In general, the optical flow and surface reconstruction steps require the most time by far (with some exceptions in the surface reconstruction step). Optical flow is computed with an almost constant 12 seconds per frame, while surface reconstruction depends entirely on the input and technique used.

Although the extra evaluation is based on the same implementation as the baseline (with the exception of one parameter for the frame filtering based on optical flow), it has a much longer running time.

5.2 Global Accuracy

Table 5.2 shows the results of the global accuracy evaluation. Here, our requirement was a distance of less than 10cm. The baseline and t-poly variations produce very similar results. All post-processing steps increase the forward distance and decrease the backward distance. s-mc produces a low forward distance, but an extremely high backward distance. Based on the results of s-mc, we can not be certain about the validity of the implementation. Therefore, these results should not be taken to heart. s-grid and s-poisson produce worse results than the baseline algorithm. Arguably though, they

Variation	d_f	d_b	d_{global}
baseline	1.91 (3.28)	1.28 (1.94)	1.63
t-poly	1.91 (3.28)	1.22 (1.46)	1.60
p-stat	2.33 (4.01)	0.98 (0.72)	1.79
p-radius	2.61 (4.36)	0.71 (0.55)	1.91
p-rmls	2.06 (3.40)	1.17 (0.97)	1.68
s-mc	0.95 (1.58)	405.96 (307.66)	287.06
s-grid	2.59 (3.93)	1.29 (0.78)	2.05
s-poisson	3.18 (5.33)	1.81 (1.95)	2.59

TABLE 5.2: The averages and standard deviations (between parenthesis) for backward ((d_b) and forward (d_f) distances, and the global distances for each variation in meters.

should be compared p-stat, as they also use statistical outlier removal. In this case they also perform worse in both metrics. When looking at the global distance, the t-poly variation performs best, with a score of 1.60.

5.3 Axis Accuracy

Variation	Perpendicular error (m)	Vertical error (m)	Parallel error (m)
baseline	-0.43	-5.04	1.19
t-poly	-0.40	-5.73	1.31
p-stat	-0.26	-5.02	1.19
p-radius	-0.37	-	0.70
p-rmls	-0.20	-5.04	1.29
s-mc	-	-	-
s-grid	-	-	-
s-poisson	-	-2.05	-
Ground truth	-0.08	0.03	-0.17

TABLE 5.3: The average errors in each main direction compared to known distances and measurements. The ground truth point cloud is included as well. Negative values indicate that the distance in the mesh was smaller than the actual distance. Dashes indicate that the distance could not be measured.

We require a maximum error of 1m in the parallel direction and 10cm in the perpendicular and vertical directions. The results from our axis accuracy evaluation are shown in Table 5.3. From the results, it was difficult to measure these distances due to a lack of color and misshapen objects. In some cases, we could still make a reasonable estimate of the locations of CCs based on color information (shadows) and taking the average position when a CC appeared stretched out over a wider area. This is illustrated in Figure 5.1. In order to measure the height of CCs, we selected the five highest point within a distance of one meter from the railway tracks in the horizontal plane and took

FIGURE 5.1: A view of the baseline mesh. Here, a CC is stretched out over a wider area (marked with blue). By taking averages and looking at clues from the surrounding environment, such as shadows (marked with red), we were able to produce an estimate of its location.

FIGURE 5.2: A close-up of the rails. The rails are recognizable by their gray color and by the shadows they cast on ground.

the average height of these points. The height of the rails, as well as the distance between rails, was much more consistent and generally clearly visible. This is shown in Figure 5.2.

Overall, these results should not be interpreted as a perfect representation of the actual errors, and some errors were impossible to measure in any reasonable manner. For completeness however, we did include the results. We also included the ground truth point cloud in this evaluation in order to validate its accuracy compared to the real-world situation based on the measurements and known distances. Due to easily recognizable objects, the measurements for the ground truth point cloud were much easier to perform and are therefore much more reliable.

5.4 New Variations

Based on our results, we selected four new variations to test. These variations appear promising based on our previous results and are described in Section 5.4.1. Since the polynomial triangulation method proved superior to the midpoint method, we use the polynomial method in all these new variations. The results of these new variations are shown in Section 5.4.2.

5.4.1 New Variation Descriptions

The s-poisson variation shows interesting results and smooth surfaces. However, in many areas, the mesh contains little detail and large blobs. This could be related to the minimum depth we selected. In an attempt to increase the level of detail, we set the minimum depth to 8 (previously 6) and the maximum depth to 11 (previously 10). This variation will be called **s-poisson-depth**. Like the s-poisson variation, we also use statistical outlier removal.

For the p-rmls variation, results look smoother than the baseline. However, not every area is smoothed, and the forward distance is larger than the baseline. To explore several variations, and gain insights into the effect of the search radius parameters on our reconstruction, we perform evaluations with a search radius of 0.25 (**p-rmls-small**) and 1.0 (**p-rmls-large**).

For the statistical outlier removal used in p-stat, we are interested in the effect of different σ multipliers. With the current setting of 1.0, it appears that too many points are removed, as the backward distance is increased by a sizable amount. Therefore, we test a σ multiplier of 1.5, in order to remove less points. This variation will be called **p-stat-large**.

5.4.2 New Variation Results

Variation	Opt. flow	Triang.	Refinement	Surf. Rec.	Total	FPS
p-stat-large	2442	94	17	2961	5514	0.04
p-rmls-small	2442	94	312	1900	4748	0.04
p-rmls-large	2442	94	6871	1824	11231	0.02
s-poisson-depth	2442	94	16	561	3113	0.07

TABLE 5.4: The running times for each step and the total running times for each of the new variations. All times are in seconds.

Table 5.4 shows the running times for the new variations. We can see that the p-rmls-large variation took the most time out of all variations, due to a large increase in the refinement step. p-rmls-small is faster than p-rmls. s-poisson-depth is slightly slower than s-poisson, but still faster than most variations. p-stat-large is comparable in speed to p-stat.

Variation	d_f	d_b	d_{global}
p-stat-large	2.11 (3.73)	1.07 (0.80)	1.67
p-rmls-small	2.28 (3.82)	0.86 (0.62)	1.72
p-rmls-large	2.04 (3.24)	1.44 (1.69)	1.76
s-poisson-depth	2.16 (3.68)	1.68 (1.62)	1.94

TABLE 5.5: The averages and standard deviations (between parenthesis) for backward ((d_b) and forward (d_f) distances, and the global distances for each of the new variations in meters.

The results of the global accuracy evaluation are shown in Table 5.5. Compared to p-stat, p-stat-large produces a slight improvement in the global distance, with a smaller forward distance and a slightly higher backward distance. Both p-rmls-small and p-rmls-large have a higher global distance than p-rmls. For p-rmls-small, this is due to a larger forward distance, while p-rmls-large has a larger backward distance. s-poisson-depth generates improvements over s-poisson for all distances.

Results from the axis accuracy evaluations (shown in Table 5.6) for the new variations were similar to the original variations. p-stat-large showed no significant differences with

Variation	Perpendicular error (m)	Vertical error (m)	Parallel error (m)
p-stat-large	-0.25	-4.98	1.24
p-rmls-small	-0.24	-5.01	1.14
p-rmls-large	-0.24	-4.98	0.90
s-poisson-depth	-	-5.20	-

TABLE 5.6: The average errors in each main direction compared to known distances and measurements. The ground truth point cloud is included as well. Negative values indicate that the distance in the mesh was smaller than the actual distance. Dashes indicate that the distance could not be measured.

p-stat. Neither did p-rmls-small and p-rmls-large compared to p-rmls. For s-poisson-depth, most parts of the mesh, including CCs, became smaller compared to s-poisson. Therefore, the absolute vertical error became larger.

Chapter 6

Discussion

We will now discuss the results shown in the previous chapter. First, we will discuss every variation separately in Section 6.1. We will then draw our general conclusions in Section 6.2. In Section 6.3, we will discuss our reconstruction and evaluation methods and possible shortcomings. In Section 6.4 we will answer our research questions. Finally, we will list some areas that seem promising for future work for both the technical side and the physical setup used to record the data in Section 6.5.

6.1 Individual Variations

In this section we will look at each variation individually and look at their results in both a quantitative and qualitative manner.

6.1.1 Baseline

The baseline algorithm provides a quite solid result that can be used to compare other variations against. It actually produces the smallest forward distance (although only barely smaller than t-poly) and a backward distance around the median of all variations. As visible in Figure B.1, the mesh contains quite a lot of noise. This can easily be explained by the noise generated in the optical flow and triangulation steps, combined with greedy projection triangulation, which attempts to use every point in its original location. Although the general structure of the environment is reflected pretty well,

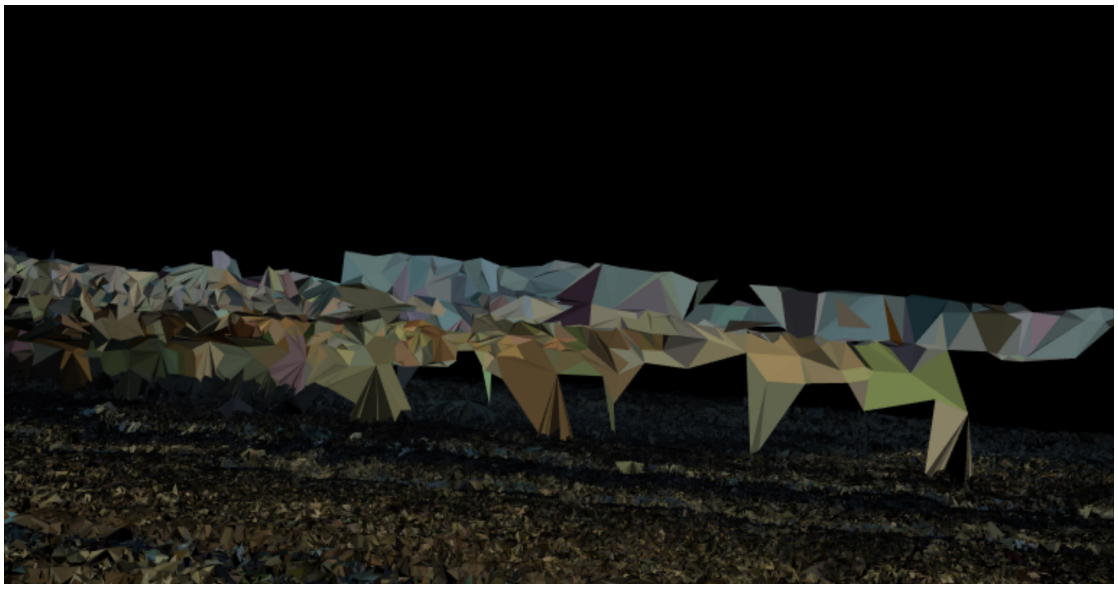


FIGURE 6.1: An example of a poorly reconstructed CC from the baseline algorithm. As shown in the image, the CC appears distorted and is spread out over a large distance parallel to the railway.

details and thinner structures, such as signs and CCs produce poor or no results (see Figure 6.1). This is highlighted by the fact that the forward distance is larger than most areas in the CCs and power cables, as visible in Figure B.3. The baseline algorithm performs around the median in terms of time.

6.1.2 Polynomial triangulation (t-poly)

Polynomial triangulation produces results very similar to the midpoint method used in the baseline algorithm. The forward distance is nearly exactly the same as the baseline, while the backward distance is slightly better. Visually, the results are almost indistinguishable. The running time for the polynomial method was slightly lower than that of the baseline. The triangulation is slightly slower, while the surface reconstruction step is slightly faster, although this could be coincidental. Due to better triangulation, less noise might be present, explaining the faster surface reconstruction. To confirm this assumption, we applied statistical outlier removal to the point cloud after polynomial triangulation to compare the amount of points removed. Indeed, less points were removed with polynomial triangulation compared to midpoint triangulation. After polynomial triangulation, 43966 points (1.3%) were removed, whereas after midpoint triangulation, 46156 points (1.4%) were removed.

6.1.3 Statistical Outlier Removal (p-stat)

Statistical outlier removal removes 1.4% of points and clearly increases the forward distance and decreases the backward distance. This is consistent with what one would expect from an outlier removal technique. The fact that the backward distance decreases indicates that the majority of the removed points indeed consisted of outliers that had a negative effect on the overall accuracy. Since this technique only removes points, the completeness, and thus the forward distance, will very likely increase. However, in an optimal scenario, we would expect the forward distance to remain similar to the unfiltered results. The running time of this variation is comparable to the baseline and it takes longer to perform surface reconstruction. This is counter-intuitive, as we would expect the running time to decrease with less points. A possible explanation could be an increased number of gaps between points, making surface reconstruction more difficult.

6.1.4 Radius Outlier Removal (p-radius)

The results from the radius outlier removal are very similar to the statistical outlier removal, but are slightly more skewed in favor of the backward distance. This is expected, as more points are removed (7.9%). Overall, the radius outlier removal performed very similarly to the statistical outlier removal. This is logical, as both have the same general goal. The main difference is that the statistical outlier removal does not require much tweaking of parameters based on the input, while for radius outlier removal, we have to examine the density of the point cloud in various locations in order to make an informed decision.

6.1.5 Robust Moving Least Squares (p-rmls)

Based on quantitative evaluation, RMLS produces results comparable to the baseline algorithm. The backward distance is slightly lower while the forward distance is slightly higher. This is consistent with outlier removal techniques. RMLS removes 3.1% of points and moves points to create a smoother surface. If this matches the ground truth surface well, accuracy will be increased because points will lie closer to this surface. In addition to quantitative results, the qualitative effects are visible in Figure B.17. The surface looks much smoother in areas that contained noise (such as the bottom right side) in

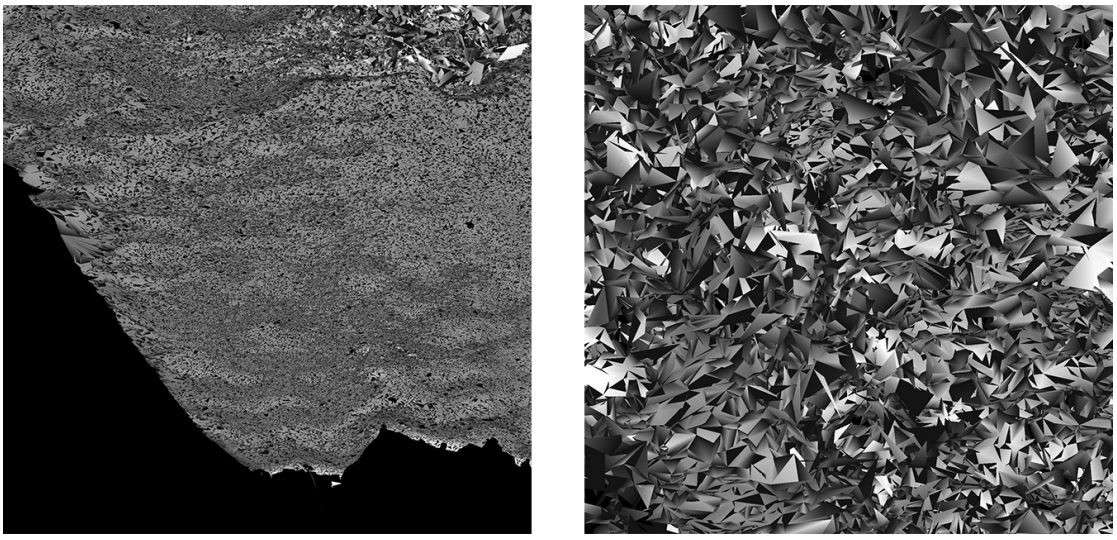


FIGURE 6.2: Close-ups of an area that was smoothed by the RMLS method (left) and an area that remained noisy (right). The left image was taken near the starting point of the camera. In this area there are fewer points because not many point clouds from separate frames overlap. The right image was taken further away from the camera trail. In this area, point clouds overlap and generally more noise is present.

the baseline result. In areas that contained a lot of noise (left side), the impact is minimal however. These areas generally lie further away from the camera trail. Most likely, the distance between points and the surface they should represent in these areas is larger than the search radius, and therefore surface fitting produces inconsistent results. A comparison is shown in Figure 6.2. The RMLS step took a large portion of the running time, but reduced the surface reconstruction time significantly. The entire algorithm ended up being slower than the baseline. It should be noted that the running time of RMLS depends greatly on the search radius parameter used, which becomes apparent from the results of p-rmls-small and p-rmls-large.

6.1.6 Marching Cubes (s-mc)

The results from the Marching Cubes surface reconstruction were very unexpected and might be a result of incorrect implementation or unsuitable parameters. Therefore, they should not be used as an indication about the properties or quality of Marching Cubes. We do report them, as they were part of our research setup. The reconstructed mesh is much larger and denser (in terms of faces and vertices) than any of the other meshes. The shape of the environment is barely recognizable.

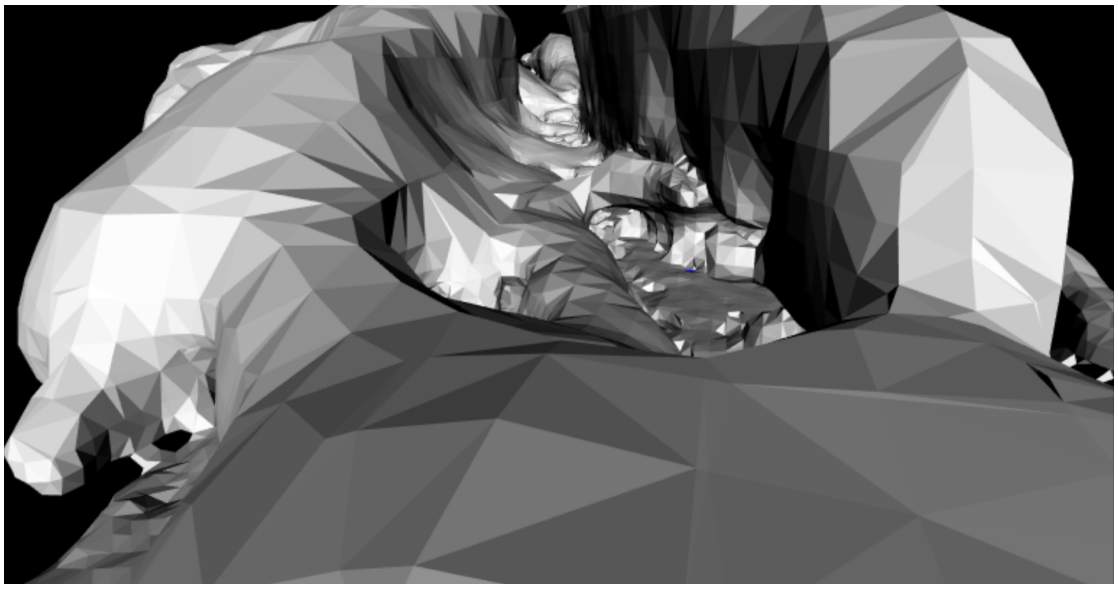


FIGURE 6.3: A close-up of the s-poisson mesh, viewed from a forward-facing camera. Some parts of the environment appear fairly detailed, while other parts appear inflated and larger than they should be. Additionally, tunnel shapes are created near the rails (center of the image).

6.1.7 Grid Projection (s-grid)

Grid Projection produces a backward distance similar to the baseline while increasing the forward distance. Although some parts of the mesh are smoother, other parts contain large irregularities (see Figure B.24). Like the variations that use Greedy Projection Reconstruction, Grid Projection does not reconstruct CCs very well, creating larger forward distances when compared to the ground truth (see Figure B.26). The technique appears to deal with noise well in some cases, while it responds poorly in other cases. A clear link between the amount of noise and the results of the technique is not directly visible. The poor results in some areas might simply be cases of 'bad luck' in terms of space partitioning. This surface reconstruction method took the largest amount of time.

6.1.8 Poisson Surface Reconstruction (s-poisson)

Poisson surface reconstruction produces a very smooth surface. However, it appears to sacrifice a lot of details in order to achieve this. A close-up is shown in Figure 6.3. In Figure B.27, we can see that objects, height variations or noise generate large protuberances. As a result, it is difficult to compare the mesh to the actual environment or ground truth data. The forward and backward distances are both higher than the

baseline. Overall, the results of this variation appear poor, but we hypothesize that this method would work better with a more accurate input point cloud. This surface reconstruction method was very fast after applying the statistical outlier removal.

6.1.9 Statistical Outlier Removal 1.5 (p-stat-large)

As expected, using a larger σ multiplier (1.5) caused less points to be removed (0.55%). This resulted in a decrease of 0.22 in the forward distance and an increase of 0.90 in the backward distance compared to p-stat. The global distance was therefore reduced by 0.12. Due to faster surface reconstruction, p-stat-large outperformed p-stat in terms of speed, although not by much. This is a quite unexpected result, as p-stat-large contains more points. A possible explanation for this could be that less holes are present in p-stat-large, potentially making it easier for the surface reconstruction to find connections between points. Out of all outlier removal variations, p-stat-large performed the best in terms of accuracy, and was only slightly slower than p-radius. Relative to all variations, p-stat-large performed quite well. This leads us to believe that 1.5 is a good σ multiplier for our application.

6.1.10 Robust Moving Least Squares 0.25 (p-rmls-small)

p-rmls-small performed slightly worse overall compared to p-rmls. The backward distance was reduces by a decent amount, while the forward distance was increased. Looking at the mesh, it spans a smaller area than p-rmls. This explains the larger forward distance and likely also causes the smaller backward distance, as most of the missing area contains a lot of noise. Still, a lot of noise is visible, even in the less noisy areas. This RMLS variation did not seem to perform much smoothing. The small search radius likely makes it difficult for the algorithm to fit surfaces around points. This is comparable to the issues p-rmls has with noisy areas. The running time was improved, as mostly the refinement step was faster. Due to the smaller search radius, less points are required to fit curves. This results in less time required per point.

6.1.11 Robust Moving Least Squares 1.0 (p-rmls-large)

Somewhat expectedly, p-rmls-large swayed results in the opposite direction compared to p-rmls-small. The forward distance was decreased slightly compared to p-rmls and the backward distance was increased. The global distance was larger than p-rmls and p-rmls-small. Qualitatively however, improvements in very noisy areas were visible. Even in most of these areas, smoothing is visible to some degree. Therefore, p-rmls-large could be a good alternative for applications that do not require high accuracy, but do have to look good. Looking at the running time however, we see a drastic increase. The refinement step takes almost two hours. Therefore, this variation can not be considered very practical in most cases.

