Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2018

Road Surface Preview Estimation using a Monocular Camera

Marcus Ekström

Master of Science Thesis in Electrical Engineering

Road Surface Preview Estimation using a Monocular Camera

Marcus Ekström

LiTH-ISY-EX--18/5173--SE

Supervisor: **Abdelrahman Eldesokey**
ISY, Linköpings universitet
Karl Schärlund
Veoneer AB
Daniel Wrang
Veoneer AB

Examiner: **Lasse Alfredsson**
ISY, Linköpings universitet

Computer Vision Laboratory
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Copyright © 2018 Marcus Ekström

Abstract

Recently, sensors such as radars and cameras have been widely used in automotives, especially in Advanced Driver-Assistance Systems (ADAS), to collect information about the vehicle's surroundings. Stereo cameras are very popular as they could be used passively to construct a 3D representation of the scene in front of the car. This allowed the development of several ADAS algorithms that need 3D information to perform their tasks. One interesting application is Road Surface Preview (RSP) where the task is to estimate the road height along the future path of the vehicle. An active suspension control unit can then use this information to regulate the suspension, improving driving comfort, extending the durabilitiy of the vehicle and warning the driver about potential risks on the road surface. Stereo cameras have been successfully used in RSP and have demonstrated very good performance. However, the main disadvantages of stereo cameras are their high production cost and high power consumption. This limits installing several ADAS features in economy-class vehicles. A less expensive alternative are monocular cameras which have a significantly lower cost and power consumption. Therefore, this thesis investigates the possibility of solving the Road Surface Preview task using a monocular camera. We try two different approaches: structure-from-motion and Convolutional Neural Networks. The proposed methods are evaluated against the stereo-based system. Experiments show that both structure-from-motion and CNNs have a good potential for solving the problem, but they are not yet reliable enough to be a complete solution to the RSP task and be used in an active suspension control unit.

Acknowledgments

I would like to thank Veoneer for the opportunity to work on this Master Thesis. Thank you to all of you who supported and helped me by giving advice and answering questions. I want to direct a special thank you to my two supervisors at Veoneer, Daniel Wrang and Karl Schärlund for all their contributions to this Master Thesis. I also want to say thank you to my third supervisor Abdelrahman Eldesokey for all the positive encouragement, assistance and knowledge passed on to me during the thesis. Finally, I would like to thank my examiner Lasse Alfredsson for his valuable feedback and input on the report and the thesis work.

Contents

List of Figures

List of Tables

1 Introduction

Recently, sensors such as radars and cameras have been widely used in automotives, especially in Advanced Driver-Assistance Systems (ADAS), to collect information about the vehicle's surroundings. A common type of these sensors is the stereo cameras which consist of two cameras simulating the biological vision system. These stereo cameras enable constructing a 3D representation of the environment in front of the vehicle which opens up for several applications, e.g. obstacle detection, collision avoidance and Road Surface Preview (RSP). In RSP, the road surface ahead of the vehicle is estimated based on the 3D representation from the stereo camera. Further, this estimation is used to regulate and control the suspension of the vehicle which contributes to improving driving comfort, increasing the durability of the vehicle, and warning the driver of potential risks on the road surface.

Stereo cameras have demonstrated a great success in different ADAS, especially in the task of Road Surface Preview. However, stereo cameras are relatively expensive and they consume more power than monocular cameras which leads to a higher production cost. Consequently, more attention has been given to developing ADAS using monocular cameras. This would allow integrating these systems in a wider range of economical vehicles extending their durability and providing safety and comfort for more people.

1.1 Motivation

In this thesis, we aim to investigate how monocular cameras would perform on the task of Road Surface Preview (RSP) compared to stereo cameras. This work was conducted at Veoneer and the proposed approach is evaluated against the

existing stereo-based RSP developed by Veoneer. A monocular-based RSP would open up for several comfort and safety features in vehicles with low-price tags. Furthermore, it would help making those vehicles more durable and reliable for users. Finally, this work would show some indications about the potential of utilizing monocular cameras as a cheaper replacement for stereo cameras.

1.2 Problem Formulation

The main strength of stereo cameras is the availability of depth information which allows for an accurate estimation of the road profile. On the other hand, monocular cameras lacks depth information which imposes an additional challenge to estimate the depth. Several approaches have been proposed to estimate the depth from monocular images. However, for the RSP, a high precision is required to perform the task efficiently. In this thesis, we investigate two different approaches for estimating the depth and 3D information. First, we propose a structure-from-motion based approach to construct a 3D point cloud of the road surface from a sequence of images. This point cloud is then used to provide depth information for the RSP algorithm. Secondly, we examine the possibility of using a Convolutional Neural Network to predict the depth from a single monocular image. The estimation of the depth is then used in a similar fashion to how a disparity image would be used in a stereo camera.

1.3 Research Questions

In this thesis, we aim to answer the following questions:

- Is it possible to solve the Road Surface Preview problem using a monocular vision system with good enough accuracy to be used in an active suspension control unit?
- How well does a Structure from Motion approach perform on this task?
- What is the performance of a Convolutional Neural Networks based approach compared to the one based on Structure from Motion?