6.1.12 Poisson Surface Reconstruction 8–11 (s-poisson-depth)

Compared to s-poisson, s-poisson-depth covered a larger area in the horizontal plane. At the same time, most of the large blobs in the mesh became smaller. A comparison is shown in Figure 6.4. This led to a small forward distance and backward distance. More of the ground truth was covered, while sections that matched poorly (blobs) became smaller. This caused an overall better result compared to s-poisson. The mesh also looks smoother in some areas, because more faces are used. However, many objects were still poorly reconstructed, and the qualitative issues with s-poisson were largely unsolved. The running time was increased, as the surface reconstruction took longer. This is expected, as the mesh contains more faces and is more detailed.

6.2 Conclusions

Looking at the results, it is immediately clear that our algorithms and implementations were not able to meet the speed and accuracy requirements. In this section we will discuss the results with respect to our requirements. We will examine the running times of our variations in Section 6.2.1, and the accuracy in Section 6.2.2 and Section 6.2.3. In Section 6.2.4, we will examine the results of our additional evaluation. We will then look back at our research questions and answer them in Section 6.4.

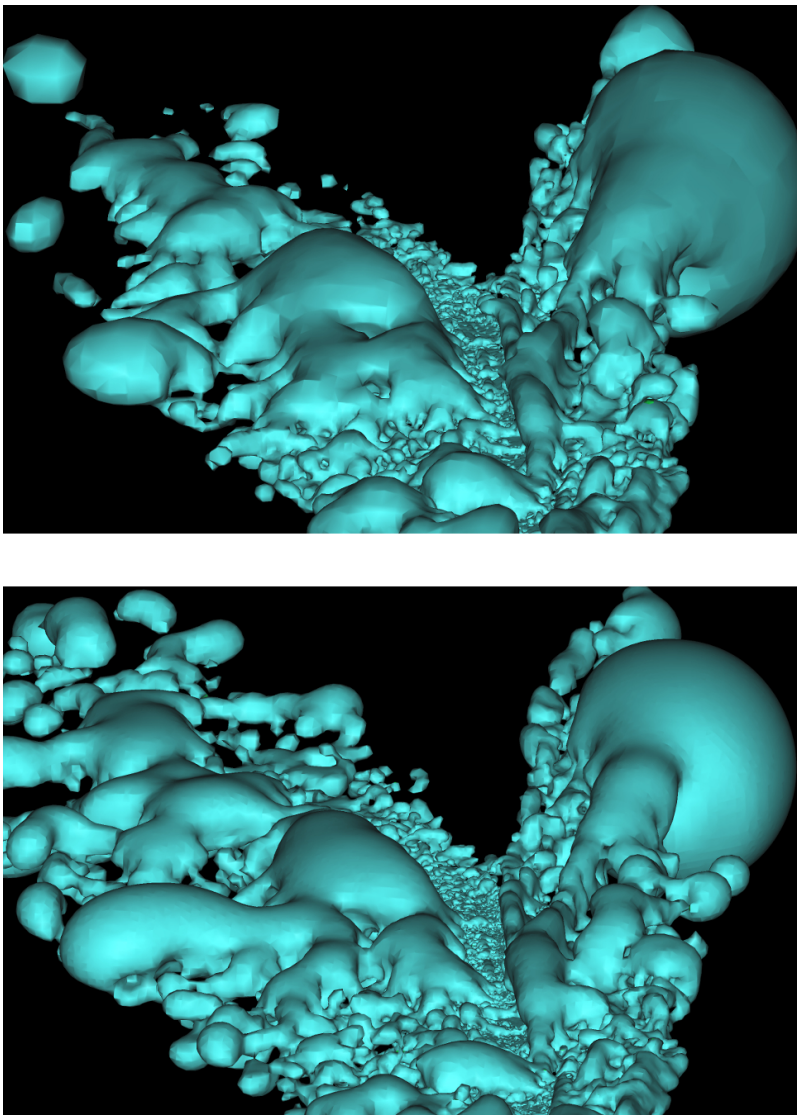


FIGURE 6.4: A comparison between s-poisson (top) and s-poisson-depth (bottom).

6.2.1 Running Times

The running times for our implementations ranged from 0.018 to 0.077 frames per second, while we required a running time of 1.3 frames per second. Optical flow and surface reconstruction steps generally took the most time. Post processing techniques also took long for some variations. Although this does say something about the speed of the techniques, one can imagine that a better parallelized implementation running on a better system could meet these requirements. Since the optical flow and triangulation steps only take two consecutive frames as input, these steps could be run for many frames at once. Refinement and surface reconstruction steps could also be parallelized to some degree, by stitching point clouds or meshes together at the end of the pipeline. This

would also result in more controllable memory usage, as only smaller portions of the point cloud would have to be in memory at one time and memory could be distributed among multiple machines.

6.2.2 Global Accuracy

In terms of accuracy, we can see that t-poly, p-stat, p-radius and p-rmls all improved the backward distance compared to the baseline, although none of these techniques met the requirements. Overall, t-poly performed best out of all the variations, with a more or less equal forward distance and a lower backward distance compared to the baseline.

Promising results are also shown by the initial post-processing variations (p-stat, p-radius and p-rmls). Each of these techniques appears to sacrifice completeness for accuracy. p-rmls does so most efficiently, with the lowest average forward and backward distance of the three. A comparison between the three techniques and the baseline is shown in Figure 6.5.

Since these techniques are dependant on parameters, we expect slightly better results to be achievable by tweaking these parameters. Indeed, this was achieved with p-stat-large. With a σ multiplier of 1.5, less points were removed compared to p-stat and p-radius. This resulted in a better trade-off between forward and backward distances. Even though, p-stat-large performed worse than the baseline and t-poly, this variation could be preferred when completeness is less important. By looking at the results, all outlier removal techniques remove almost exclusively noise. Therefore, we conclude they are working well. To increase the quality of results based on outlier removal, addressing the preceding steps in the pipeline would be a better choice than to focus on improving outlier removal.

p-rmls-small and p-rmls-large both perform worse than p-rmls, but do tell us something about the behaviour of the search radius parameter. Increasing this parameter increases the amount of smoothing, but does not improve the accuracy in our case. Decreasing this parameter decreases the amount of smoothing, but reduces the size of the resulting mesh. A comparison between the three settings is shown in Figure 6.6. Since p-rmls performed best out of all RMLS variations, 0.5 appears to be a good value for the search radius. Increasing this value, does however result in smoother surfaces, which could be

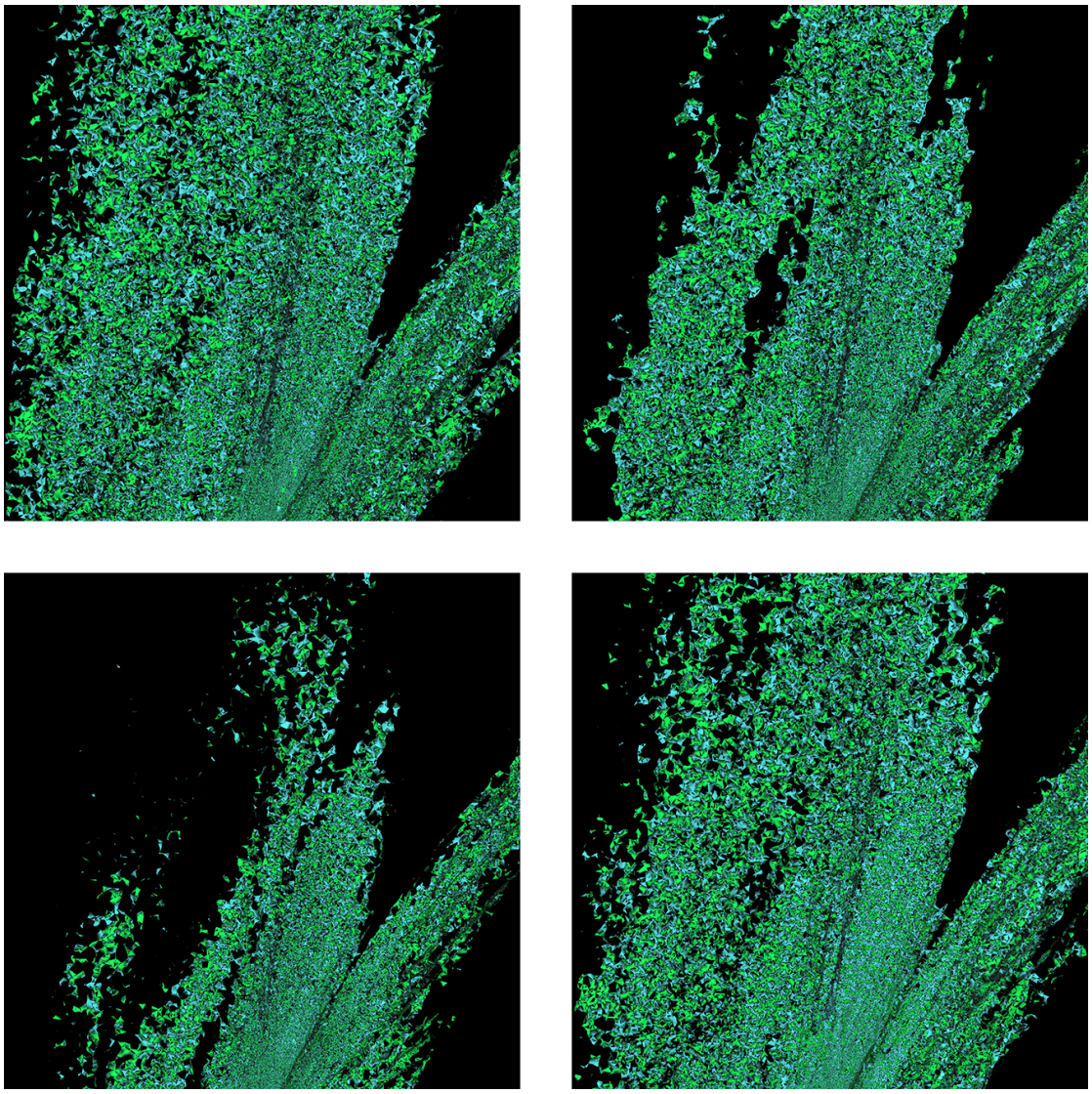


FIGURE 6.5: A side-by-side comparison of the baseline (top-left), p-stat (top-right), p-radius (bottom-left) and p-rmls (bottom-right). The image shows that p-radius removes the most points. In some areas, too much noise is present for p-rmls to generate smooth surfaces.

preferred in some cases. On the other hand, decreasing this value improves the running time.

For the surface reconstruction, Grid Projection and Poisson surface reconstruction seem viable options, although the baseline algorithm with greedy projection triangulation outperforms them. s-poisson-depth showed better results than s-poisson. Increasing the depth even further resulted in memory issues, with over 80GB of memory being used. Perhaps with a more efficient implementation and distributed computation, this could be circumvented.

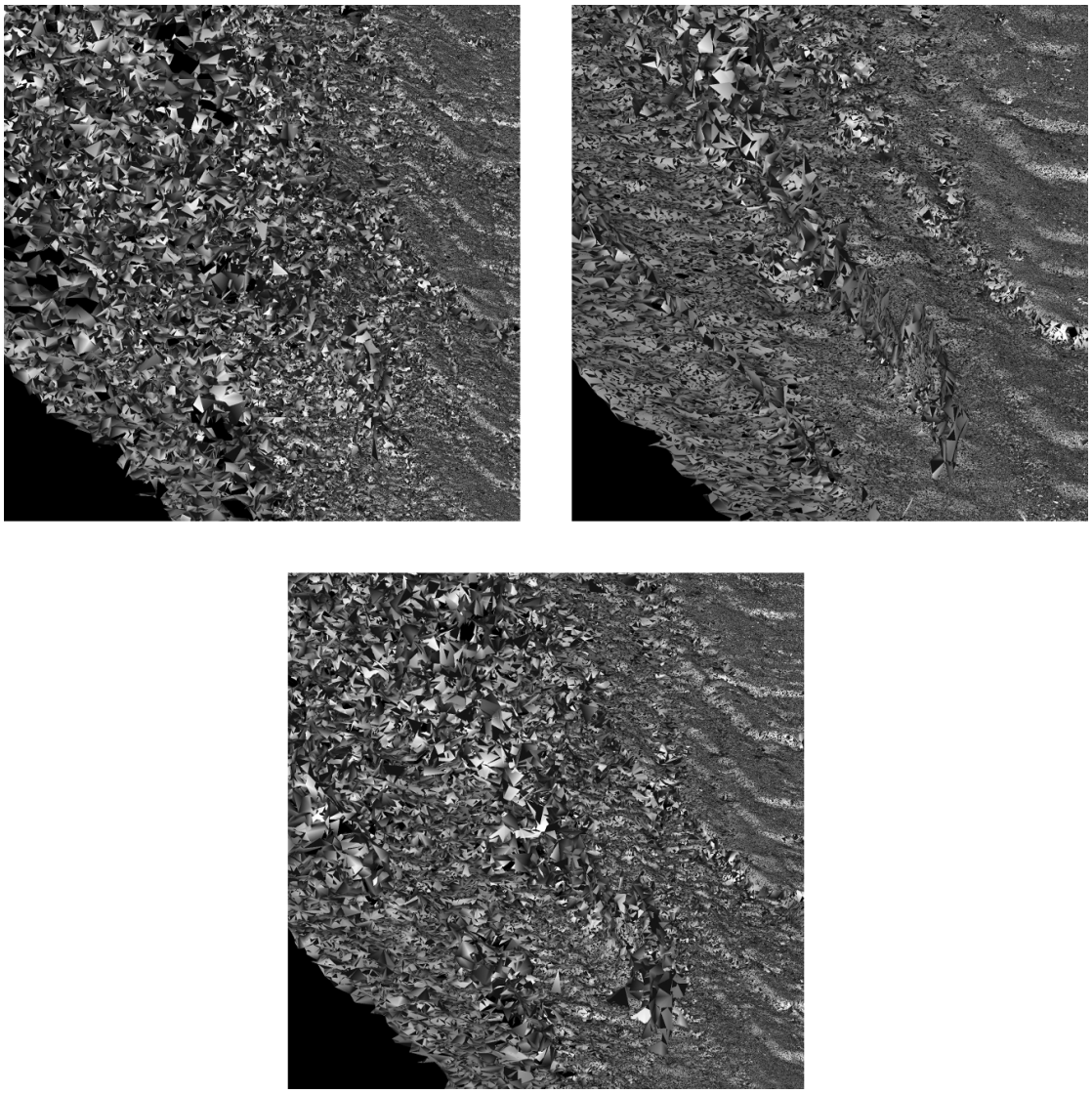


FIGURE 6.6: A comparison between the different RMLS variations. Top left: search radius 0.25. Top right: search radius 1.0. Bottom: search radius 0.5. Smoothness is increased with higher search radii.

With better results from earlier steps, all our surface reconstruction techniques except for Marching Cubes (which we can not say much about) appear suitable for our problem as they generate smoother surfaces when less noise is present. We tested this assumption by applying the surface reconstruction methods on the ground truth point cloud, which contained little noise. Greedy Projection Triangulation (Figure 6.7) showed detailed results, although the mesh did contain holes. Certain areas also appeared noisy. However, most details were preserved. The result of Grid Projection (Figure 6.8) contained less noise and overall looks good. Details are preserved, but sometimes objects look somewhat voxelated. Poisson surface reconstruction (Figure 6.9) showed many of the same issues as with the reconstructed point cloud, as large blobs appear around the mesh, and

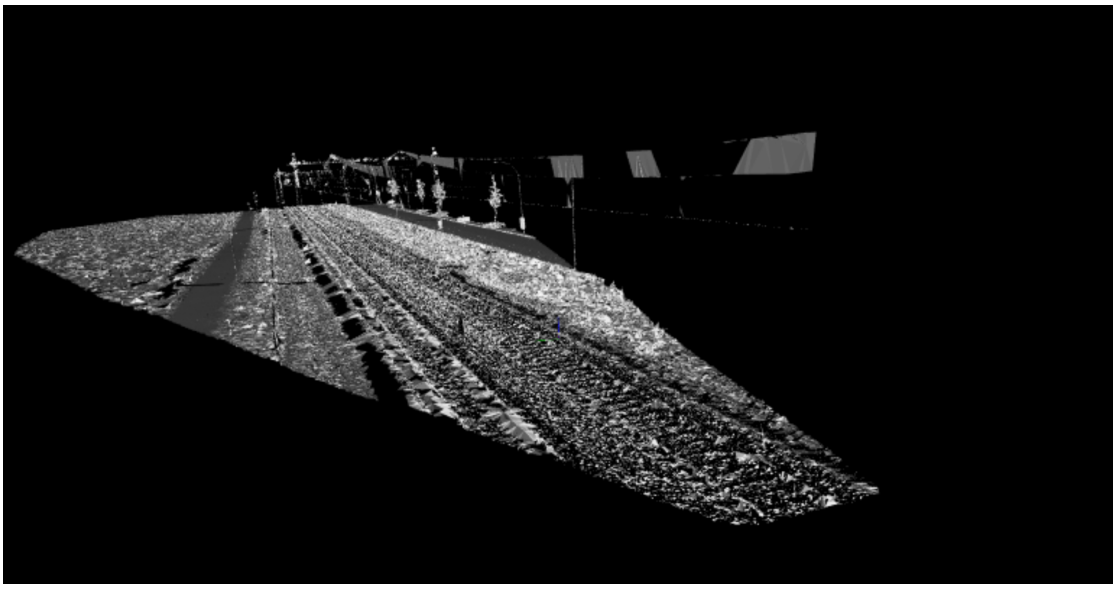


FIGURE 6.7: The mesh generated by applying Greedy Projection Triangulation to the ground truth point cloud.

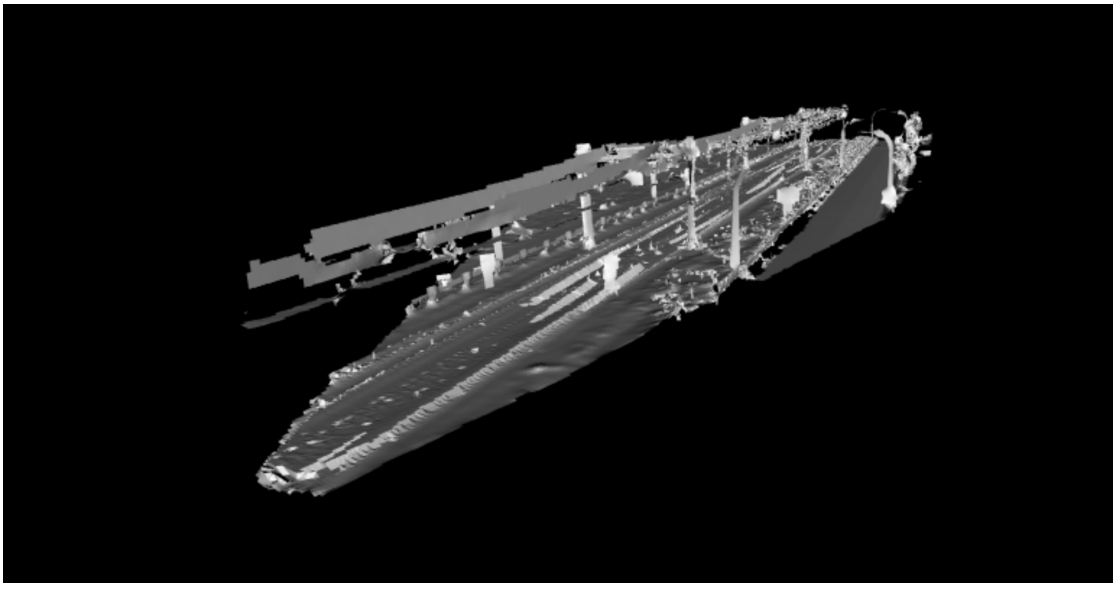


FIGURE 6.8: The mesh generated by applying Grid Projection to the ground truth point cloud.

the railway is transformed in some sort of tunnel construction. This leads us to believe that it has issues reconstructing thin structures such as CCs or signs unless the density of the point cloud is increased. Indeed, most of the examples shown by Kazhdan et al. [44] are based on dense point clouds, while smoothing is applied in less dense areas. This could explain the smooth but incorrect shapes we see in our mesh. One thing all surface reconstruction methods have issues with are the power cables. These relatively very thin cables are only represented by a small amount of points. They are either reconstructed

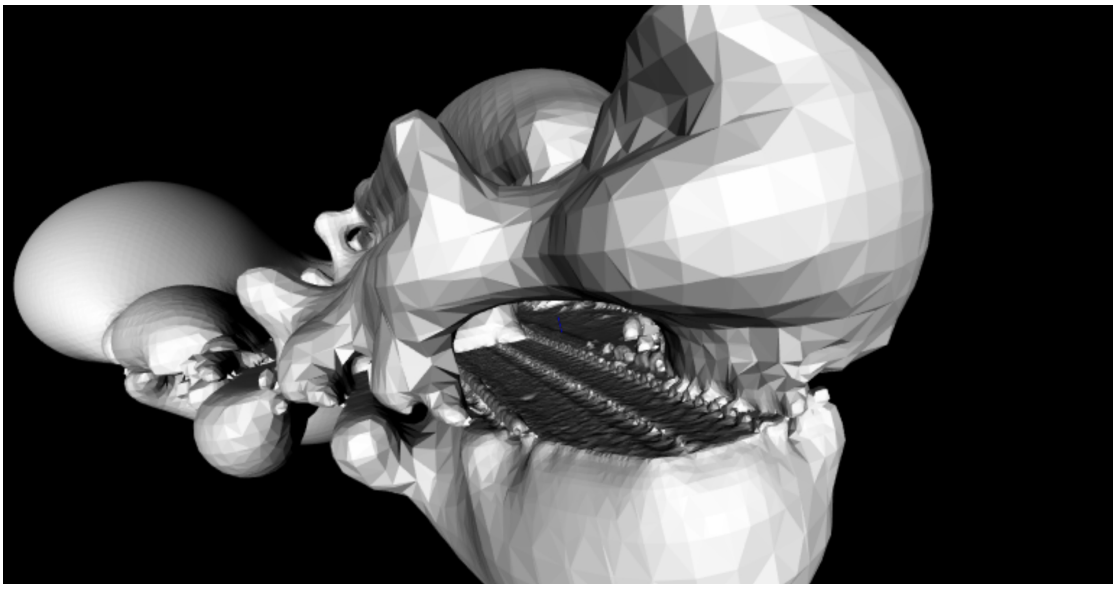


FIGURE 6.9: The mesh generated by applying Poisson surface reconstruction to the ground truth point cloud.

as planes connecting multiple cables, small disconnected faces, or large blobs.