1.4 Limitations

Due to time constraints, we used the existing framework for RSP developed at Veoneer and we only altered the depth estimation component. Due to the limited computational resources, we tested only one CNN architecture that was proven to perform well on the task of depth estimation while requiring low computational resources.

1.5 Outline of the Thesis

We start with an introduction and a description of the problem in Chapter 1. Next, we describe the RSP algorithm developed at Veoneer as a baseline for this work in Chapter 2. In Chapter 3 and Chapter 4, the structure-from-motion approach and the CNN-based approach are described respectively. Both these chapters contain a description of relevant theory, experiments and a discussion related to the performance of the described method. Finally Chapter 5 provides conclusion remarks as well as some suggestions for future improvement.

2

Road Surface Preview

The Road Surface Preview (RSP) problem that is investigated in this thesis has been solved using a stereo system at Veoneer. This algorithm estimates the elevation of the road near the predicted wheel paths in front of the vehicle. An overview of the existing system is shown in Figure 2.1.

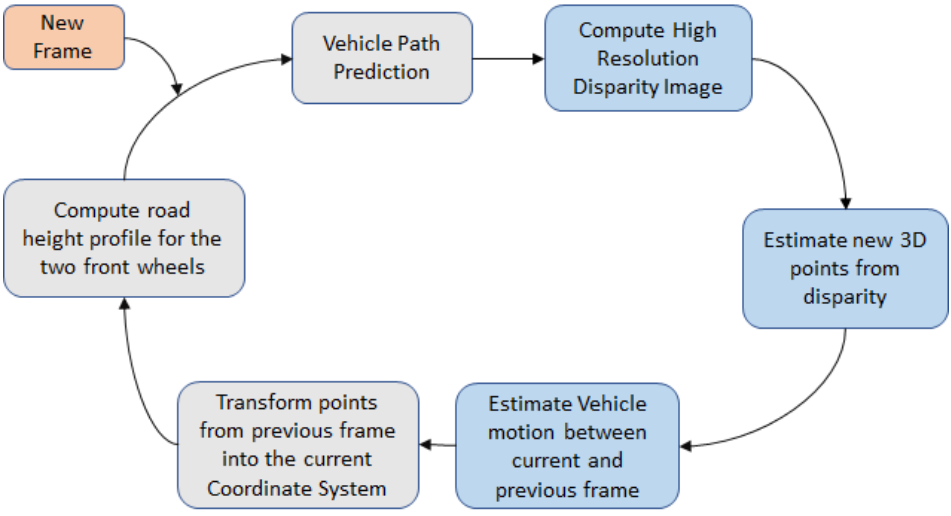


Figure 2.1: *An Overview of the existing stereo algorithm for Road Surface Preview developed at Veoneer. The blue components will be modified in our work.*

First, a path predictor estimates the future position of the front wheels with respect to the wheels-axis coordinate system. Afterwards, images from the stereo cameras are used to compute a high resolution disparity map at these positions in the camera coordinate system. The disparity map is used to generate a 3D point cloud describing the road surface. Additionally, the 3D point cloud from the previous frame is transformed to the current frame using an estimated transformation. This increases the density of the point cloud and contributes to a more robust estimation of the road profile. Further, the point cloud is used to compute the description of the road profile including the height. The suspension system of the vehicle is then adjusted according to that estimated height. An example of the output is presented in Figure 2.2. The two curves describe the height of the road in front of the left (top) and right front wheel.