6.2.3 Axis Accuracy

Based on the axis accuracy, it is difficult to compare the different variations. As mentioned, measuring the distances in the meshes was difficult and in some cases impossible. Therefore, the accuracy of this evaluation is questionable. However, there is a clear trend between the distances in each direction. Excluding the ground truth and taking the absolute values, every parallel error is larger than every perpendicular error, and every vertical error is larger than every parallel error.

We suspect that the vertical errors are large due to poor reconstructions of CCs. In many cases, large portions were missing, making them appear smaller in the reconstructions (see Figure 6.10). This is clearly related to the fact that the top of the CCs is only visible when the CC is far away from the camera. At this point, the CC is poorly visible and the optical flow is often small enough to filter it out. The top of the CC moves out of the view quickly and is only visible and nearby enough for a few frames.

The parallel error could also appear large due to poor localization of the CCs. In all cases, one CC was reconstructed multiple times in different locations after triangulation, resulting in a large blob after surface reconstruction (see Figure 6.1 for example). By

FIGURE 6.10: CCs in the baseline mesh highlighted in red. The green lines indicate their actual height. The portions above the railway are also mostly missing.

taking the center point of these blobs as our measurement anchors, we could still measure distances, but accuracy was lost. The perpendicular error was easiest to measure in most cases, as the rails were generally easy to locate.

The measurements were quite consistent within each mesh and between meshes. This leads us to believe that the reconstructions were overall smaller in the vertical and perpendicular direction than they should be. This could be due to inaccurate camera parameters, positional data or consistent errors in the optical flow. The former two options seem most plausible, as the optical flow technique used is not biased towards larger or smaller optical flow values. With inaccurate camera parameters, the camera angle could be underestimated, reducing distances. If the camera position or orientation is not accurate, objects could become distorted due to poor triangulation. In this case, certain distances might become smaller. Although errors in the optical flow were clearly visible, these mostly occurred around thin structures such as signs, while areas such as the ground and rails were mostly very accurate. When errors did occur in these regions, they consisted of small patches of outliers, which should not consistently affect the size of the reconstruction. Because distances in the parallel direction are heavily influenced by GPS, errors in this direction can be large locally, but are expected to even out over large distances.

Taking all the potential issues into account, we would expect the perpendicular and

vertical errors should be similar under good circumstances. For a front-facing camera, lines along these directions pass the camera in similar fashion, appearing near the vanishing point and moving towards the edge of the frame. However, issues with vertical camera movement could increase the vertical error. Additionally, the width of the frame is larger than the height, which causes vertically oriented objects to disappear partially from the field of view more quickly. The parallel error is mostly reliant on the accuracy of GPS data. Even with poor reconstructions, this error should be small if the GPS data is accurate, as long as accurate measurements are possible.

6.2.4 Additional Evaluation

The goal of the extra evaluation is to examine the effect of turns and passing trains on the reconstruction. Although we can not evaluate the accuracy of the extra evaluation, we can evaluate the results qualitatively and evaluate the running time.

As shown in Table 5.1, the reconstruction for this evaluation was much slower than the reconstruction for the baseline algorithm, even though the same techniques were used. This difference is caused exclusively by the surface reconstruction step, as the running times for the optical flow computations are nearly identical. We suspect that this is due to the high amount of noise that is present, which could provide difficulties for the greedy projection triangulation.

From the results, we can see that a lot of noise is present in the point cloud and mesh. This is visible in Figure C.3 (bottom) and Figure C.4 (bottom-left). Many points are located behind their corresponding camera location after triangulation. These points are not positioned correctly. Looking at these points, we see that they mostly do not match the color of the passing train (yellow). This leads us to believe that most of these erroneous points are caused by the motion of the train on which the camera is mounted. The motion related to a turn could potentially amplify small errors in the camera parameters or GPS. The triangulation step can not correct these errors and as a result will place points behind the camera. The errors mostly occur near the center of the frames, or in other areas with small optical flow values. In these cases, small errors can drastically change the location of a point in 3D space.

FIGURE 6.11: A schematic example of an issue that may occur if the orientation of the train or camera is incorrect. The blue lines resemble the railway tracks reconstructed from multiple frames. In this case, the orientation of the train lags behind its actual orientation, affecting the orientation of the tracks and causing a zig-zag pattern.

Sections that are reconstructed decently are comparable to the results of our other evaluations, with the exception of two details. Firstly, a zig-zag pattern is visible in the rails. We again expect this to be related to the turn combined with small errors in camera parameters and GPS. This is illustrated in Figure 6.11. Another issue is a gap in the reconstruction. This is caused by the frame filter, and particularly the filter based on optical flow. This is likely caused by the passing train, which increases optical flow values in the filtering window, causing more frames to be removed.

The train itself can not be located. Although some parts are visible (Figure C.2), most points are heavily distorted and scattered all around the point cloud. Additionally, a lot of pixels are filtered out based on their brightness.

From this evaluation we conclude that our implementation is not yet generic enough to be used with passing trains. Although turns also caused issues, we believe this is largely related to errors in input data. These errors could be fixed by increasing the accuracy of sensors, or by finding ways to correct them or filter them out.

6.3 Method Discussion

While executing our reconstruction and evaluation, a couple of issues became apparent related to the method used. These issues were related to the data, time requirements and choice of evaluation. In this section we will discuss these issues.

First of all, the data used for the evaluation was less accurate compared to our previous testing data and contained issues with vertical camera movement, as discussed in

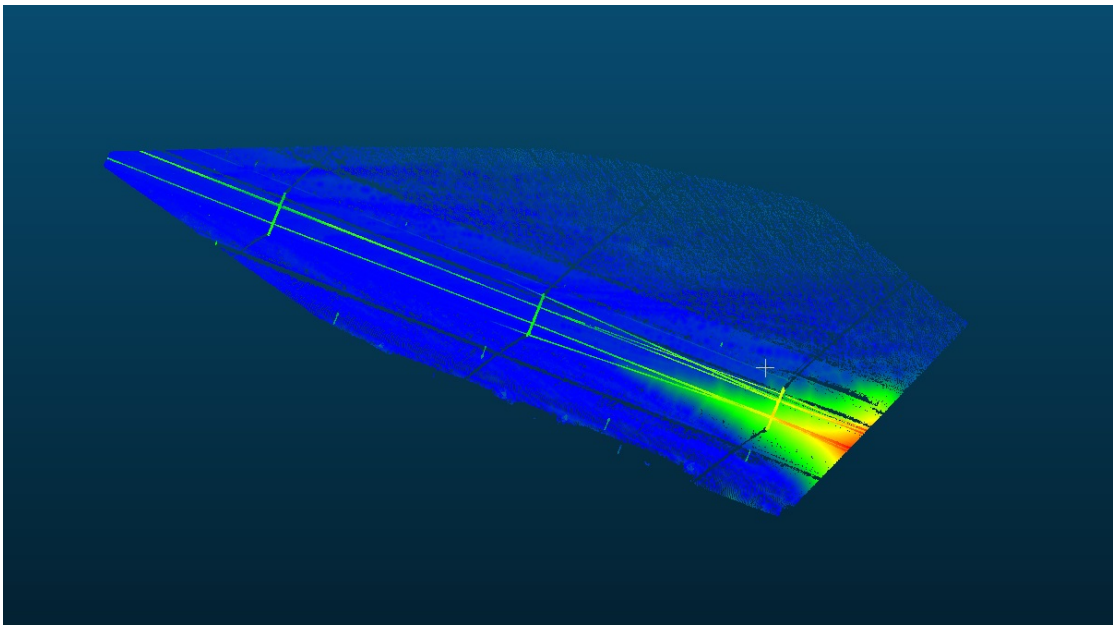


FIGURE 6.12: This figure shows the errors in forward distance that occur near the end of the segment. The color scale ranges from blue (small errors) to red (large errors). The green lines accross the point cloud show that the CCs and power cables are poorly reconstructed. the triangular area with large errors on the right side is a result of a missing section. This image is from the p-rmls variation, but similar patterns occurred in almost all reconstructions.

Section 4.2.2. The GPS data showed irregularities, occasionally containing jumps much larger than 0.5m or staying the same between frames. On the other hand, our testing data almost only contained displacements close to 0.5m, which would be expected. The inaccuracies forced us to filter out frames based on optical flow or GPS information in order to obtain reasonable results. The filters used worked well, but are not very generic. We expect better results to be possible if these issues are not present.

We also underestimated the amount of time required to perform certain steps in the pipeline. Particularly surface reconstruction. In some cases, this would even cause errors due to memory issues. We therefore chose to only use one sixteenth of all pixels (sampled in a regular grid across the image). Combined with our frame filters, this led to a much lower amount of points (129600 as opposed to 2073600), making surface reconstruction possible. However, the intermediate point clouds were much less dense, which may have led to poorer reconstructions and loss of detail.

The time constraints also forced us to use a segment of 200 meters long for our evaluation. As shown in Section 4.3.1, the ground truth data was selected in such a way that it matched the general shape of the reconstruction. However, particularly near the

end of the reconstruction, part of the ground truth data was not represented in the reconstructions. This has a negative effect on the forward distance, which is visible in Figure 6.12. With the size of our reconstructed segment, this effect was very significant. By using a much larger segment, this effect would become negligible.

Using a shorter segment also made it more difficult to pick distances to measure, resulting in fewer measurements. This could increase the effect of outliers and make the evaluation less reliable. Additionally, the vertical and parallel errors are unreliable due to poor reconstructions of CCs. Overall, the axis accuracy was hard to evaluate as details were not as clearly visible as we expected. As a result, the axis accuracy evaluation somewhat became a case of which distances are best measurable, as opposed to which distances are most accurate, although we could still use some of the results for our discussion and conclusions.

Finally, a small issue with our implementation is that some of the reconstructed meshes are not colorized. Solving this problem should be quite trivial in theory, but was not possible for all results within the constraints of our research. Only the variations using Greedy Projection Reconstruction yielded colorized results.

6.4 Research Questions

We will now reiterate our research questions and answer them based on our results and previous conclusions. We will start with our main research question:

Main research question. *How accurately can we create 3D reconstructions of railway environments based on a single front-facing camera mounted on a train, using structure from motion techniques?*

Based on our techniques and implementations, we were able to reconstruct the environment of 200 meters of railway up to a global distance of 1.60 meters. This was done using polynomial triangulation method by Hartley & Sturm [37] (the t-poly variation) on top of our baseline algorithm. The average of the forward and backward distance for this variation was 1.57. Intuitively, this means that when picking a random point on the mesh or ground truth, its corresponding point on the other 3D structure lies at an average distance of 1.57 meters. For 11 out of the 12 total variations, the backward

distance was lower than the forward distance (by 1.06 meters on average for these 11 variations). Since these two measures are not directly comparable, we can not conclude whether or not this means that the accuracy was generally better than the completeness, although we suspect that this is the case based on observations. We did not meet the accuracy requirements in any of the main directions nor on a global level, but we were able to achieve decent results and gather a lot of insights into the techniques and data used. Overall, we conclude that our method reconstructs the general shape of the environment well, but leaves out details. It is also currently very reliant on the quality of the data and certain assumptions. However, we do expect most of these issues to be solvable by improving certain steps in the pipeline using already existing techniques.

Subquestion 4. *How do different triangulation techniques perform in terms of speed and accuracy?*

The midpoint method and polynomial method performed very similarly when combined with the rest of our pipeline. In terms of running time, the differences are too minor to conclude that the midpoint variation is faster, as differences could be a result of chance or slightly different circumstances. The increase in accuracy with the polynomial method does seem plausible, as this method deals with 2D errors optimally. In our pipeline, this should lead to better results when errors occur in the optical flow step. Therefore, we would recommend using the polynomial method over the midpoint method in most cases. However, If time is an important issue, or little noise is expected in the point matches, the midpoint method would be a good option.

Subquestion 5. *How do different surface reconstruction techniques perform in terms of speed and accuracy?*

Although the Greedy Projection Triangulation performs best in terms of accuracy, the s-poisson method does appear promising with tweaked parameters or denser point clouds. s-grid performed quite well, but showed inconsistent and sometimes unrecognizable results. As discussed, s-mc performed poorly and needs adaptations before it can be considered useful. s-mc and s-poisson were much faster than the other methods. However, s-poisson required outlier removal to run without crashing. With increased depth parameters, s-poisson-depth produced better results, but still shows much of the issues of s-poisson. The same applies to our tests with the ground truth point cloud. Based on

our findings, we believe Greedy Projection Triangulation is the best choice for accuracy if noise is not important. Grid Projection shows smoother results and could be used in cases where this is required. The resolution parameter has a large effect on the level of detail and the running time, which is a trade-off. Poisson surface reconstruction is expected to work well with very dense point clouds.

Subquestion 6. *How do different filtering techniques perform in terms of speed and accuracy?*

Filtering techniques can be difficult to evaluate, particularly because they are heavily reliant on parameters. As expected, all filters decreased the backward distance. However, they also increased the forward distance. With our parameters p-radius removed a large number of points, resulting in a large forward distance. The results of p-stat and p-rmls were comparable, though the results from p-rmls looked smoother. With the right parameters, we would expect statistical outlier removal and radius outlier removal to perform similarly, although finding these parameters is easier for statistical outlier removal. Overall, the RMLS method appears most promising and could be an improvement over the baseline with different parameters. Even with a higher global distance, the qualitative improvements on the reconstructed surfaces could be worth it. Outlier removal techniques can reduce the running time of the surface reconstruction if enough points are removed, while RMLS appears to increase the total running time unless a small search radius is chosen. The search radius parameter of the RMLS causes a trade-off between running time and accuracy on one side, and the amount of smoothing on the other.

6.5 Future Work

To conclude, we will look at potential areas for future work for both the implementation side (Section 6.5.1) as well as the physical setup used (Section 6.5.2). Our research has yielded decent results, but there is still work to be done in order to make the results usable for our applications, as the requirements were not met.

6.5.1 Implementation

First we will look at the steps of the pipeline to identify steps that can be improved in order to increase the accuracy of the reconstruction. We note that if the positional data, camera data and optical flow computations were entirely correct, the point cloud resulting from the triangulation step would be a perfectly accurate representation of the boundary of the physical world. In this case it would not matter which triangulation method we use, as rays cast through corresponding pixels would always intersect. Sky regions could then be filtered out by removing pixels with extremely low optical flow values. Based on the results from the surface reconstruction techniques, we are confident that the greedy projection triangulation can generate very accurate meshes if the point cloud itself is accurate. In this case, point cloud processing techniques would not be required to increase accuracy, although they could be used to reduce the number of points and increase efficiency. This leaves the optical flow computation and input data as bottlenecks for the accuracy.

In order to improve the positional data, techniques used in SLAM discussed in Section 2.2, such as the Kalman filter [50], could be interesting. However, simpler techniques based on assumptions about the data could also be used. For example, we know that the train should move approximately the same distance (0.5m in our case) between frames. Therefore, simply drawing a curve through all GPS points and redistributing the points evenly across this curve could fix wavering motions. To rely less on positional and camera data, pose estimation techniques described in Section 2.1.2 [32–34] based on homography and point matching could be applied. Methods by Pollefeys et al. [49] and Newcombe & Davison [51] use similar principles to achieve promising results in real-time and could be combined with our pipeline.

Optical flow could be improved by tweaking parameters or using techniques that rank high on the various benchmarks. However, the latter would likely come at a huge price in terms of running time. Even with the current optical flow method, which is relatively fast, computations take around 12 seconds per frame. This is more than expected based on the Middlebury benchmark [4, 5] (2.5 seconds per frame). This is likely mostly related to the fact that images used in the Middlebury benchmark are much smaller (467 by 310 pixels) compare to our input (1920 by 1080 pixels). Additionally, we use a large patch radius and median filter size. Setting both these parameters to 5, we found that

FIGURE 6.13: An example of an optical flow simulation using a patch radius and median filter radius of 5.

the running time is reduced to an average of 5.1 seconds per frame. However, with these parameters, edges were not clearly preserved and results were much more noisy, as can be seen in Figure 6.13.

To circumvent the speed problem in the optical flow step, less dense point matching techniques could be used. Only matching keypoints would likely increase speed as well as accuracy, as these points are generally easier to match than points on edges or in textureless regions. This could potentially reduce detail in the reconstructions, but investigating this trade-off seems an interesting research problem. Regarding the optical flow technique by Bao et al. [20], using different parameters could improve performance. Although we experimented with many different settings, trying all possible combinations was not possible within our research. Examples of parameters that could be tweaked are the patch radius, number of pyramid levels and median filter radius. The effect of the patch radius on our results is not very necessarily intuitive, except for the fact that a higher patch radius increases the running time. We found mixed results with different settings and were limited in the maximum patch size due to memory issues. For the image pyramids, downscaling twice with a factor of 0.5 is common. We found that downsampling only once improved the optical flow for small structures. However, it would also decrease performance for large objects near the camera. Perhaps using a different downscaling factor could yield the best of both worlds. A stronger median

filter clearly reduces the number of optical flow outliers, but also reduces accuracy for small structures such as CCs. Before computing optical flow, several image processing techniques could be used to increase the accuracy of the optical flow. For example, edges could be accentuated and noise could be reduced.

Assuming accuracy of the input data and optical flow computation will remain insufficient for our current pipeline, more sophisticated filtering techniques could be applied to improve results. For example, more dynamic frame filtering techniques should be researched to make our pipeline applicable to different types of motion and camera setups, as ours (Section 4.2.2) are specific to the assumptions based on our input data. As an example, the window used for filtering based on optical flow could be changed based the estimated movement and camera position. To filter sky regions, more sophisticated color segmentation using Guassian Mixture Models [63] or K-means [64] could provide better results in more situations, as these techniques are more generic. These techniques could also be used to filter passing trains, although constructing a model for each type of train seems infeasible. Further research is required to find ways to filter out trains. One option would be to filter pixels based on the size of their optical flow.

Improved ground alignment or point cloud registration techniques could be used to correct positional errors. Additionally, more types of environments could be supported. Our ground alignment techniques only works when most points are ground points, which is true for the data we used for our evaluation. However, it would likely fail in mountainous environments or tunnels. For these cases, further improvements and smarter techniques are possible in order to improve support for different environments and find ways to solve issues with the input data.

Although surface reconstruction does not seem to produce the most issues in terms of accuracy for our results, more techniques and implementations could be researched. Particularly, improvements regarding the running time are required. Parallel variations such as those used for Poisson surface reconstruction [65, 66] have shown promising results.

In general, nearly all techniques used in our pipeline rely heavily on parameters. Although we set these parameters mostly based on qualitative evaluations, this process is still a bit arbitrary in some cases. We believe our parameter choices were good overall,

but empirical research could provide more insights into the effects of different parameter settings under different conditions. Benchmarks such as those that exist for optical flow [4–9] also provide valuable quantitative data that can be used to select techniques. Creating these benchmarks for other research areas could prove very useful.

To evaluate our results with respect to our applications, more qualitative evaluation could provide interesting results and feedback. As an example, train drivers could be asked to drive through a simulation based on our reconstructions. In the case of simulations, their feedback could prove invaluable, as they will ultimately be the ones using the resulting 3D reconstructions. For the design applications, this might be less important, as in this case accuracy matters more than results looking nice or realistic. However, even for this application, qualitative research could be beneficial, as the architects and designers that might end up using the 3D reconstructions can give valuable input on their usability.

6.5.2 Physical Setup

We will conclude with some suggestions for the physical setup used to gather the data. Although we implemented methods to deal with inaccuracies, improving the quality of the input can be beneficial to the results. Additionally, using different camera setups increases the number of possibilities and completeness of the results. We will now discuss several suggestions.

First of all, positional data could naturally be improved by increasing the accuracy of the GPS. Overall, the current data is quite accurate, but contains outliers. Additionally, the camera position and orientation relative to the GPS sensor do not seem to be entirely accurate. Particularly the yaw of the camera might be inaccurate, as results appear slightly sheared in the direction perpendicular to the railway tracks.

On top of the currently gathered data, tracking the vertical motion of the camera could be an effective way to improve the alignment of multiple point clouds and would make it easier to perform reconstructions around non-flat areas such as overpasses or tunnels. It would also make dealing with camera shaking easier. A simple accelerometer, perhaps combined with a magnetometer and gyroscope could be sufficient to achieve these goals effectively. However, measures may have to be taken to prevent drifting.

The currently captured video data could be improved in several ways. This would likely make it easier for optical flow methods to do their work. Related to our previous notes on camera movement, a stabilizing mount for the camera could be used to prevent shaking. To improve image quality, better camera settings or a better camera could be used. The current camera appears to have issues with changing lighting conditions and sometimes seems to adapt slowly or overcompensate with respect to the image brightness. Furthermore, the colors in the captured image do not appear very vivid, and a small amount of noise is present. Improving these two features could make it easier for optical flow techniques to distinguish between objects and recognize edges. Lastly, increasing the image resolution appears to have a positive effect on results, as tests with downscaled images provided worse results. However, the current resolution of 1920 by 1080 pixels appears sufficient and increasing the resolution further would increase the overall running time.

The current position of the camera limits the completeness of the reconstruction and arguably makes optical flow computation difficult compared to other camera positions. Obviously, certain regions such as the backs of signs are not visible to the current camera and could therefore never be fully reconstructed. With the current setup, objects or regions in the environment change scale in the image as they get closer to the camera. This can be difficult to deal with for many optical flow techniques. Although Bao et al. [20] do not make explicit statements about scale-invariance, one could deduce that the technique is not entirely scale-invariant, as point matching is performed using a simple feature vector consisting of pixel colors in a (single-size) patch around the point. We conclude that side-facing cameras on both sides of the train could improve the reconstruction as objects viewed from this angle show motion that is consistent between frames and generally do not change in size. The consistent motion could make it less likely for objects to appear in multiple locations in the reconstruction, while the constant scale could make optical flow computation easier. However, in this setup, a front- or rear-facing camera would still be required to reconstruct the rails. Besides side-facing cameras, using wider-angle cameras or 360 cameras seem very interesting options as they cover a larger part of the environment. 360 cameras often use multiple lenses to achieve a complete image. Therefore, this could be seen as a solution using multiple cameras. Overall, the reconstruction would likely take longer if the cameras individually produce HD images, as simply more camera images are used. Optionally, this solution could

be combined with wide-baseline matching between the different cameras and multiple frames in order to increase accuracy for points appearing in multiple views.

To conclude, a lot of relatively cheap options exist to improve the input data. Together with more research and improvements in the pipeline, reconstructing railway environments based on camera footage appears within grasp.

Appendix A

Parameters

Optical flow [20]	
Pyramid layers	3
Pyramid ratio	0.5
Search range	128
Random guesses	8
Patch radius (r)	7
σ_s	5.25
σ_r	2.5
Median filter radius	24

TABLE A.1: The parameters used for optical flow computation.