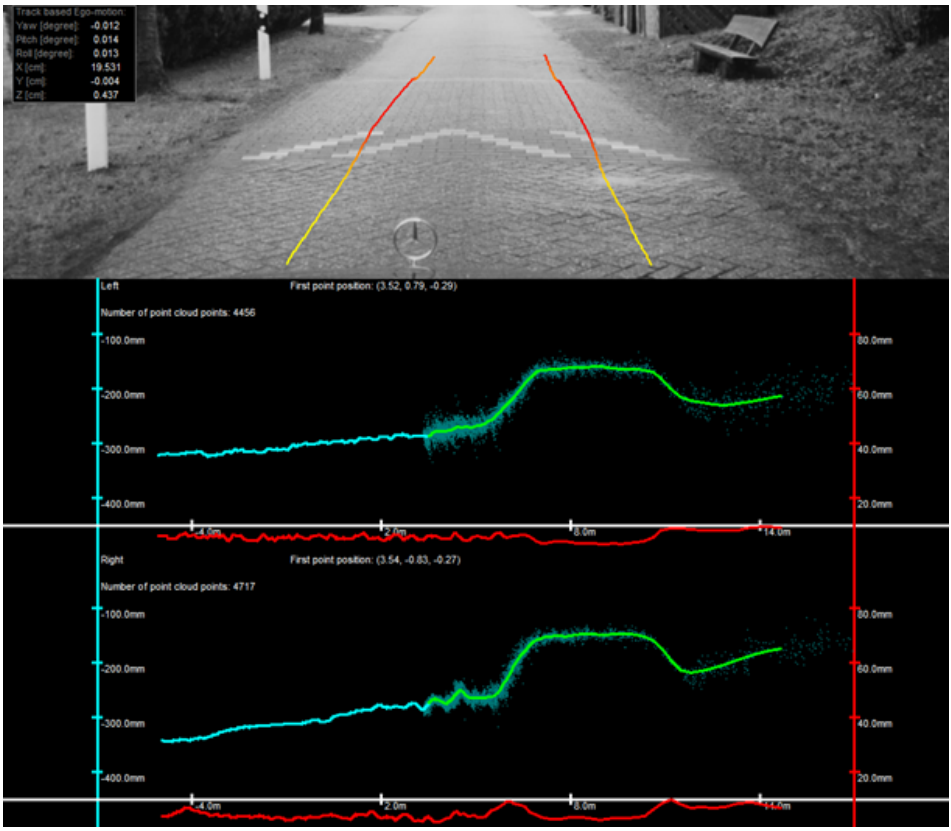


Figure 2.2: *An example of the output generated in the Road Surface Preview. The top image is the camera image with the output as an overlay. Here, the color scale yellow to red represents an increase in road height. The bottom image describes the output signal from the RSP application that is used in the suspension control. The top and bottom signal describes the left and the right wheel path respectively.*

2.1 Vehicle Path Prediction

The RSP algorithm road height estimation is used to regulate the suspension. Therefore the height estimation of the road can be limited to only along the path of the front wheels. The paths are predicted using several vehicle parameters, a motion model developed at Veoneer and the vehicle's motion between frames. An example of path prediction is seen in Figure 2.3 together with the generated output.

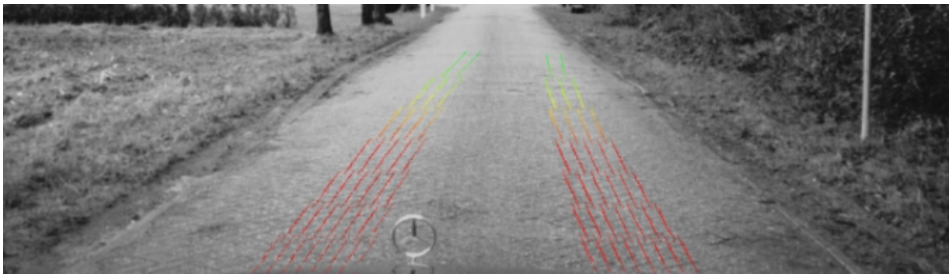


Figure 2.3: *Road Surface Preview Path Prediction*

2.2 Detection and Tracking of 3D points

A stereo camera is mounted in the windshield of the vehicle and is able to capture the scene at the same time instance from two different viewpoints. Here, the relative orientation and placement of the two camera objectives in the stereo camera with respect to each other is known. The two images captured by the stereo camera are used to estimate how each pixel has moved between the two images, which is denoted as disparity. This estimation involves finding matching points, which is further explained in Section 3.1.2. Once a pair of corresponding or matching image points has been found they can be treated as projections of the same 3D point. Triangulation methods described in Section 3.1.3 is used to find an estimate of this 3D point. A large amount of 3D points are computed in this step, which form a point cloud describing the scene in the current frame. Note that for the RSP task, the 3D points are calculated only for the regions along the predicted path from the previous step. To densify the point cloud, the point from the previous frame is transformed to the current frame using a transformation estimated from the vehicle's motion.

2.3 Computing the Road Profile

The point cloud from the previous step is merged or compressed into 74 points for each wheel describing the elevation of the road in front of the wheel. These 74 points are computed from the point cloud in an interval of four to seventeen meters in front of the front axel, giving one point every 17,5cm. Furthermore, the

set of points describing the path in the previous frame is transformed into the coordinate system used in the current frame according to the vehicle's estimated motion. An assumption is made that points closer to the vehicle are more accurate, and therefore should be used while others are discarded. This final resulting set of 74 points describe the road profile in the current frame.

2.4 Evaluation

In this section we present the evaluation data set and the evaluation metric that is used to compare and evaluate the performance of the two proposed methods.

2.4.1 Evaluation Dataset

For evaluation, a dataset is selected containing 78 recorded sequences, each with 660 camera frames collected at 22Hz. All the data is recorded by a stereo camera, mounted on a car and supplied by Veoneer. The set of sequences is evenly distributed over the set of different road types , including highway, city and rural roads. Furthermore, the set of sequences is chosen as balanced as possible with respect to weather conditions and time of day. Example images from the dataset are seen in Figure 2.4.