Pixel filtering	
Min. flow	2.0
Min. edge distance	70
Max. brightness	50%
Frame filtering	
Window (x)	$1000 \leq x \leq 1100$
Window (y)	$220 \leq y \leq 300$
Max. avg. window flow	1.0
Min. train motion	0.3
Max. train motion	0.6

TABLE A.2: The parameters used for image filtering.

Statistical outlier removal [38]	
σ multiplier	1.0
Neighbours	30
Radius outlier removal [38, 39]	
Radius	0.25
Min. neighbours	3
RMLS [38, 39]	
Search radius	0.5

TABLE A.3: The parameters used for point cloud processing.

Greedy projection triangulation [46]	
Search radius	1.0
μ	4.0
Max. nearest neighbours	2000
Max. surface angle (rad)	$\pi/4$
Min. angle (rad)	$\pi/18$
Max. angle (rad)	$2\pi/3$
Marching cubes [43, 47]	
Grid resolution	200^3 cells
Grid projection [48]	
Resolution	0.001
Padding size	3
Max. binary search level	10
Poisson [44]	
Depth	10
Min. depth	6
Degree	2
Point weight	4
Samples per node	16
Scale	1.1

TABLE A.4: The parameters used for surface reconstruction techniques.

Appendix B

Quantitative Evaluation Results

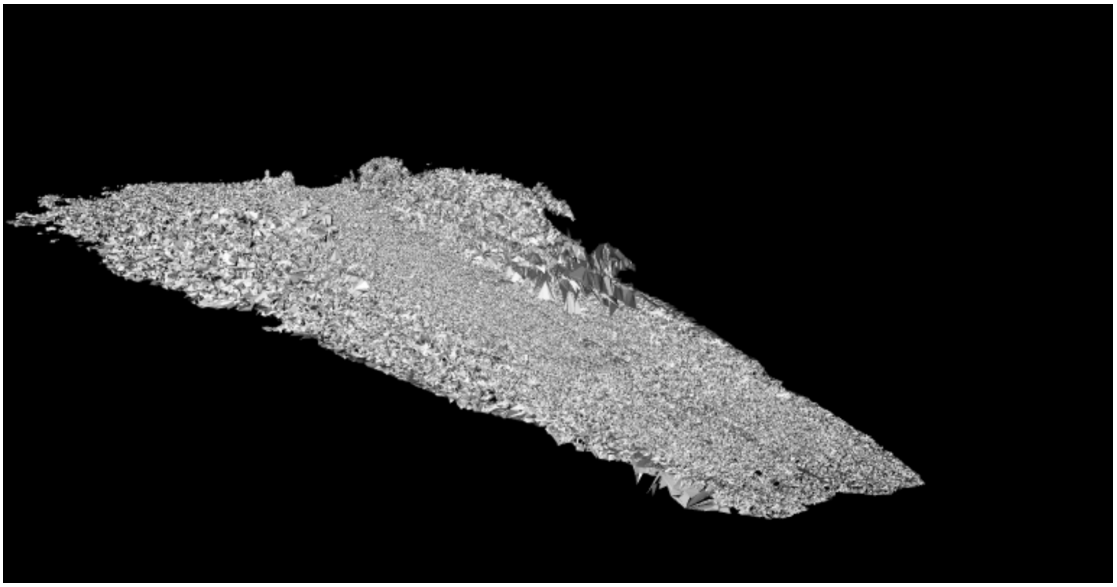


FIGURE B.1: The mesh resulting from the baseline algorithm.

FIGURE B.2: The colorized mesh resulting from the baseline algorithm.

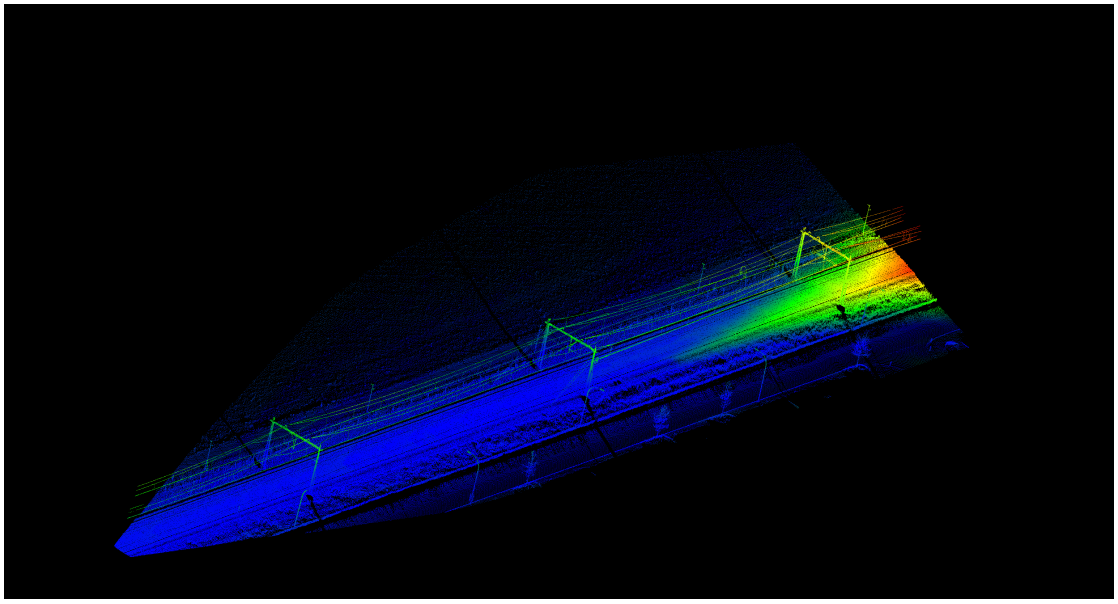


FIGURE B.3: The forward distances for the baseline algorithm mapped onto the ground truth point cloud. Ranging from 0 (blue) to 16.55 (red).

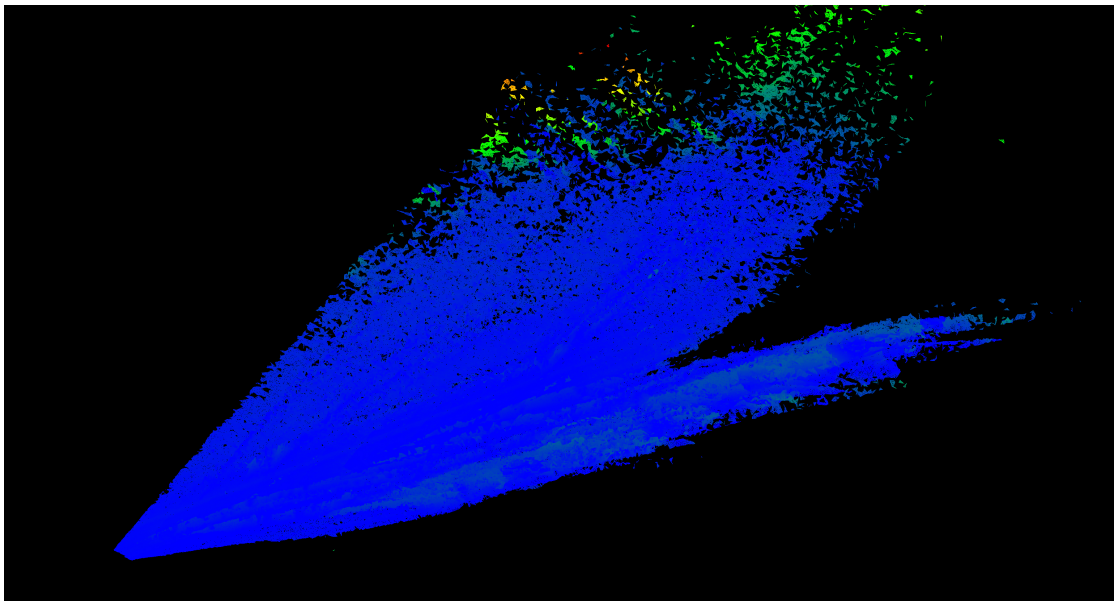


FIGURE B.4: The backward distances for the baseline algorithm. Ranging from 0 (blue) to 38.90 (red).

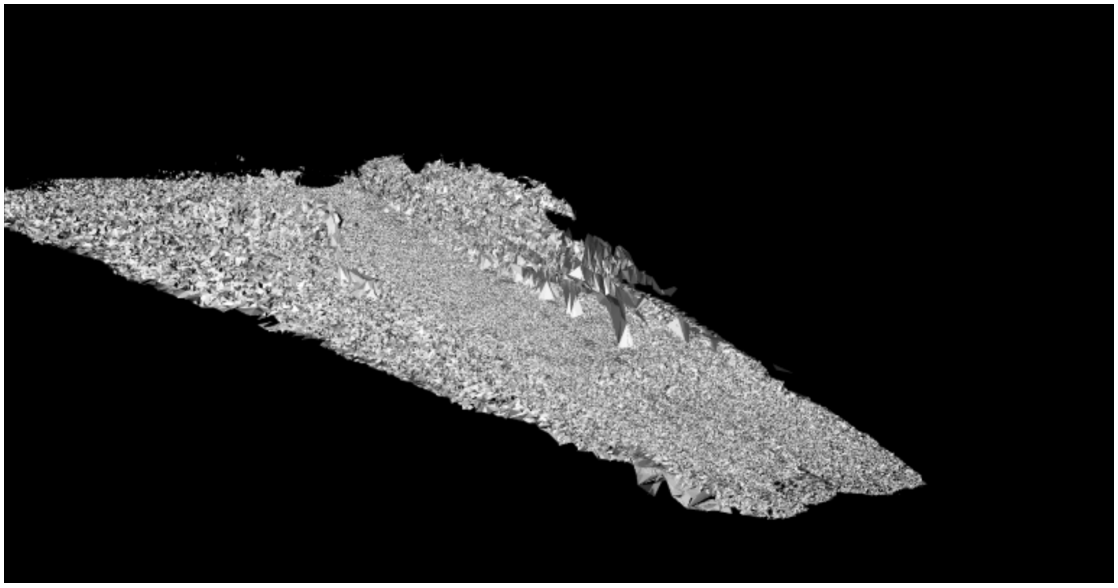


FIGURE B.5: The mesh resulting from the t-poly variation.

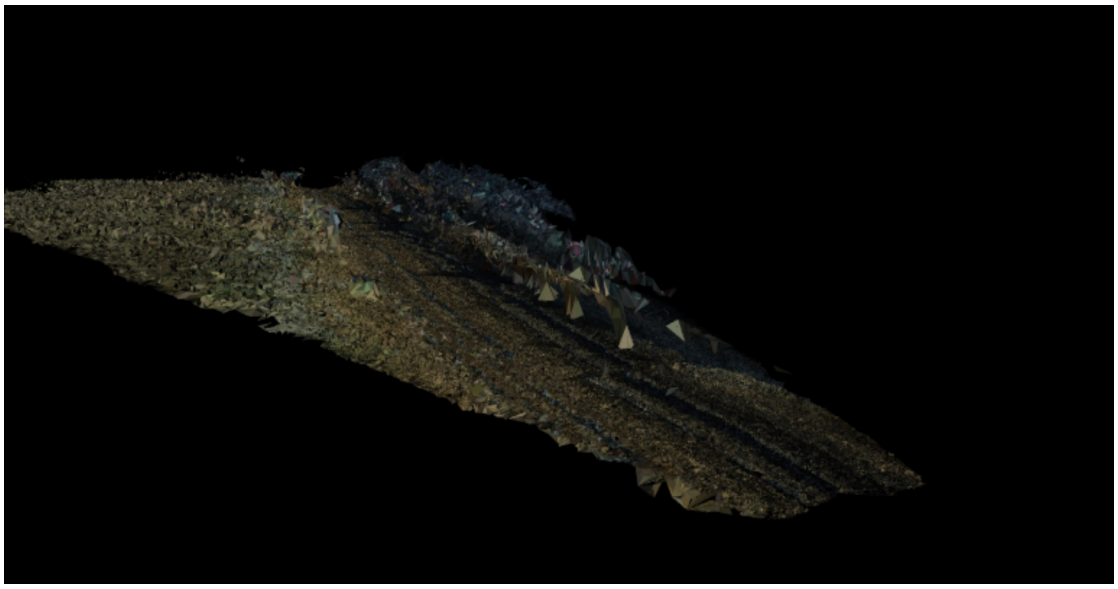


FIGURE B.6: The colorized mesh resulting from the t-poly variation.

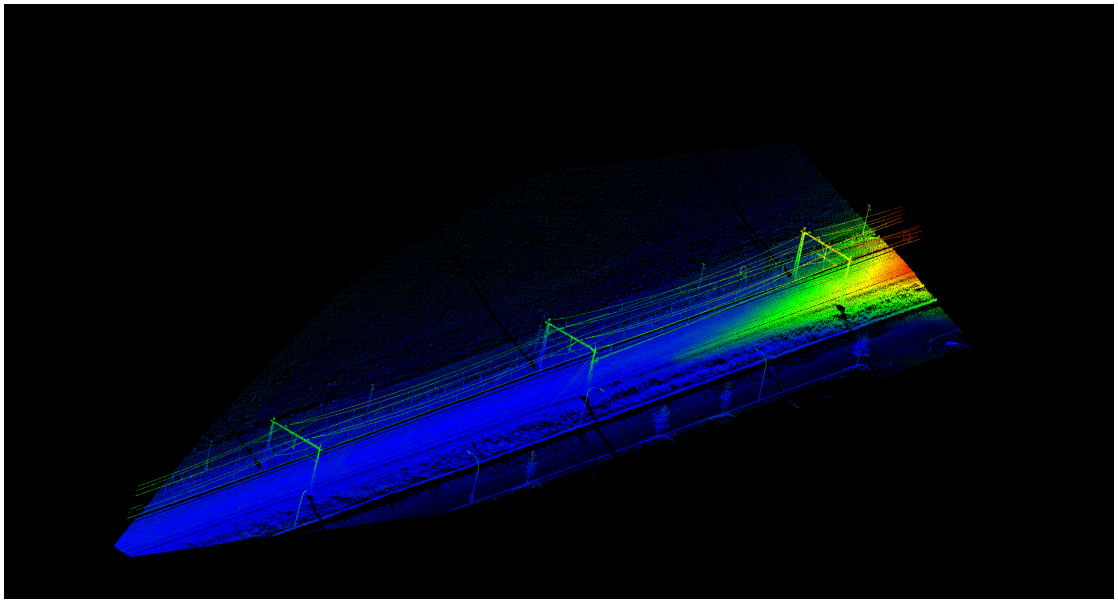


FIGURE B.7: The forward distances for the t-poly variation mapped onto the ground truth point cloud. Ranging from 0 (blue) to 16.42 (red).

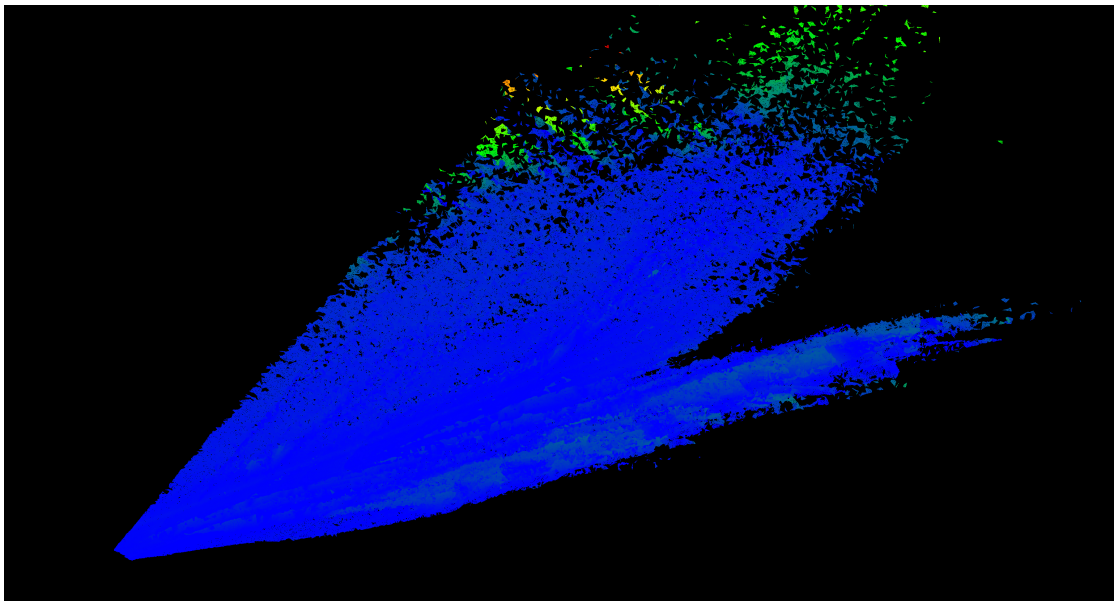


FIGURE B.8: The backward distances for the t-poly variation. Ranging from 0 (blue) to 38.72 (red).

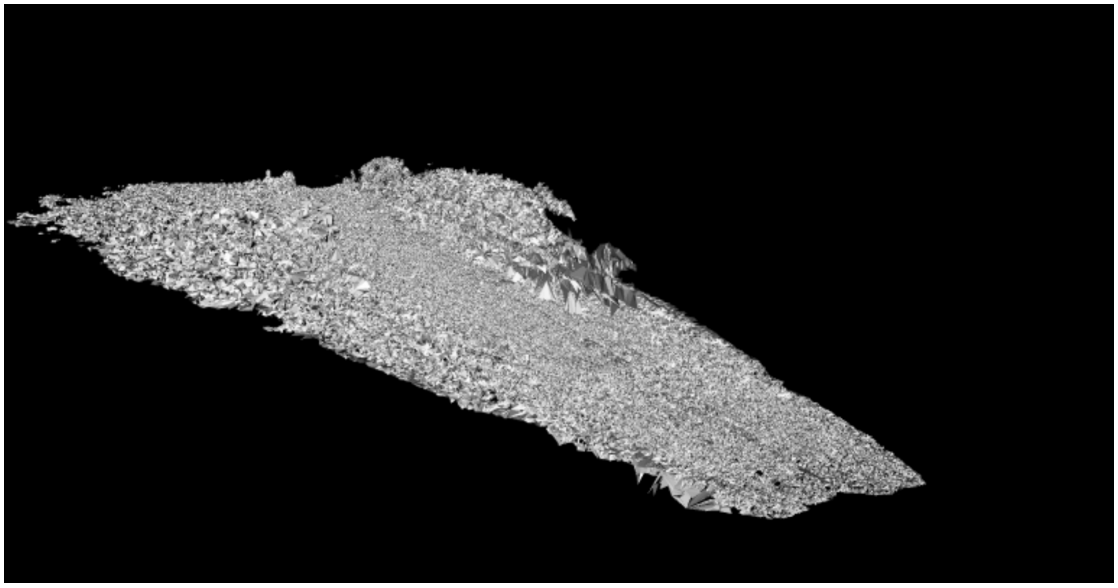


FIGURE B.9: The mesh resulting from the p-stat variation.

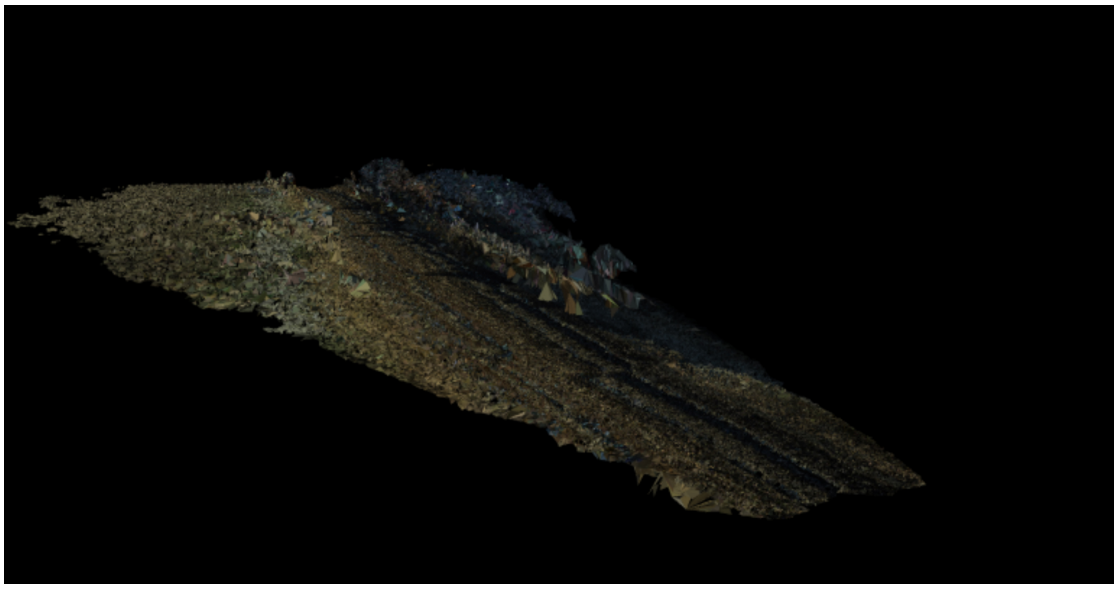


FIGURE B.10: The colorized mesh resulting from the p-stat variation.

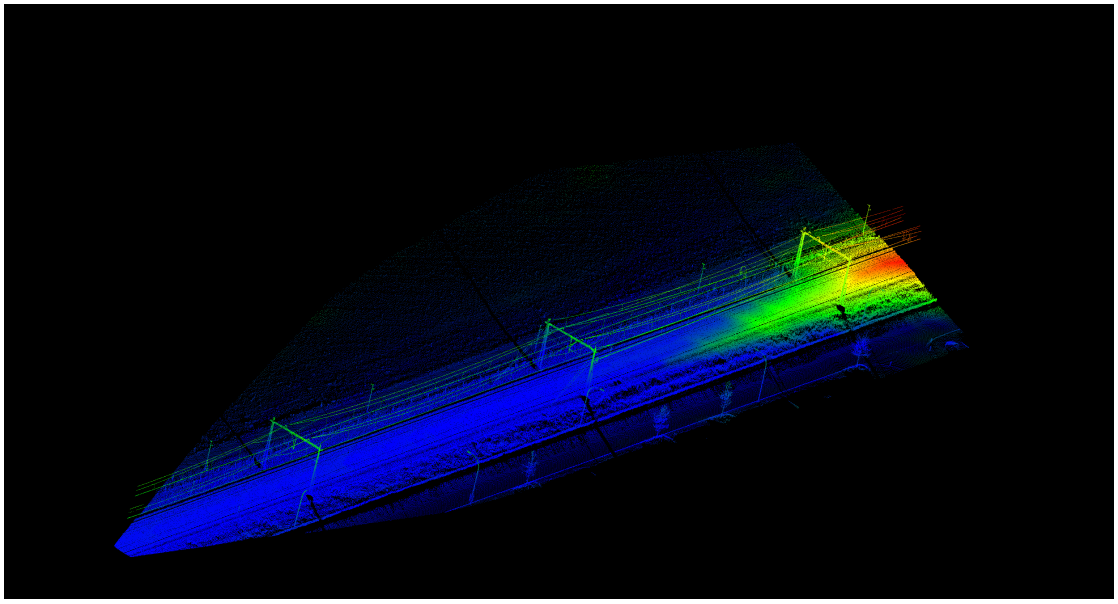


FIGURE B.11: The forward distances for the p-stat variation mapped onto the ground truth point cloud. Ranging from 0 (blue) to 18.68 (red).

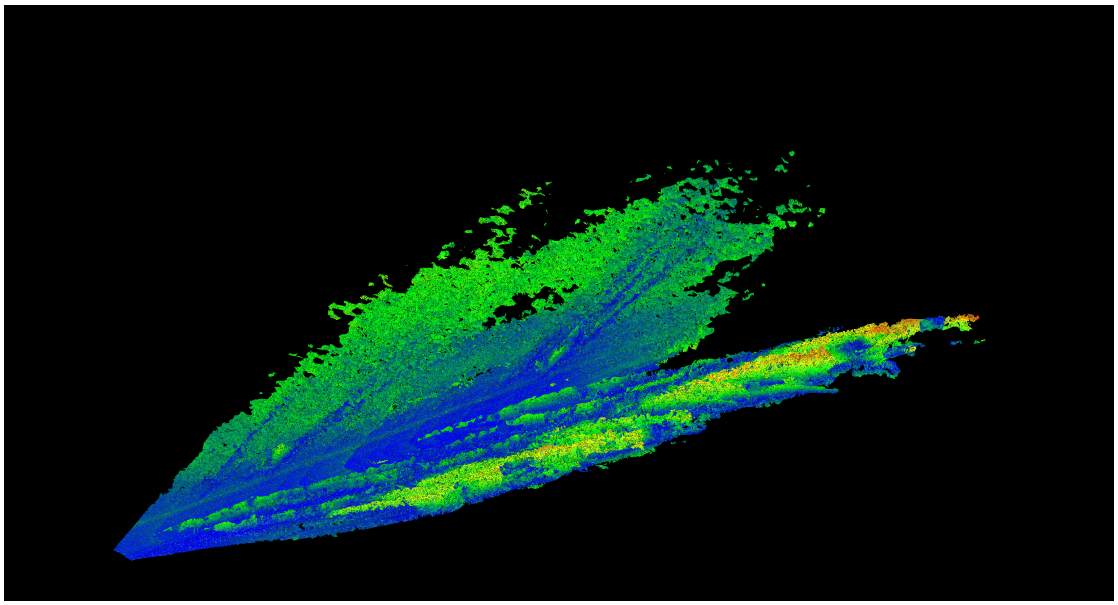


FIGURE B.12: The backward distances for the p-stat variation. Ranging from 0 (blue) to 5.30 (red).

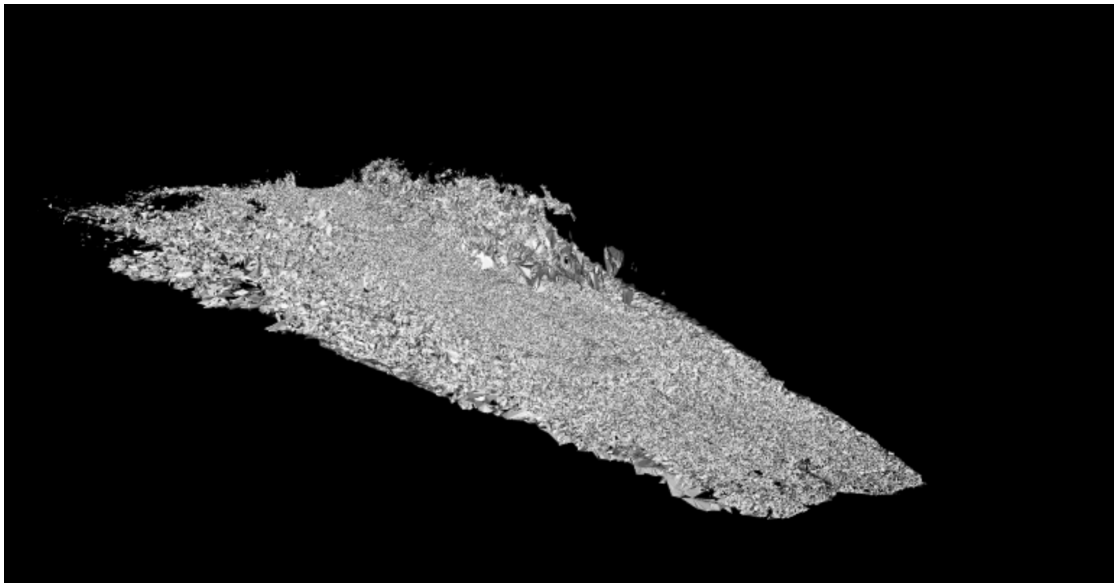


FIGURE B.13: The mesh resulting from the p-radius variation.

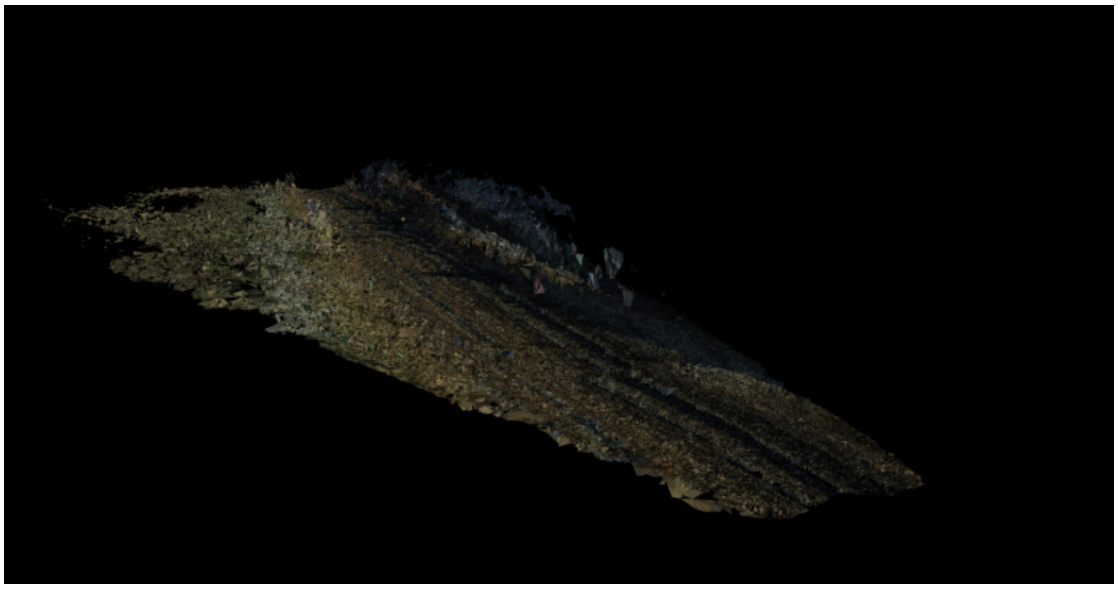


FIGURE B.14: The colorized mesh resulting from the p-radius variation.

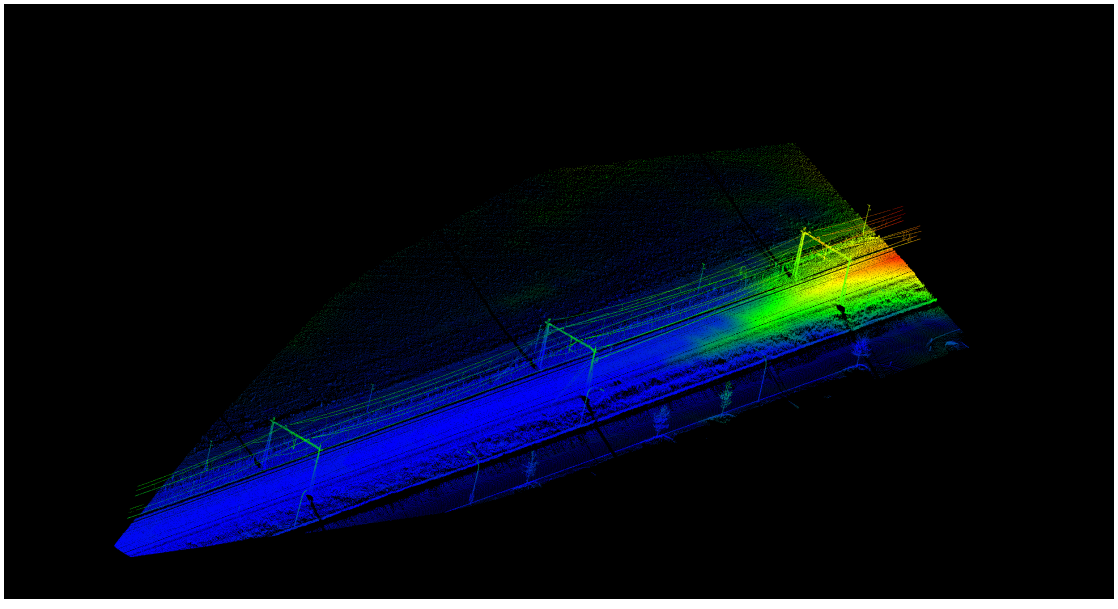


FIGURE B.15: The forward distances for the p-radius variation mapped onto the ground truth point cloud. Ranging from 0 (blue) to 16.42 (red).

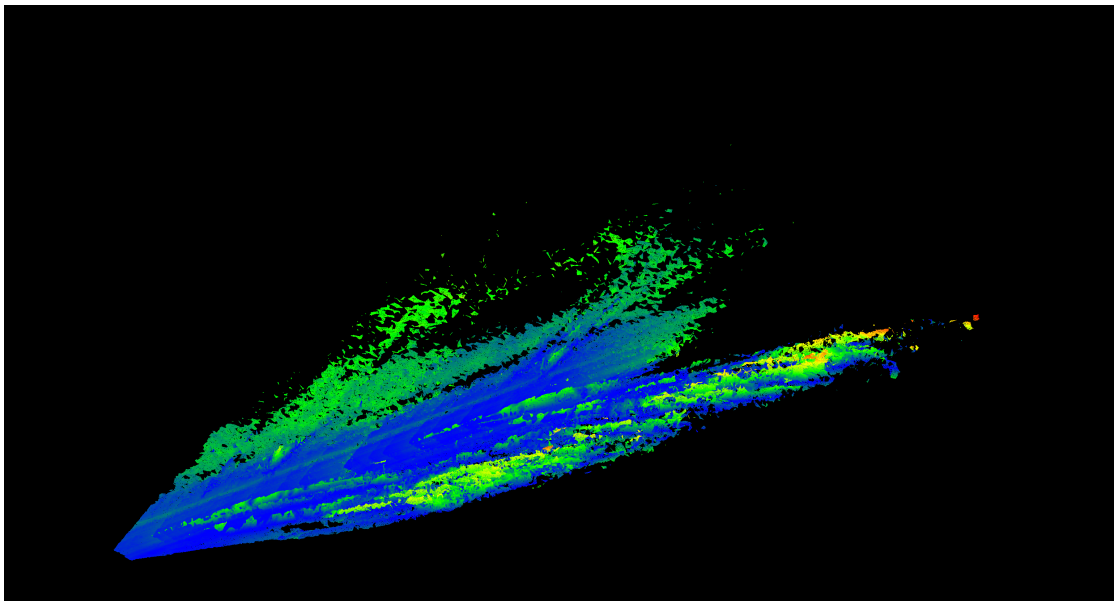


FIGURE B.16: The backward distances for the p-radius variation. Ranging from 0 (blue) to 5.07 (red).

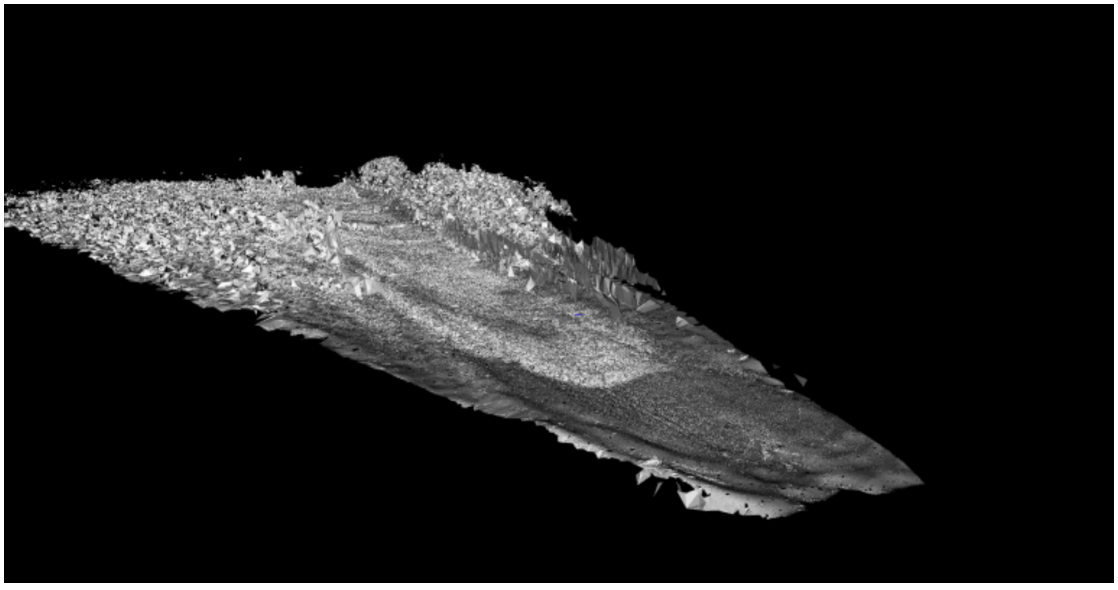


FIGURE B.17: The mesh resulting from the p-rmls variation.

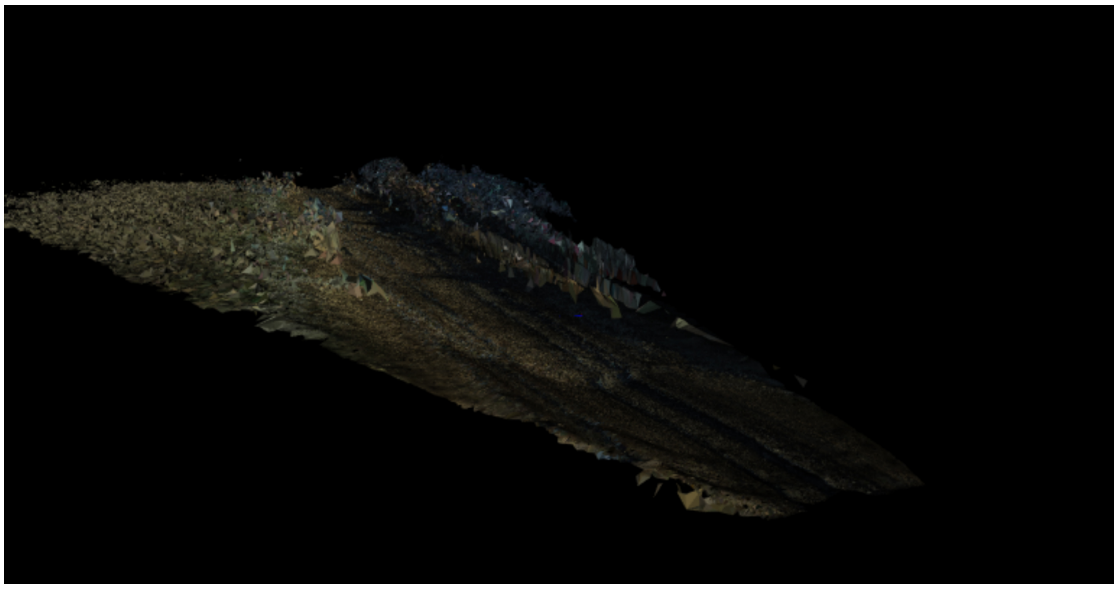


FIGURE B.18: The colorized mesh resulting from the p-rmls variation.

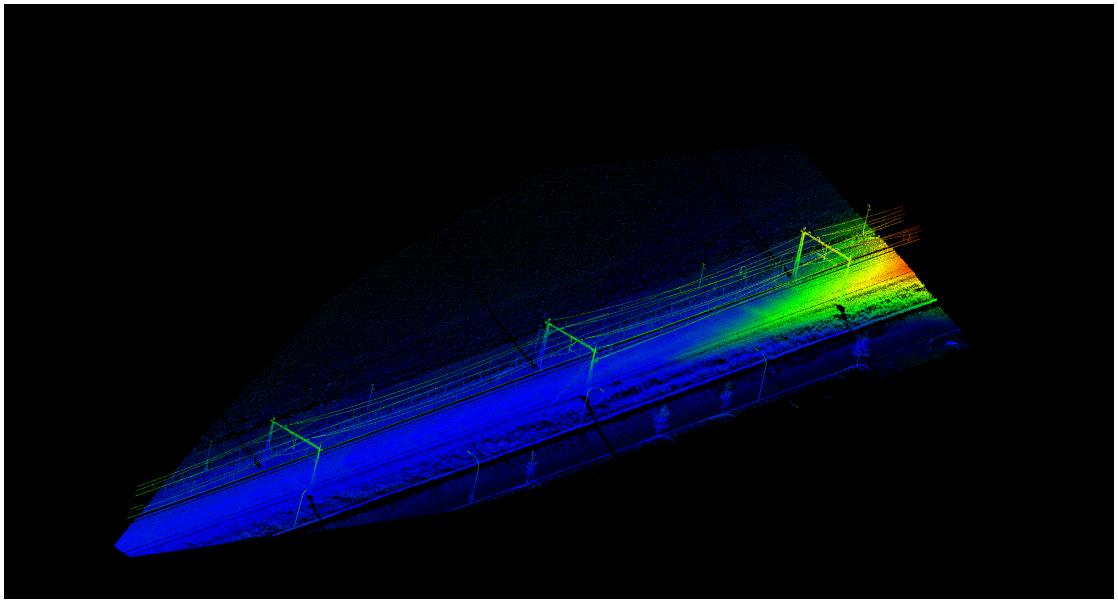


FIGURE B.19: The forward distances for the p-rmls variation mapped onto the ground truth point cloud. Ranging from 0 (blue) to 17.80 (red).

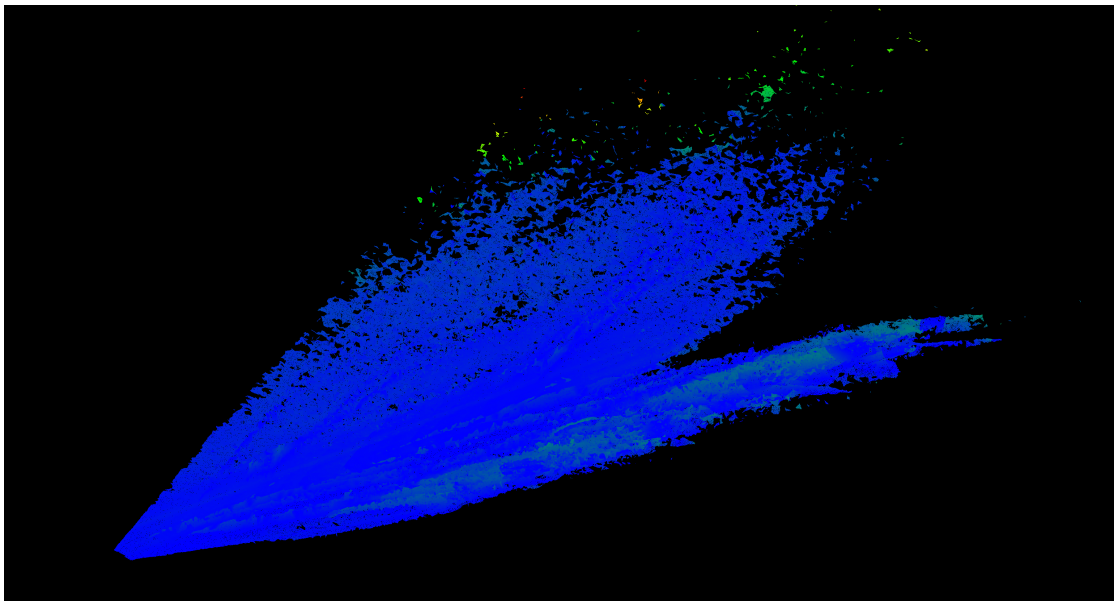


FIGURE B.20: The backward distances for the p-rmls variation. Ranging from 0 (blue) to 28.71 (red).

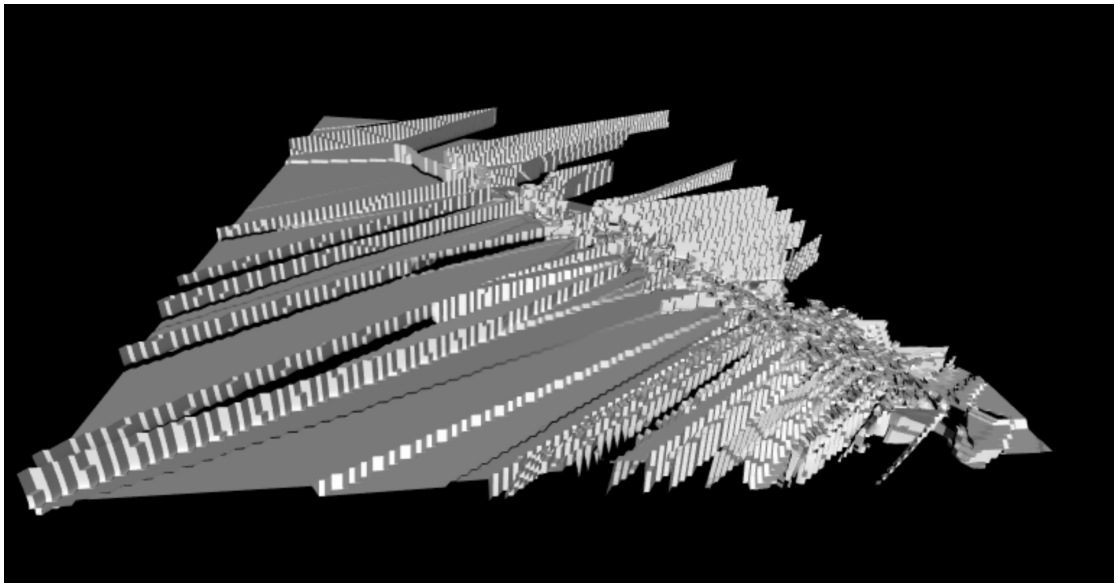


FIGURE B.21: The mesh resulting from the s-mc variation.