(a) Day *(b) Night*

(c) Rain *(d) Snow*

Figure 2.4: *Example images from the evaluation dataset.*

2.4.2 Evaluation Metric

To evaluate the performance relative to the existing stereo-based RSP, we use the Mean Square Error (MSE) defined as

$$\text{MSE: } \frac{1}{T}\sum_{i=1}^{T}(M_z(i) - S_z(i))^2 \tag{2.1}$$

where T is the total number of points estimated in the path, summed over the whole evaluation data set. $M_z(i)$ is the z-coordinate for point i estimated by one of our proposed monocular methods and $S_z(i)$ is the z-coordinate for the same point computed with the stereo-based RSP. The MSE is measured in meters.

The stereo RSP is a verified solution that is deployed in vehicles today and can be used to regulate the suspension, which makes the stereo RSP suitable for evaluation of our proposed methods.

3

Structure From Motion Approach

Strucute from Motion is the process of extracting a 3D representation of a scene from a set of images. These images are captured with a moving camera, where each image is captured at a distinct viewpoint. Our proposed method utilizes this approach to describe the scene in front of the moving vehicle and use this description in the Road Surface Preview Algorithm. An overview of the complete system is presented in Figure 3.1.

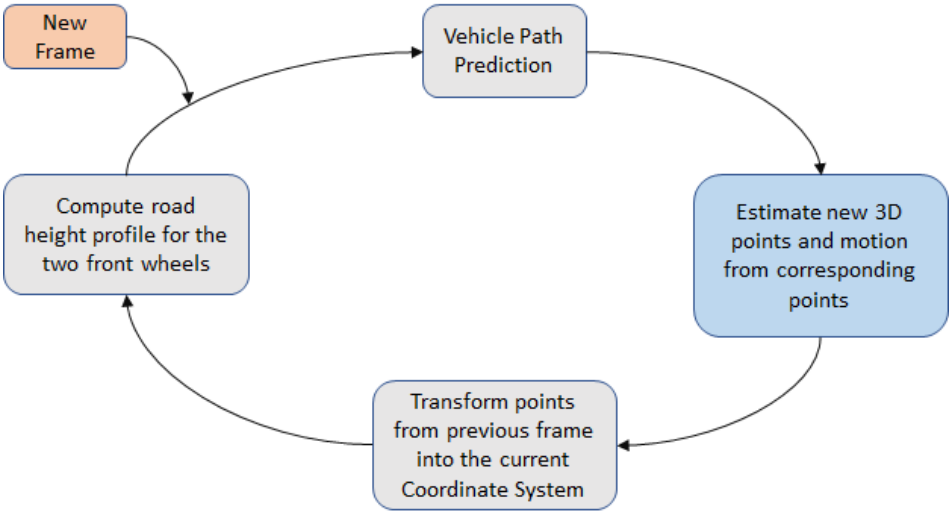


Figure 3.1: *Overview of the complete system used in the proposed Structure from Motion approach.*

3.1 Theory

Structure from Motion is a commonly used framework to perform 3D reconstruction of a scene from a set of images, captured at distinct viewpoints. Features are extracted in an image and then searched for in the feature set extracted in another image. Matching image points can then be used to perform the 3D reconstruction. In the following sections, the different parts of the structure from motion framework is described. For a more detailed and complete description on the topic of structure from motion, see Hartley and Zisserman [16].

3.1.1 Feature Point Detector

In the field of structure from motion, a set of features or image points are extracted to be matched with other features in another frame. To find corresponding features is a problem described in Section 3.1.2. However, for this to be possible the position of the extracted features need to be chosen wisely. The structure tensor is a way to describe the local image region around an image point. The structure tensor $\mathbf{T}$ computed in point $\mathbf{p}$ for image I is defined as

$$\mathbf{T}(\mathbf{p}) = \sum \omega_2(\mathbf{x})[\nabla I](\mathbf{x})[\nabla I]^T(\mathbf{x}), \tag{3.1}$$

where $\mathbf{x}$ represents an offset from $\mathbf{p}$ and $\omega_2(\mathbf{x})$ is a low pass filter, typically Gaussian. The structure tensor $\mathbf{T}$ is a 2x2 matrix that is computed in every pixel in the image and it forms the foundation for feature extraction and locating good features.

A feature detector that is based on the structure tensor is the *Harris-Stephens* feature detector [15]. It utilizes the structure tensor as computed above to calculate a response measure denoted C_H as follows:

$$C_H(\mathbf{p}) = \det\mathbf{T} - k(\mathrm{trace}\mathbf{T})^2 = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2, \tag{3.2}$$

where λ_1, λ_2 are the two eigenvalues to the matrix $\mathbf{T}$ and k is an empirically determined constant, $k \in [0.04, 0.06]$. The measurement C_H is calculated for every pixel giving a response map C_H. This response map is thresholded using a predefined threshold τ, where $C_H > \tau$ leaves points with high responses and they define the Harris feature points.

3.1.2 Descriptors and Matching

A descriptor is an n-dimensional vector representing or describing the local image region around its connected feature. For each feature point extracted in the image, its descriptor is computed. To find correspondence between features across different frames is then a matter of finding the most similar descriptor. Similarity between descriptors is computed as the distance between the two n-dimensional descriptors, in the n-dimensional vector space they lie in. Matching descriptors are the pair with the shortest distance of all descriptors, below a threshold to exclude poor matches. Different distance measures can be used depending on the application and the type of descriptor. For the BRIEF descrip-

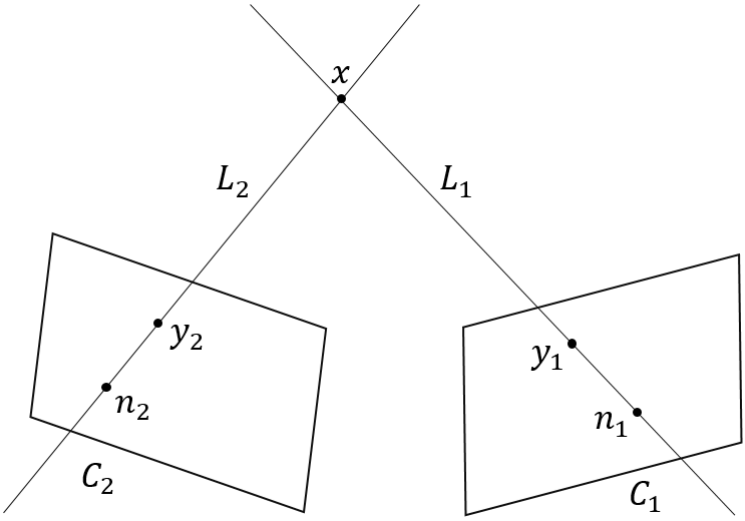


Figure 3.2: *Triangulation of 3D point* $\mathbf{x}$ *from two corresponding image points* $\mathbf{y}_1$ *and* $\mathbf{y}_2$.

tor we use in our work the hamming distance is typically used. The hamming distance between two binary descriptors $\mathbf{a}, \mathbf{b}$ is computed as the number of bits that are different, more formally expressed as the population count of ones in $XOR(\mathbf{a}, \mathbf{b})$. The BRIEF descriptor itself will not be described in more detail in this thesis, for more details see [3].

3.1.3 Triangulation

A pair of corresponding image points can be treated as projections of the same 3D point. Given that the two images are captured at two distinct viewpoints, where the relative orientation and translation of the camera is known between the two views, the coordinates of the 3D point can be found through triangulation. This is illustrated in Figure 3.2 for cameras $\mathbf{C}_1$ and $\mathbf{C}_2$.

For each image point $\mathbf{y}_x$ there is a projection line $\mathbf{L}_x$ that intersects the image point and the camera center $\mathbf{n}_x$. For corresponding points from the two cameras, their projection lines intersect in the projected 3D point. The camera center $\mathbf{n}$ is defined as the zero vector to the camera matrix, the vector that satisfies

$$\mathbf{Cn} = 0. \tag{3.3}$$

With the two sets of camera center and image point the two projection lines can be formed. For the general and real case, two image points are not fully corresponding points and therefore the projection lines do not intersect. Measurement noise and other form of noise requires a more robust estimation procedure that revolves around solving an optimization problem rather than finding the intersection point.

Mid Point Method

The mid point method is a geometric solution to the triangulation problem. From the two corresponding points $\mathbf{y}_1$ and $\mathbf{y}_2$ and the corresponding camera pose for the two cameras two 3D points $\mathbf{x}_1$ and $\mathbf{x}_2$ can be found. The points $\mathbf{x}_1$ and $\mathbf{n}_1$ are distinct and both lie on the projection line L_1, while the points $\mathbf{x}_2$ and $\mathbf{n}_2$ are distinct and both lie on the other projection line L_2. The two projection lines can therefore be parameterized with parameters s and t in cartesian coordinates as

$$\begin{aligned} \mathbf{L}_1(s) &= s\mathbf{x}_1 + (1-s)\mathbf{n}_1 \\ \mathbf{L}_2(t) &= t\mathbf{x}_2 + (1-t)\mathbf{n}_2 \end{aligned}. \tag{3.4}$$

To determine the 3D point $\mathbf{x}$ that corresponds to the two image points an error function is defined as

$$\epsilon_1 = \|\mathbf{x}_1(s) - \mathbf{x}\|^2 + \|\mathbf{x}_2(t) - \mathbf{x}\|^2. \tag{3.5}$$

The triangulation problem can then be expressed as finding $\mathbf{x}$ together with s and t that minimize ϵ_1. The problem can however be reformulated based on that s and t must be chosen such that $\mathbf{x}_1(s)$ and $\mathbf{x}_2(t)$ are the two points on the two projection lines L_1 and L_2 that are as close together as possible. In other words the error function can be formulated as

$$\epsilon_2 = \|\mathbf{x}_1(s) - \mathbf{x}_2(t)\|^2 \tag{3.6}$$

and the optimization problem as finding s and t that minimize ϵ_2. The details for how to solve this optimization problem is described in [16].