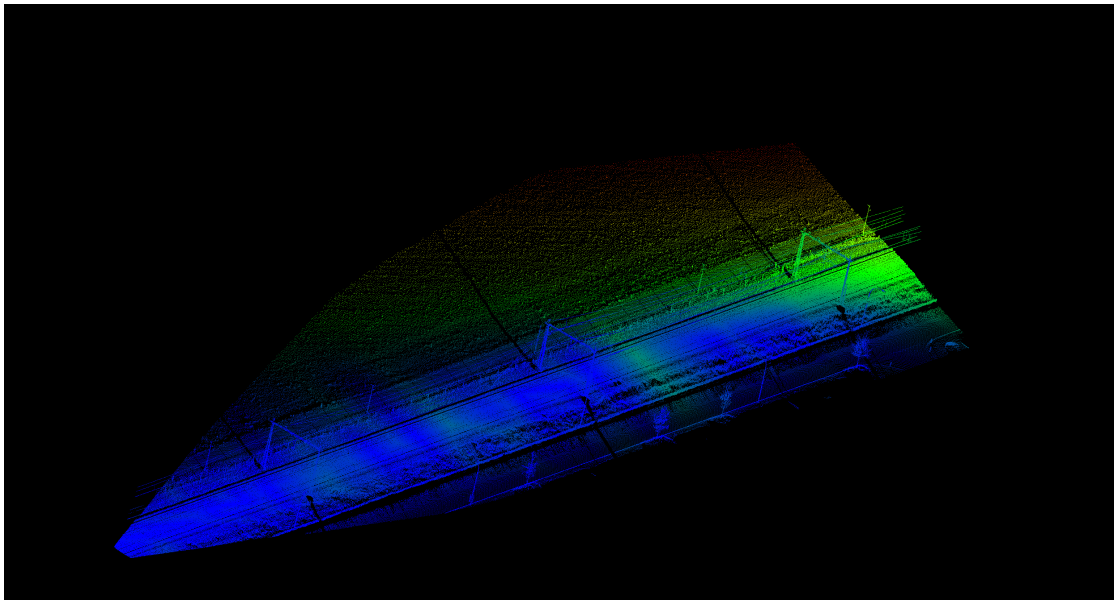


FIGURE B.22: The forward distances for the s-mc variation mapped onto the ground truth point cloud. Ranging from 0 (blue) to 56.16 (red).

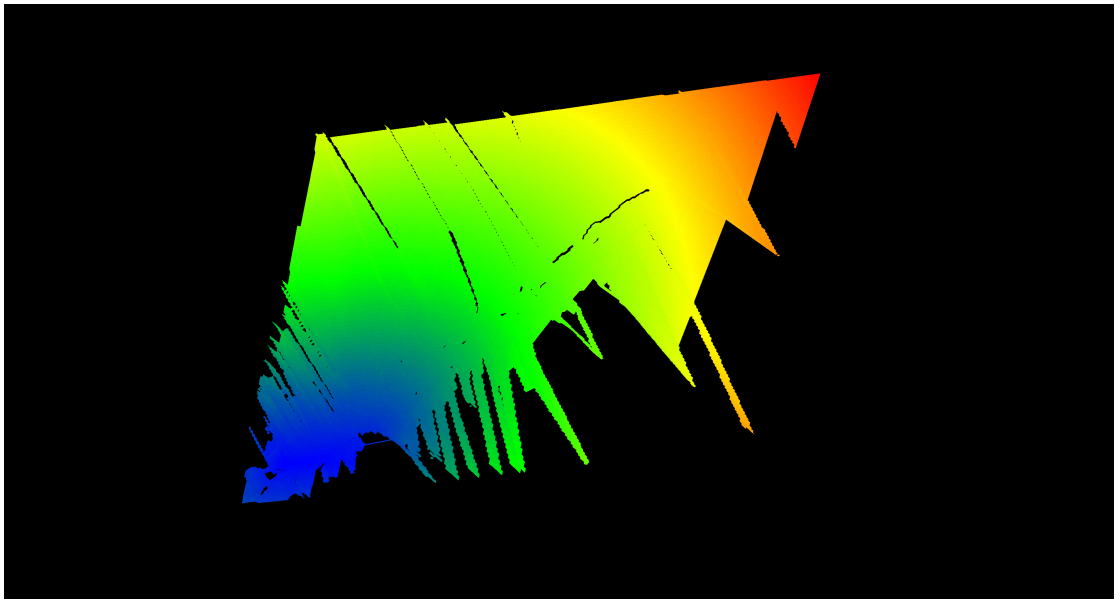


FIGURE B.23: The backward distances for the s-mc variation. Ranging from 0 (blue) to 1309.42 (red).

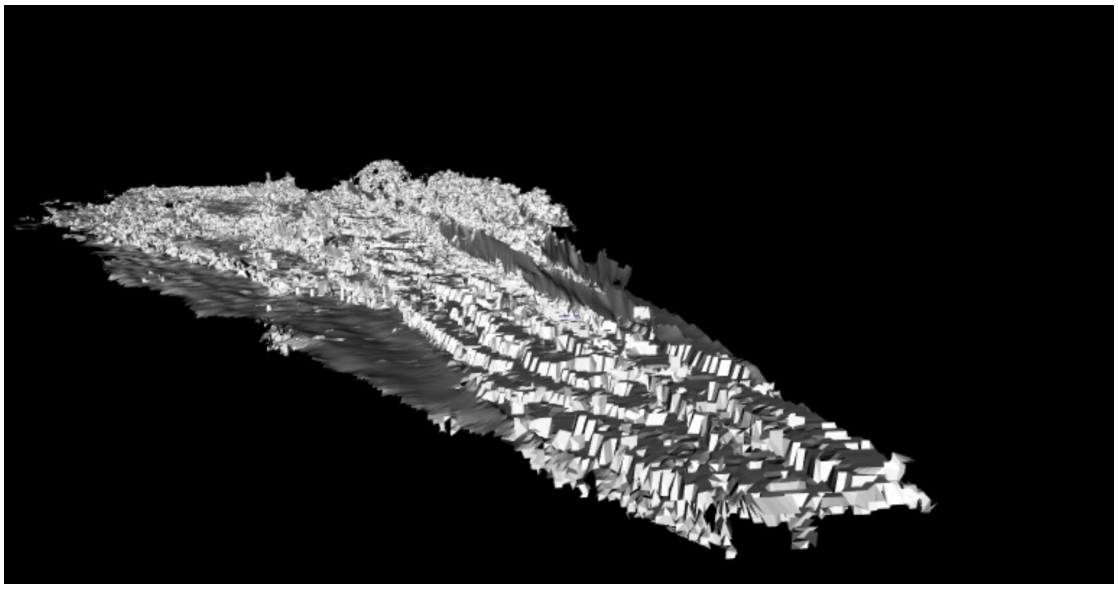


FIGURE B.24: The mesh resulting from the s-grid variation.

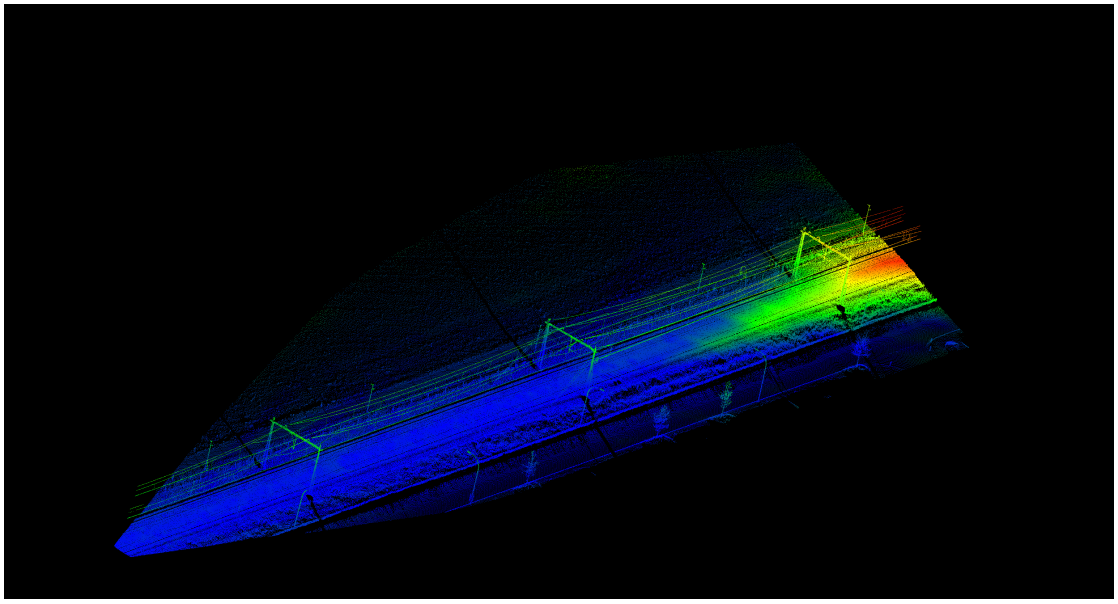


FIGURE B.25: The forward distances for the s-grid variation mapped onto the ground truth point cloud. Ranging from 0 (blue) to 19.08 (red).

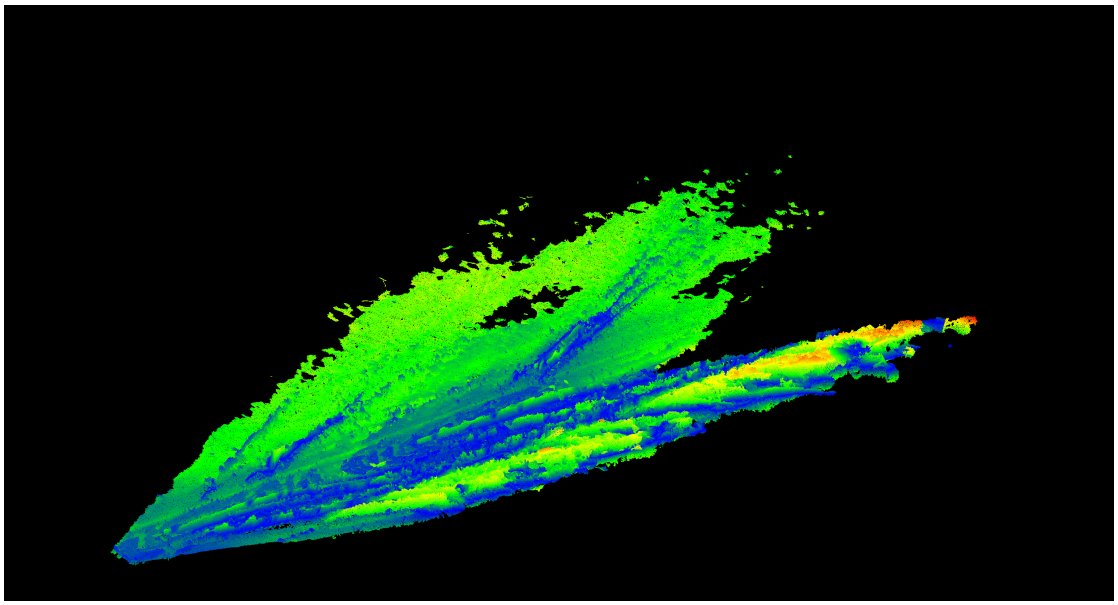


FIGURE B.26: The backward distances for the s-grid variation. Ranging from 0 (blue) to 4.82 (red).

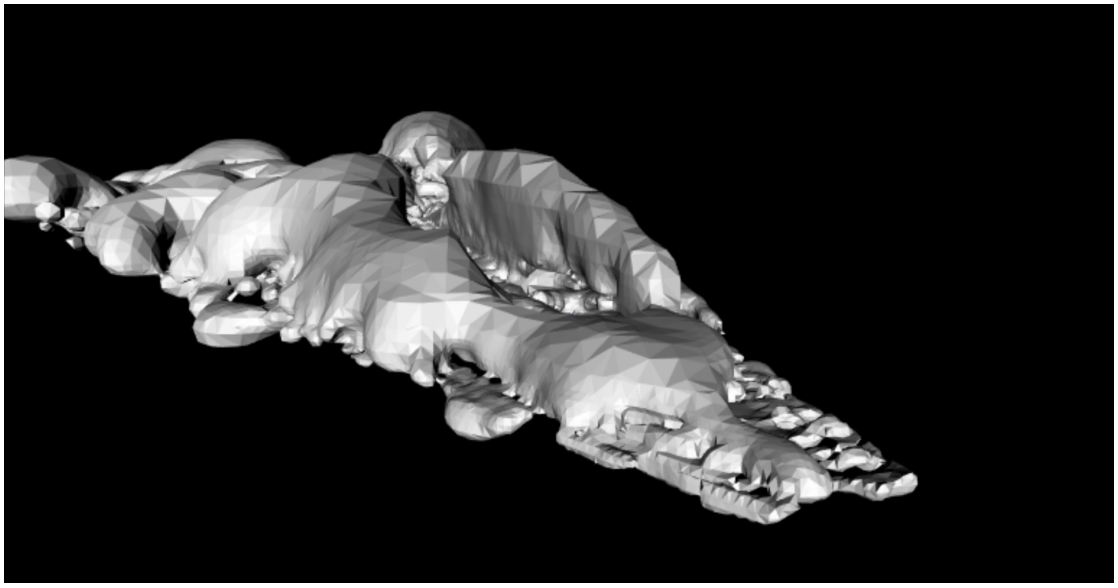


FIGURE B.27: The mesh resulting from the s-poisson variation.

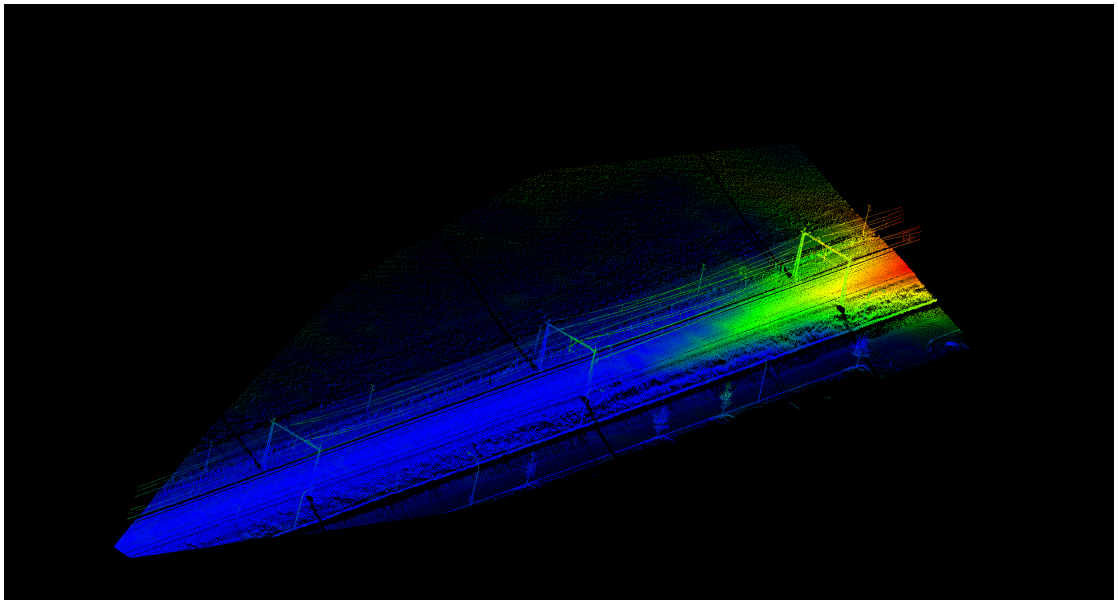


FIGURE B.28: The forward distances for the s-poisson variation mapped onto the ground truth point cloud. Ranging from 0 (blue) to 22.58 (red).

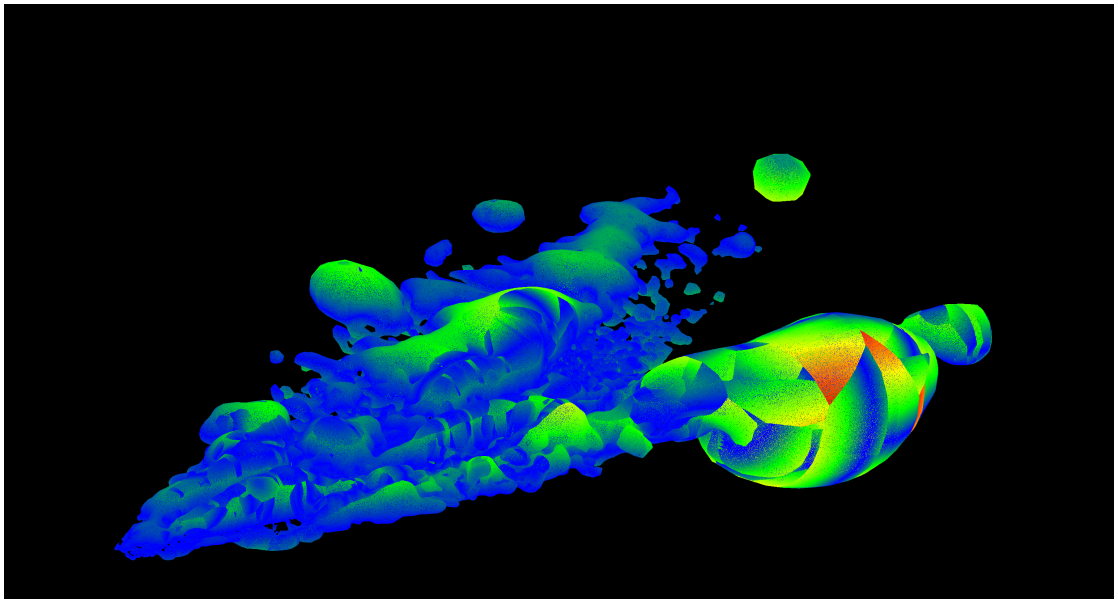


FIGURE B.29: The backward distances for the s-poisson variation. Ranging from 0 (blue) to 14.93 (red).

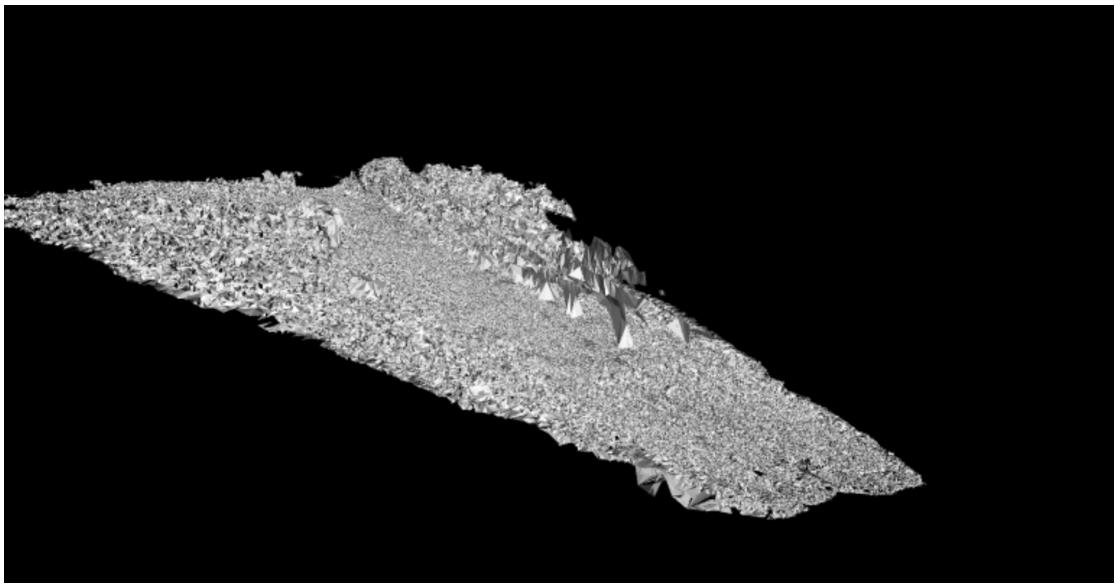


FIGURE B.30: The mesh resulting from the p-stat-large variation.

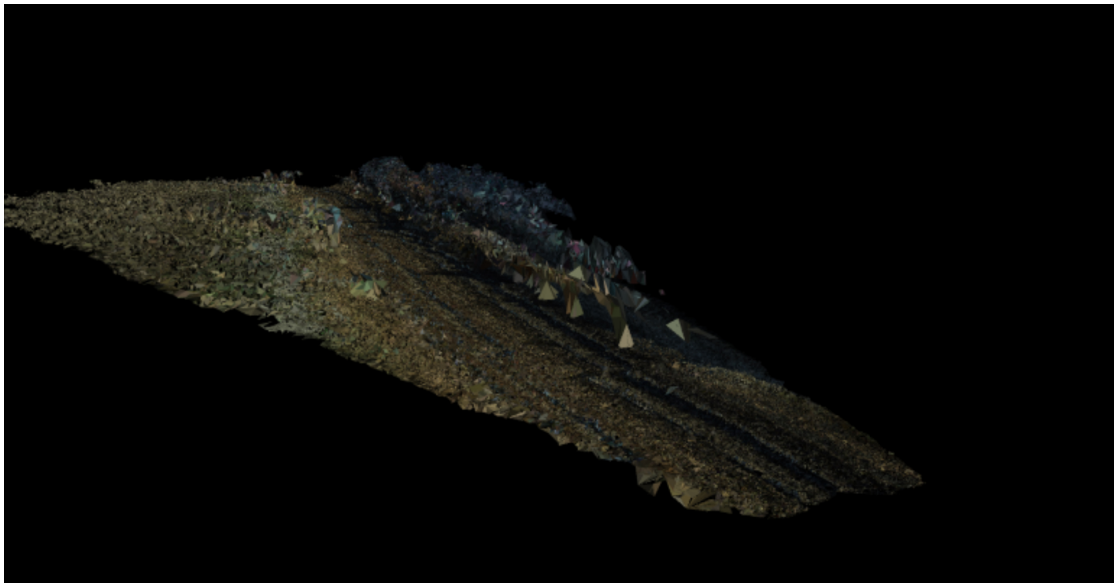


FIGURE B.31: The colorized mesh resulting from the p-stat-large variation.