3.1.4 Bundle Adjustment

The tracking of feature points often contain considerable amount of noise. Therefore a secondary optimization procedure called Bundle Adjustment is often used. This is a process of solving an optimization problem aiming to minimize the reprojection error over the set of 3D points P and all the camera poses. This error is denoted as ϵ_{BA} and it is defined as

$$\epsilon_{BA} = \sum_{k=1}^{m} \sum_{j=1}^{p} w_{kj} d_{pp}(\mathbf{y}_{kj}, (\mathbf{R}_k|\mathbf{t}_k)\mathbf{x}_j)^2. \tag{3.7}$$

Here, d_{pp} is a distance measure between image point $\mathbf{y}_{kj}$ and the projection of the 3D point $\mathbf{x}_j$ using the absolute camera pose matrix $(\mathbf{R}_k|\mathbf{t}_k)$. Furthermore, m is the total number of camera poses, p is the number of 3D points and w is a *visibility function* defined as

$$w_{kj} = \begin{cases} 1 & \text{if 3D point } \mathbf{x}_j \text{ is visible in image from camera } k \\ 0 & \text{Otherwise} \end{cases}. \tag{3.8}$$

3.1.5 Feature Point Refinement

Given a pair of points ($\mathbf{y}_1, \mathbf{y}_2$) from two images, feature point refinement aims to move one of the image points to produce a better match. In other words, the method tracks an image point $\mathbf{y}_1$ in a small region around $\mathbf{y}_2$ to find the point that minimize some error function ϵ. This can be seen as a description of the same problem solved by the KLT Tracker first proposed by Lucas and Kanade [25]. Here, a brief summary of the theory involved in the KLT tracker is given. The problem consist of finding the disparity vector $\mathbf{h}$, the displacement between two images, that minimizes some measure of the difference ϵ between $F(\mathbf{x}+\mathbf{h})$ and $G(\mathbf{x})$, where $F(\mathbf{x})$ and $G(\mathbf{x})$ are functions representing the pixel values in two different images at location $\mathbf{x}$. Commonly used error measures in the KLT tracker framework are

$$\begin{aligned} \mathcal{L}_1\text{- Norm: } \epsilon &= \sum_{\mathbf{x}\in\mathcal{R}} |F(\mathbf{x}+\mathbf{h}) - G(\mathbf{x})|, \\ \mathcal{L}_2\text{- Norm: } \epsilon &= \sqrt{\sum_{\mathbf{x}\in\mathcal{R}} [F(\mathbf{x}+\mathbf{h}) - G(\mathbf{x})]^2}. \end{aligned} \tag{3.9}$$

Here, $\mathcal{R}$ defines the search region. The implementation used in the thesis uses a variation of the $\mathcal{L}_2$ norm, formulated as

$$\epsilon = \sum_{\mathbf{x}\in\mathcal{R}} (F(\mathbf{x}+\mathbf{h}) - G(\mathbf{x}))^2. \tag{3.10}$$

Important to note is the approximation for the behaviour of $F(\mathbf{x})$, in the neighbourhood of $\mathbf{x}$ for small $\mathbf{h}$, that is expressed in [25] as

$$F(\mathbf{x}+\mathbf{h}) \approx F(\mathbf{x}) + \mathbf{h}F(\mathbf{x}). \tag{3.11}$$

To minimize ϵ with respect to $\mathbf{h}$, we set

$$\begin{aligned} &\frac{\partial \epsilon}{\partial \mathbf{h}} = \text{(Using Equation (3.11))} \\ =&\frac{\partial}{\partial \mathbf{h}} \sum_{\mathbf{x}} (F(\mathbf{x}) + \mathbf{h}F(\mathbf{x}) - G(\mathbf{x}))^2 \\ =&\sum_{\mathbf{x}} 2F(\mathbf{x})\,(F(\mathbf{x}) + \mathbf{h}F(\mathbf{x}) - G(\mathbf{x})) = 0, \end{aligned}$$

which leads to

$$\mathbf{h} = \frac{\sum_{\mathbf{x}} F(\mathbf{x})\,(G(\mathbf{x}) - F(\mathbf{x}))}{\sum_{\mathbf{x}} F(\mathbf{x})^2}. \tag{3.12}$$

With this estimate for the disparity vector $\mathbf{h}$, $F(\mathbf{x})$ can be moved by $\mathbf{h}$ and the procedure can be repeated, yielding an iterative refinement method for the estimation of $\mathbf{h}$. The resulting disparity vector $\mathbf{h}$ in turn describes the location of the new refined image point.