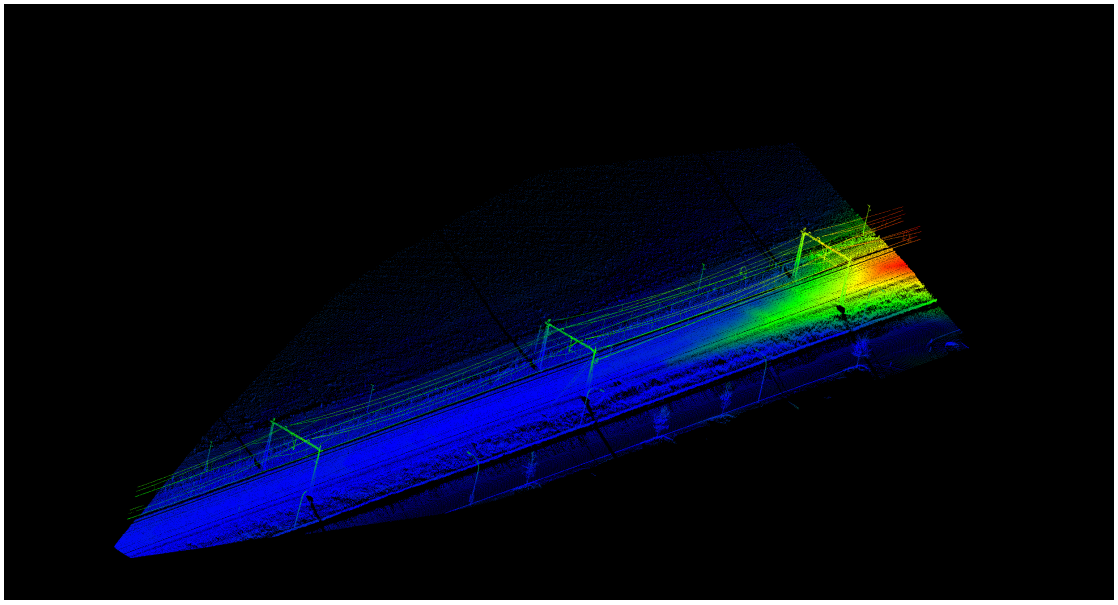


FIGURE B.32: The forward distances for the p-stat-large variation mapped onto the ground truth point cloud. Ranging from 0 (blue) to 17.85 (red).

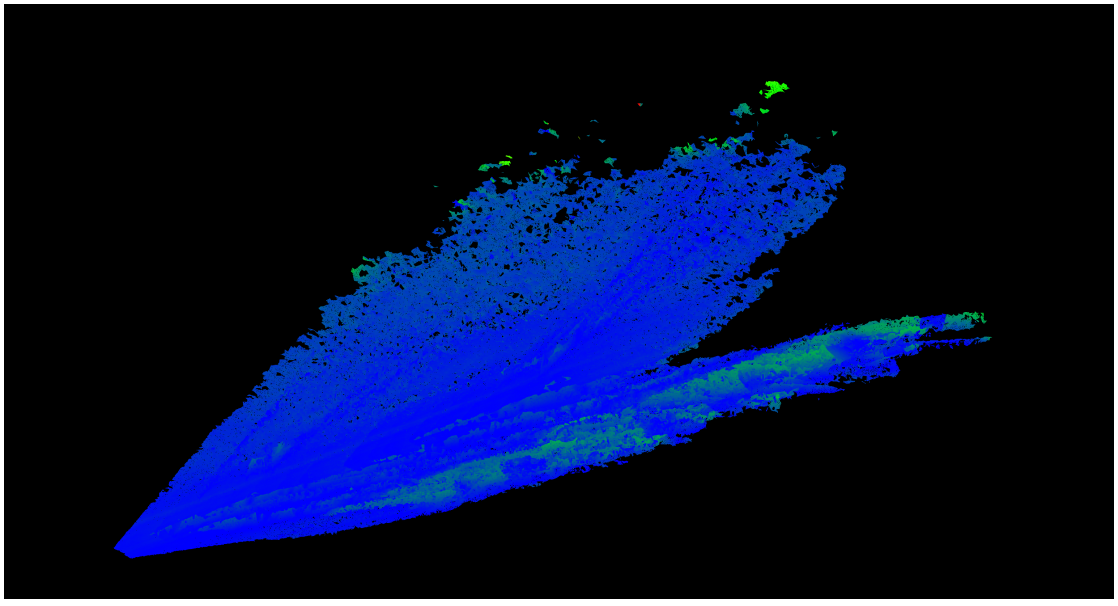


FIGURE B.33: The backward distances for the p-stat-large variation. Ranging from 0 (blue) to 20.50 (red).

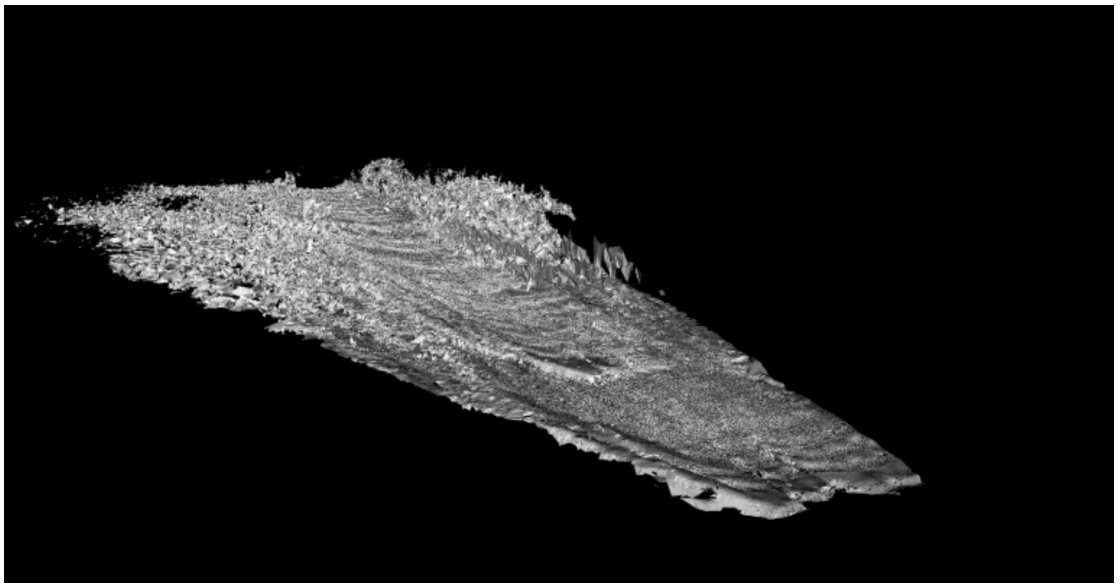


FIGURE B.34: The mesh resulting from the p-rmls-small variation.

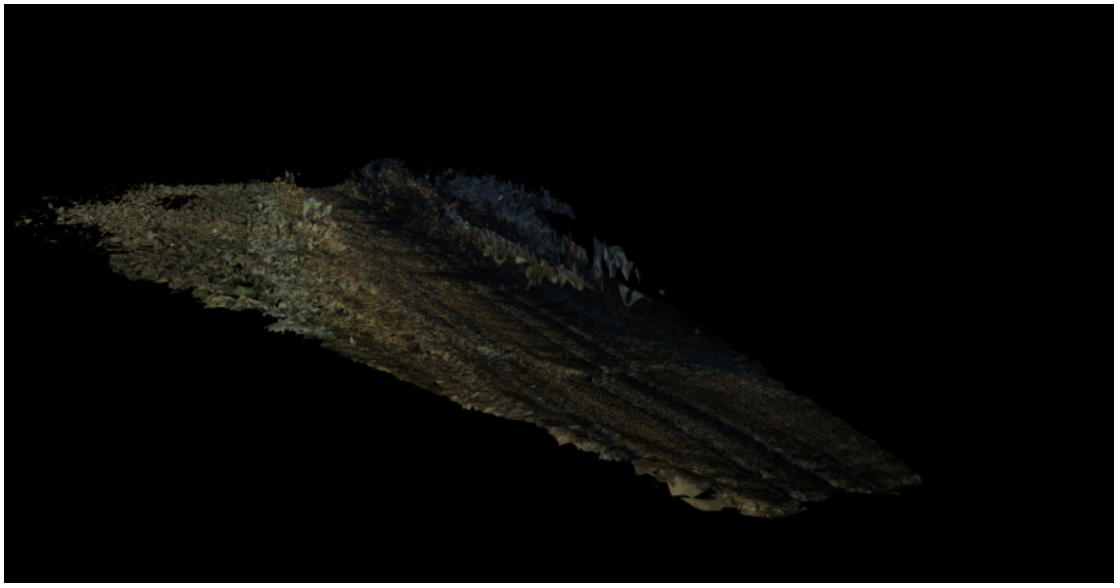


FIGURE B.35: The colorized mesh resulting from the p-rmls-small variation.

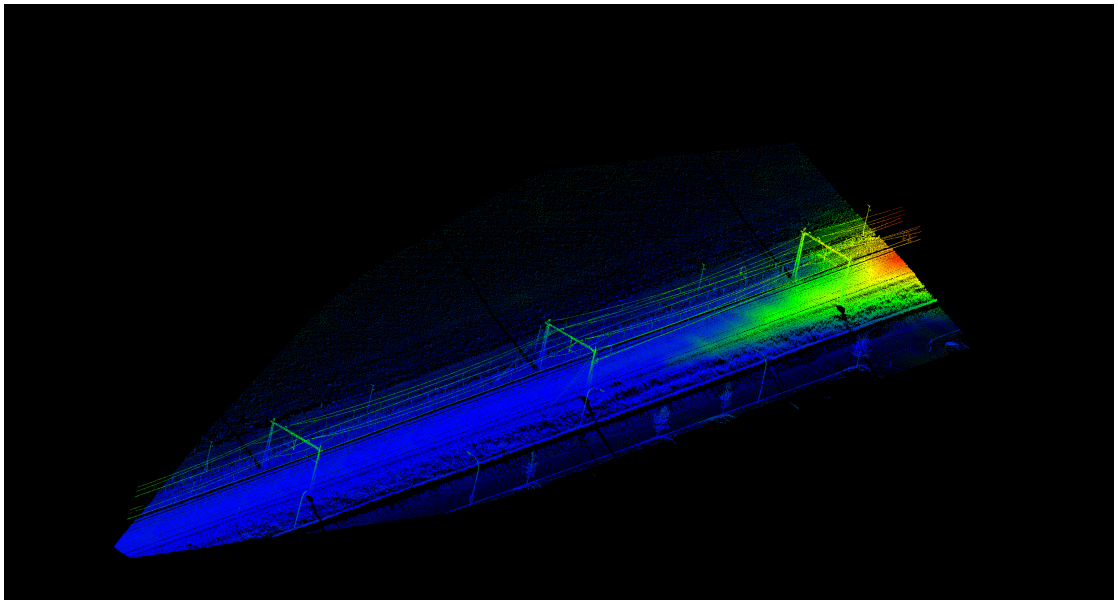


FIGURE B.36: The forward distances for the p-rmls-small variation mapped onto the ground truth point cloud. Ranging from 0 (blue) to 20.24 (red).

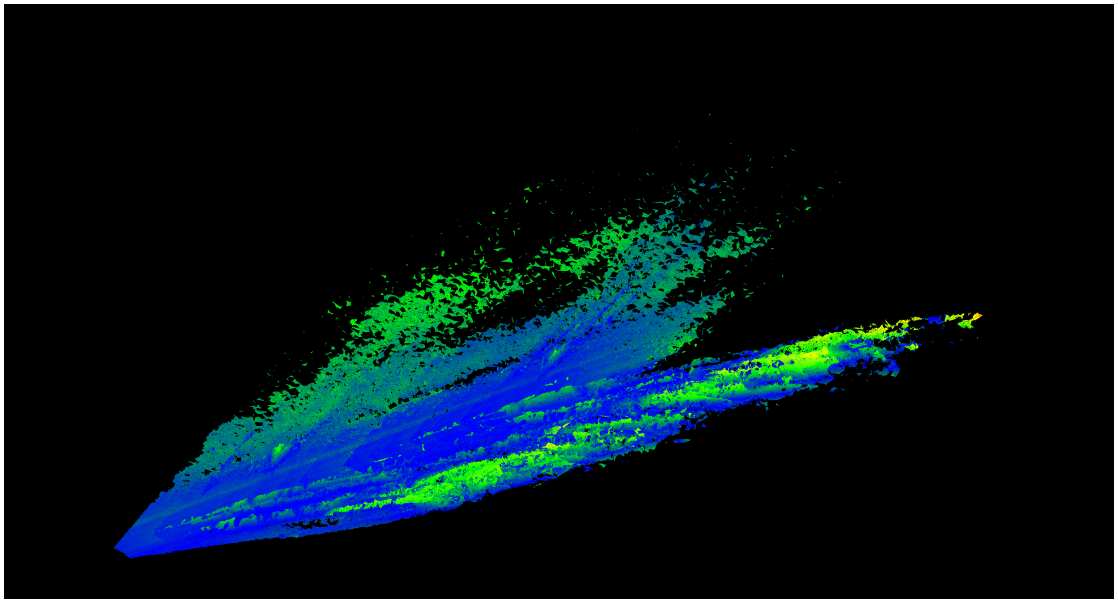


FIGURE B.37: The backward distances for the p-rmls-small variation. Ranging from 0 (blue) to 6.72 (red).

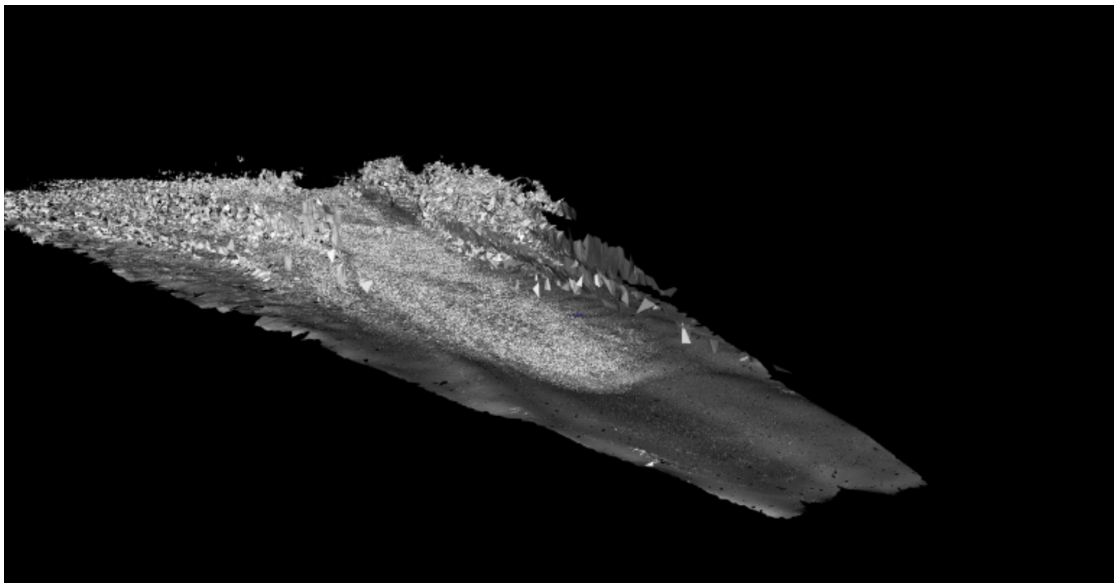


FIGURE B.38: The mesh resulting from the p-rmls-large variation.

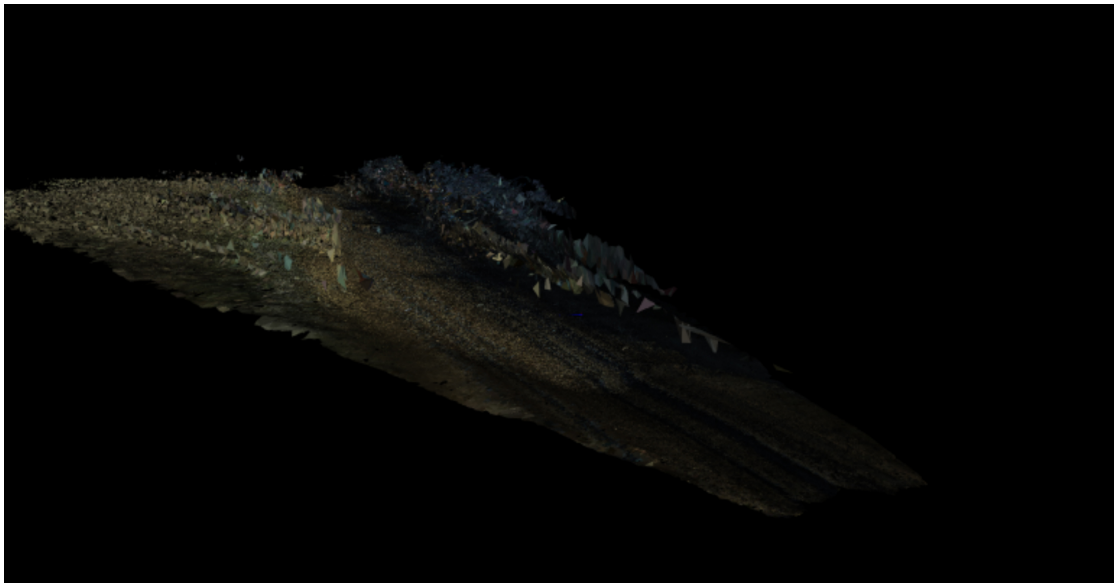


FIGURE B.39: The colorized mesh resulting from the p-rmls-large variation.

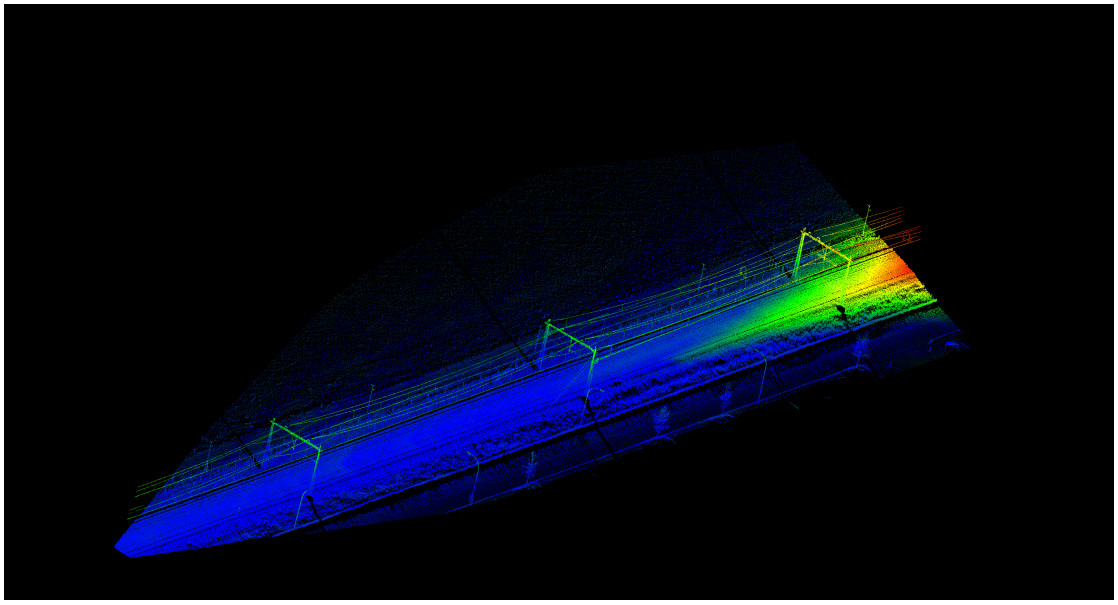


FIGURE B.40: The forward distances for the p-rmls-large variation mapped onto the ground truth point cloud. Ranging from 0 (blue) to 16.39 (red).

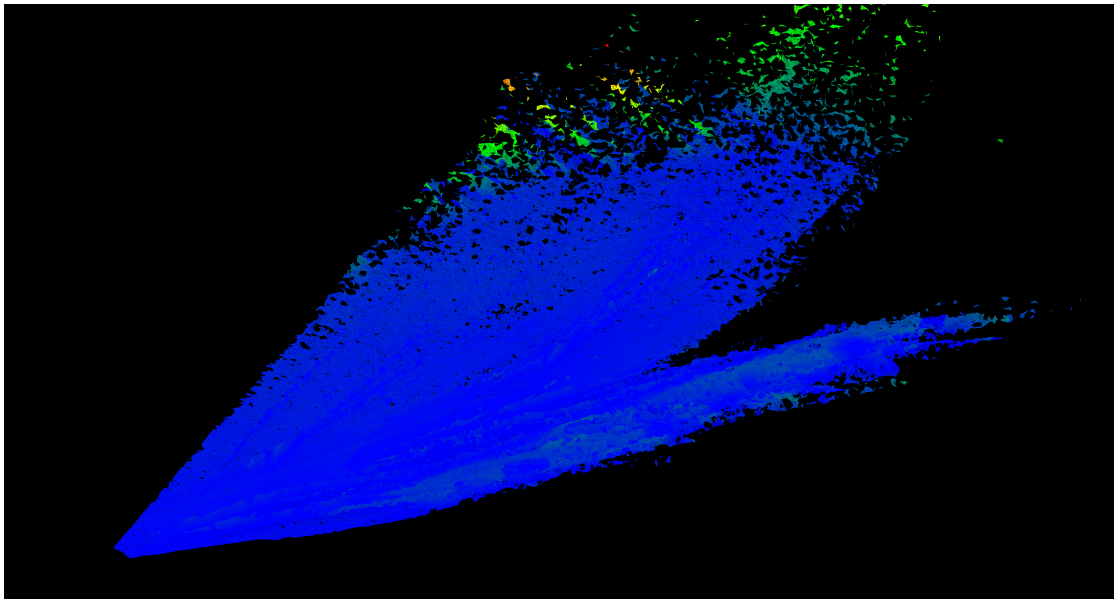


FIGURE B.41: The backward distances for the p-rmls-large variation. Ranging from 0 (blue) to 38.76 (red).