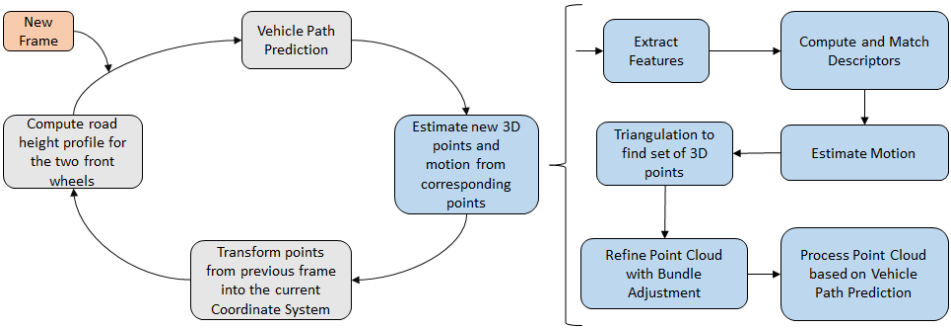


Figure 3.3: *Flowchart describing the structure from motion baseline.*

3.2 The Baseline

The baseline approach is to use the standard Structure from Motion framework to estimate the 3D description of the road surface. A flowchart describing the different steps in the baseline is presented in Figure 3.3.

First, a set of features are extracted in the current and previous frame as described in Section 3.1.1. Next, descriptors are computed and matching is performed using the hamming distance, as presented in Section 3.1.2. The pairs of corresponding image points together with signals from the vehicle are then used to estimate the motion between the two camera images, in a process often referred to as Visual Odometry. We then use the pairs of corresponding image points together with the estimated motion to form a set of 3D points P describing the scene in front of the camera by using the mid point triangulation method, implemented according to the description by S.Ramalingam *et. al* [27].

Without an accurate 3D representation of the scene, the output from the road surface preview application will not be valid or accurate enough. Therefore, Bundle Adjustment (see Section 3.1.4) optimized over only the set of 3D points P is used to improve the accuracy in the estimated 3D points. We optimize the Bundle Adjustment over only the 3D points P to reduce the complexity but still maintain good performance.

The set of 3D points are then divided into two sets, one for each front wheel. These sets contain the 3D points that lie close to the predicted wheel path found in the Vehicle Path Prediction, see Section 2.1. The estimated motion of the vehicle, and therefore also the camera that is mounted in the windshield, together with the two sets of 3D points are then used to compute the road profile.

3.3 Our Method

In the Structure from Motion framework, features are typically extracted from image regions with good structure to generate distinct and descriptive features

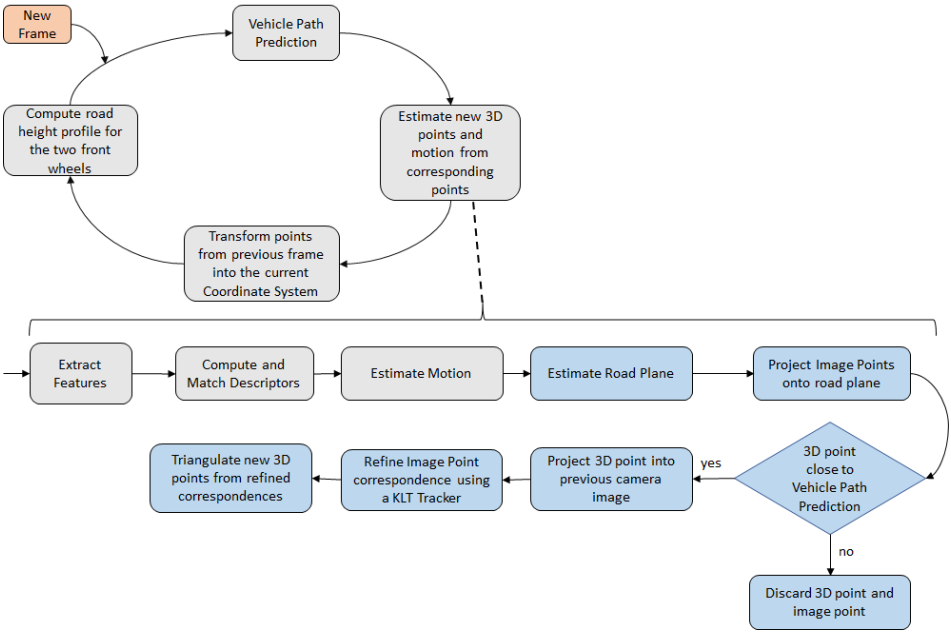


Figure 3.4: *Flowchart describing all the components in our structure from motion-based approach. The blue components are different compared to the structure from motion baseline.*

that are easier to track accurately. The road surface may not meet these qualities, which was compensated for in the baseline by extracting weaker features in these image regions. However, weaker features may result in inaccurate feature matching which in turn leads to poor accuracy in the road surface preview. As an alternative approach we propose to use the 3D information about the scene produced by the Structure from Motion framework to estimate the road plane, and then use this estimated road plane to improve the 3D representation of the road surface. A flowchart presenting all the components in our method is presented in Figure 3.4.

To form a new set of feature points, all image points are projected onto the estimated road plane. The output from the Vehicle Path Prediction component (see Section 2.1) is here used to discard an image point and a projected 3D point if the 3D point is not close to the estimated path for one of the front wheels. The remaining 3D points are transformed to the coordinate system of the camera in the previous frame and projected into the previous camera image to form pairs of corresponding image points.