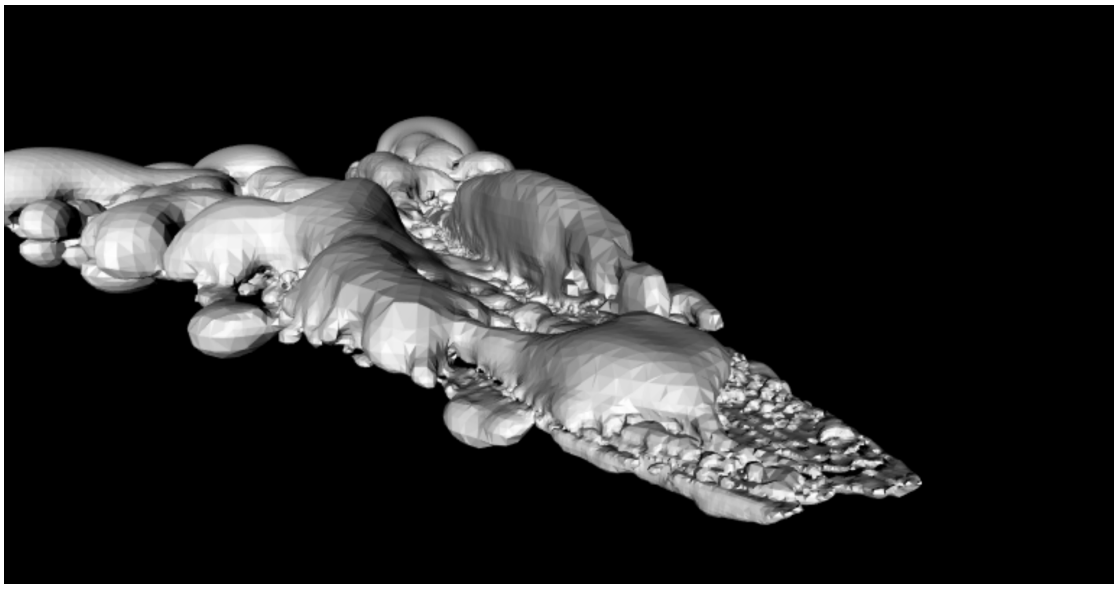


FIGURE B.42: The mesh resulting from the s-poisson-depth variation.

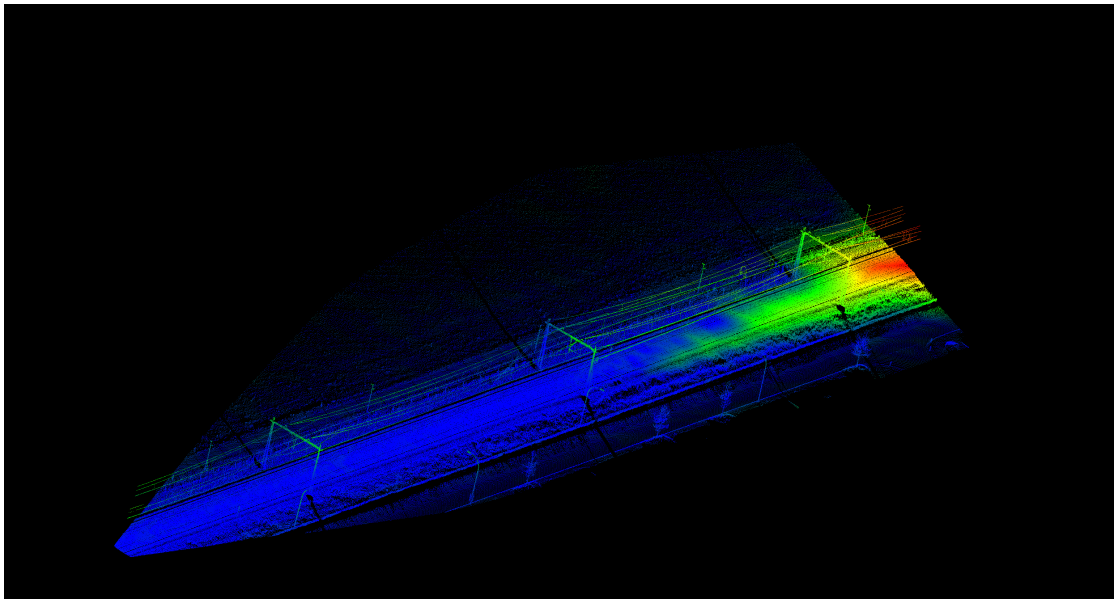


FIGURE B.43: The forward distances for the s-poisson-depth variation mapped onto the ground truth point cloud. Ranging from 0 (blue) to 17.48 (red).

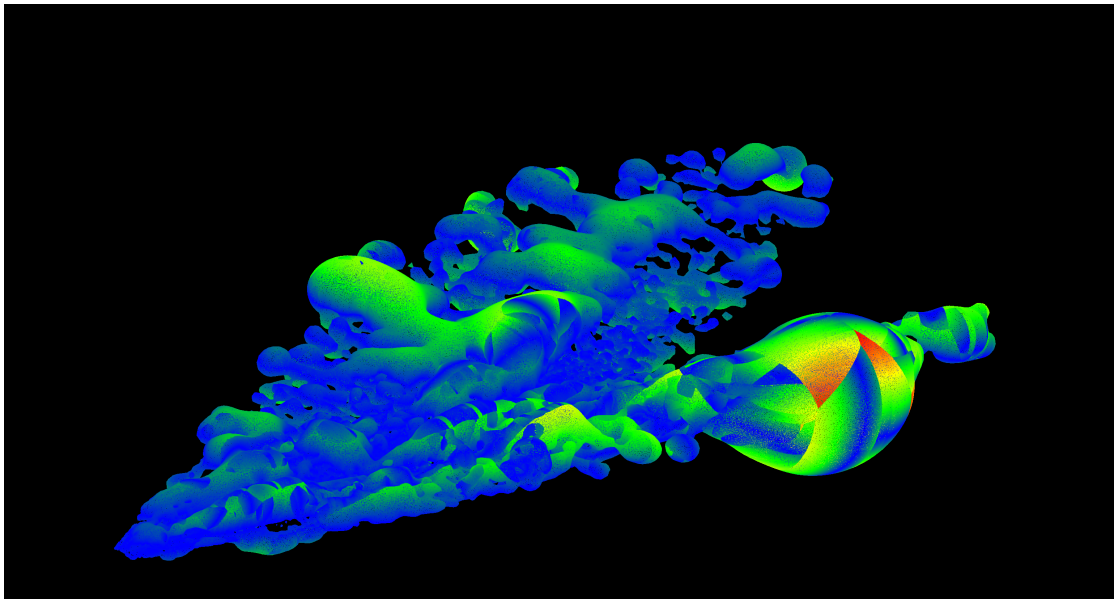


FIGURE B.44: The backward distances for the s-poisson-depth variation. Ranging from 0 (blue) to 13.65 (red).

Appendix C

Additional Evaluation Results

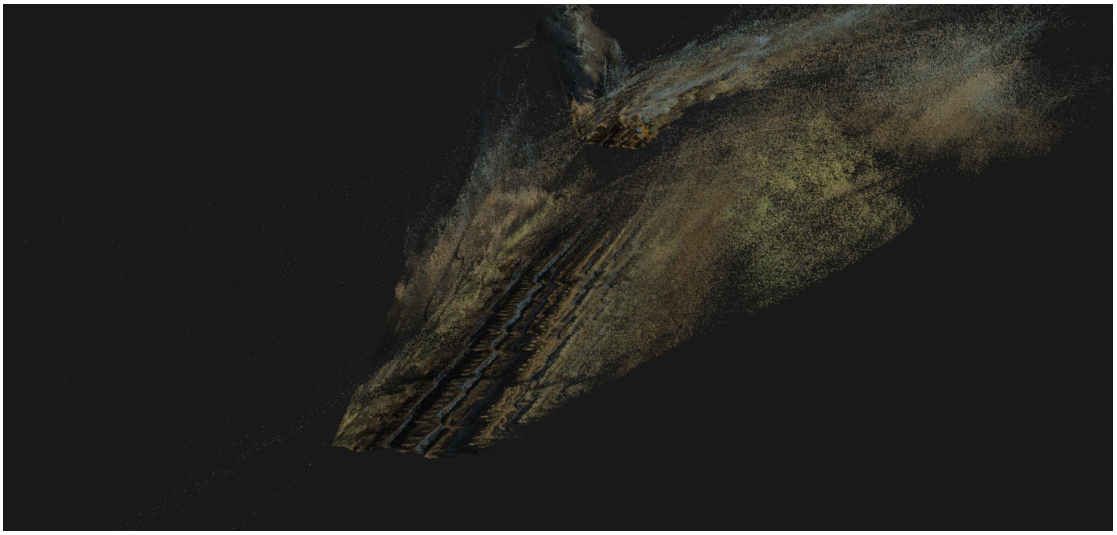


FIGURE C.1: The intermediate point cloud after triangulation in the additional evaluation. A large amount of noise and a gap in the reconstruction are clearly visible.

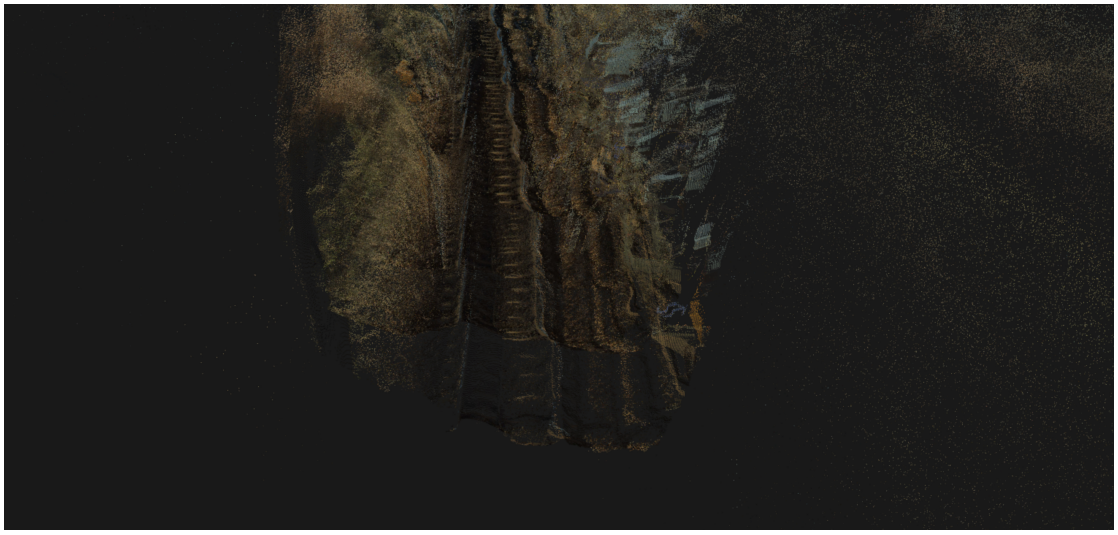


FIGURE C.2: The intermediate point cloud after triangulation in the additional evaluation. This view shows the part of the reconstruction after the gap. Parts of the passing train (the NS logo and yellow and gray parts) can be seen on the right side of the railway, but the train is not completely visible.

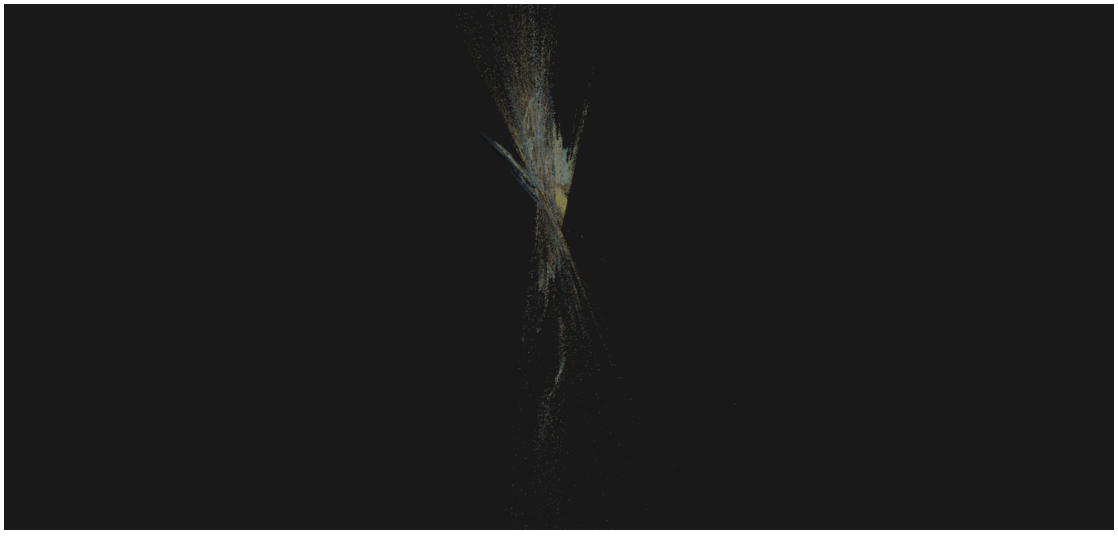


FIGURE C.3: Top-down view of the intermediate point cloud after triangulation in the additional evaluation. The starting point of the camera was approximately in the center of this image. Clearly, a lot of points are located behind the camera.

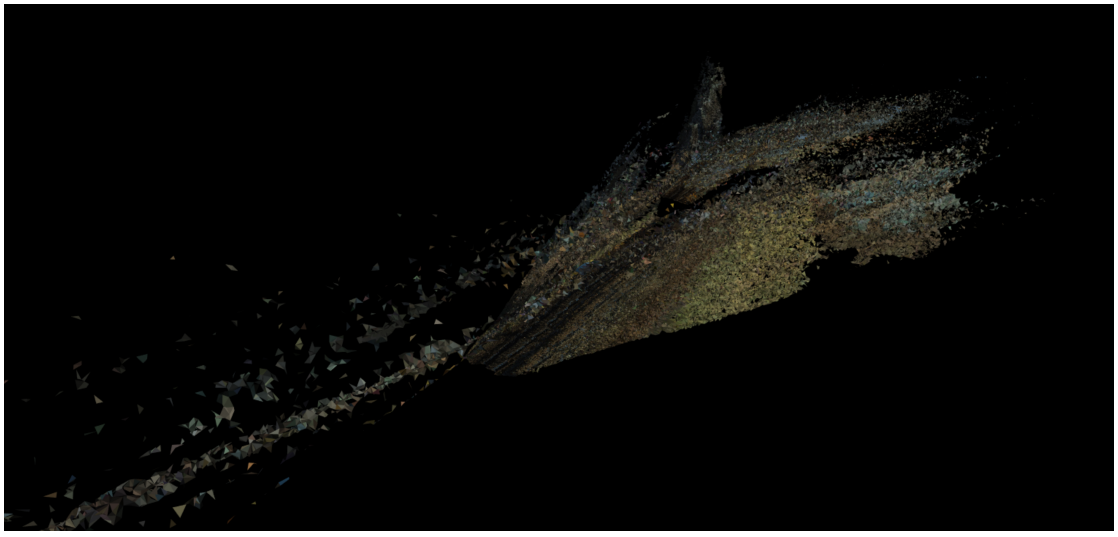


FIGURE C.4: The mesh resulting from the additional evaluation.

Bibliography

[1] Digitaal Schouwen: buiten = binnen. https://www.prorail.nl/nieuws/digitaal-schouwen-buiten-binnen, 2016. Accessed: 2016-09-26.

[2] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[3] Denis Fortun, Patrick Bouthemy, and Charles Kervrann. Optical flow modeling and computation: a survey. *Computer Vision and Image Understanding*, 134:1–21, 2015.

[4] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[5] Middlebury optical flow evaluation. http://vision.middlebury.edu/flow/, 2011. Accessed: 2016-09-20.

[6] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision*, pages 611–625. Springer, 2012.

[7] MPI Sintel Flow Dataset. http://sintel.is.tue.mpg.de/, 2015. Accessed: 2016-09-20.

[8] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3061–3070, 2015.

[9] KITTI Vision Benchmark Suite. http://www.cvlibs.net/datasets/kitti/, 2015. Accessed: 2016-09-20.

[10] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *European conference on computer vision*, pages 25–36. Springer, 2004.

[11] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[12] Victor Lempitsky, Stefan Roth, and Carsten Rother. Fusionflow: Discrete-continuous optimization for optical flow estimation. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[13] A Wedel, T Pock, J Braun, U Franke, and D Cremers. Duality tv-l1 flow with fundamental matrix prior. In *2008 23rd International Conference Image and Vision Computing New Zealand*, pages 1–6. IEEE, 2008.

[14] Andreas Wedel, Thomas Pock, Christopher Zach, Horst Bischof, and Daniel Cremers. An improved algorithm for tv-l 1 optical flow. In *Statistical and Geometrical Approaches to Visual Motion Analysis*, pages 23–45. Springer, 2009.

[15] Deqing Sun, Stefan Roth, and Michael J Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, 106(2):115–137, 2014.

[16] James R Bergen, Patrick Anandan, Keith J Hanna, and Rajesh Hingorani. Hierarchical model-based motion estimation. In *European conference on computer vision*, pages 237–252. Springer, 1992.

[17] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Joint Pattern Recognition Symposium*, pages 214–223. Springer, 2007.

[18] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):500–513, 2011.

[19] Zhuoyuan Chen, Hailin Jin, Zhe Lin, Scott Cohen, and Ying Wu. Large displacement optical flow from nearest neighbor fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2443–2450, 2013.

[20] Linchao Bao, Qingxiong Yang, and Hailin Jin. Fast edge-preserving patchmatch for large displacement optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3534–3541, 2014.

[21] Jiangbo Lu, Hongsheng Yang, Dongbo Min, and Minh N Do. Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1854–1861, 2013.

[22] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG*, 28(3):24, 2009.

[23] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Citeseer, 1988.

[24] Tony Lindeberg. Detecting salient blob-like image structures and their scales with a scale-space primal sketch: a method for focus-of-attention. *International Journal of Computer Vision*, 11(3):283–318, 1993.

[25] Geert Willems, Tinne Tuytelaars, and Luc Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *European conference on computer vision*, pages 650–663. Springer, 2008.

[26] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. IEEE, 1999.

[27] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer vision and image understanding*, 110(3):346–359, 2008.

[28] Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and trends® in computer graphics and vision*, 3(3):177–280, 2008.

[29] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

[30] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[31] Ethan Eade and Tom Drummond. Scalable monocular slam. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 469–476. IEEE, 2006.

[32] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, MA Fischler and O. Firschein, eds*, pages 61–62, 1987.

[33] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):756–770, 2004.

[34] Olivier Saurer, Pascal Vasseur, Rémi Boutteau, Cédric Demonceaux, Marc Pollefeys, and Friedrich Fraundorfer. Homography based egomotion estimation with a common direction. *IEEE transactions on pattern analysis and machine intelligence*, 2016.

[35] Paul Beardsley, Andrew Zisserman, and David Murray. Navigation using affine structure from motion. *Computer VisionECCV'94*, pages 85–96, 1994.

[36] Paul A Beardsley, Andrew Zisserman, and David W Murray. Sequential updating of projective and affine structure from motion. *International journal of computer vision*, 23(3):235–259, 1997.

[37] Richard I Hartley and Peter Sturm. Triangulation. *Computer vision and image understanding*, 68(2):146–157, 1997.

[38] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927–941, 2008.

[39] Radu Bogdan Rusu, Nico Blodow, Zoltan Marton, Alina Soos, and Michael Beetz. Towards 3d object maps for autonomous household robots. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3191–3198. IEEE, 2007.

[40] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[41] Philip HS Torr and Andrew Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

[42] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

[43] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. *Surface reconstruction from unorganized points*, volume 26. ACM, 1992.

[44] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[45] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011.

[46] Zoltan Csaba Marton, Radu Bogdan Rusu, and Michael Beetz. On fast surface reconstruction methods for large and noisy point clouds. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3218–3223. IEEE, 2009.

[47] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76. ACM, 2001.

[48] Ruosi Li, Lu Liu, Ly Phan, Sasakthi Abeysinghe, Cindy Grimm, and Tao Ju. Polygonizing extremal surfaces with manifold guarantees. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling*, pages 189–194. ACM, 2010.

[49] Marc Pollefeys, David Nistér, J-M Frahm, Amir Akbarzadeh, Philippos Mordohai, Brian Clipp, Chris Engels, David Gallup, S-J Kim, Paul Merrell, et al. Detailed

real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, 78(2-3):143–167, 2008.

[50] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[51] Richard A Newcombe and Andrew J Davison. Live dense reconstruction with a single moving camera. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1498–1505. IEEE, 2010.

[52] MWM Gamini Dissanayake, Paul Newman, Steve Clark, Hugh F Durrant-Whyte, and Michael Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on robotics and automation*, 17(3):229–241, 2001.

[53] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Aaai/iaai*, pages 593–598, 2002.

[54] Andrew J Davison. Real-time simultaneous localisation and mapping with a single camera. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1403–1410. IEEE, 2003.

[55] Nicolas Aspert, Diego Santa Cruz, and Touradj Ebrahimi. Mesh: measuring errors between surfaces using the hausdorff distance. In *ICME (1)*, pages 705–708, 2002.

[56] Cloud-to-Mesh Distance (CloudCompare). http://www.cloudcompare.org/doc/wiki/index.php?title=Cloud-to-Mesh_Distance, 2015. Accessed: 2017-03-22.

[57] Cloud-to-Cloud Distance (CloudCompare). http://www.cloudcompare.org/doc/wiki/index.php?title=Cloud-to-Cloud_Distance, 2015. Accessed: 2017-03-22.

[58] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

[59] CUDA implementation of the paper "Fast Edge-Preserving PatchMatch for Large Displacement Optical Flow". https://github.com/linchaobao/EPPM, 2016. Accessed: 2017-03-08.

[60] Simple triangulation with OpenCV from Hartley & Zisserman. http://www.morethantechnical.com/2012/01/04/simple-triangulation-with-opencv-from-harley-zisserman-w-code/, 2012. Accessed: 2017-03-08.

[61] Mesh - Sample points (CloudCompare). http://www.cloudcompare.org/doc/wiki/index.php?title=Mesh%5CSample_points, 2015. Accessed: 2017-03-22.

[62] Boris Delaunay. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2, 1934.

[63] Haim Permuter, Joseph Francos, and Ian Jermyn. A study of gaussian mixture models of color and texture features for image classification and segmentation. *Pattern Recognition*, 39(4):695–706, 2006.

[64] Max Mignotte. Segmentation by fusion of histogram-based k-means clusters in different color spaces. *IEEE Transactions on image processing*, 17(5):780–787, 2008.

[65] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(3):29, 2013.

[66] Matthew Bolitho, Michael Kazhdan, Randal Burns, and Hugues Hoppe. Parallel poisson surface reconstruction. *Advances in Visual Computing*, pages 678–689, 2009.