The pair of corresponding image points are all still projections of points on the estimated road plane. Our proposed method aims to refine these correspondences by tracking the image points in the current frame in the previous camera image in a local region around the initial correspondences. The idea is that these refined

correspondences will describe 3D points on the actual road surface and no longer on the estimated road plane. The mid point triangulation method described in Section 3.1.3, implemented based on [27], is then once again used to find the 3D representation of the road surface from the new refined correspondences.

3.3.1 Road Plane Estimation

The process of estimating a plane can be described as fitting a model to a dataset. The resulting plane is represented by its normal vector $\mathbf{n} = (n_x, n_y, n_z)$. The method used is based on the work by Fanani *et al.* [8]. The plane is estimated from the extracted corresponding features, and the 3D points they describe. The plane with the largest inlier set is chosen as the final estimate.

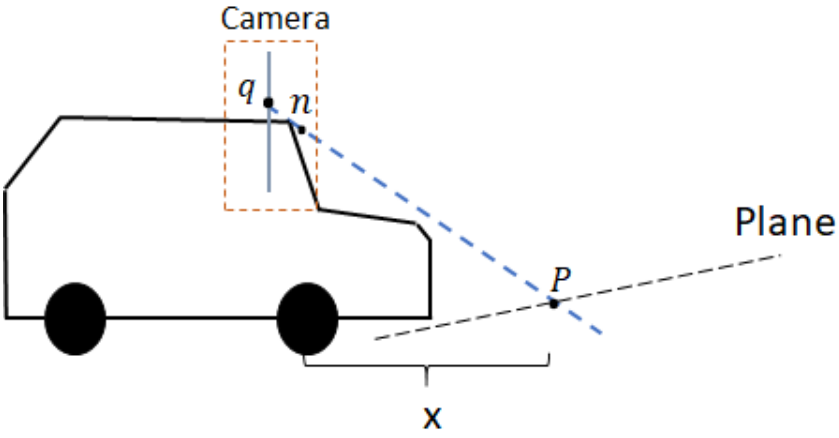
3.3.2 Finding Good-Features-To-Track on the Road Plane

First, as illustrated in Figure 3.5a, a set of 3D points P on the estimated road plane is found by projecting each image point $\mathbf{q}$ through the camera and finding the intersection point with the estimated road plane. This is explained in more detail by Nordberg [16]. Here, the representation of the plane needs to be extended with a point $\mathbf{p}_0$ in the plane. The assumption is made that a point at the bottom of the front bumper is a point in the estimated road plane. This point is a known feature of the camera setup and calibration, defined in the set of different coordinate systems related to the car.

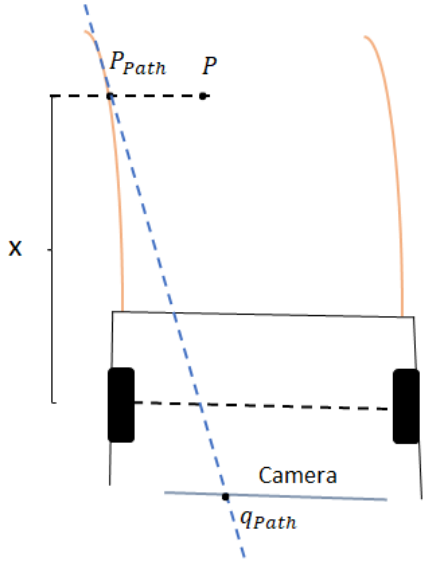
The next step illustrated in Figure 3.5b uses the distance x along the vehicle's forward direction to each 3D point P, together with the predicted wheel path is to find a 3D point P_{path} belonging to the predicted wheel path for one of the front wheels at the distance d. The 3D point P_{path} is then projected into the camera to find an image point $\mathbf{q}_{path} = (x_{path}, y_{path})$. Finally, the image point $\mathbf{q} = (x, y)$, that was used to find the 3D point P, is said to be close to the predicted wheel path if x is close enough to x_{path}. What we mean by this is illustrated in Figure 3.5c. The set of new good feature points to use is chosen as the set of all image points that are close to the predicted wheel path in the image.

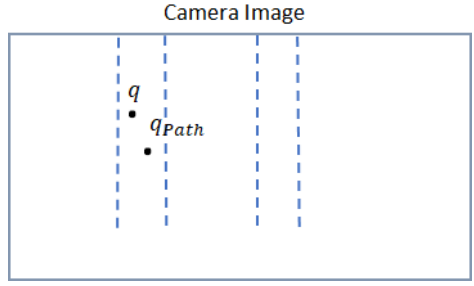
3.3.3 Patch Based Refinement

The extracted pair of corresponding image points could be refined to produce an even better match. Here, a standard KLT Tracker is used to optimize the position of the image point in the previous frame. The theory that is used is presented in Section 3.1.5. The refinement method is described in detail in Algorithm 1. Various tests showed an increase in accuracy by converting the two input images to gradient magnitude images, and performing the refinement on these gradient images. A restriction is added that only small refinement steps are expected, primarily since tests showed that a larger step had a negative impact on the method's performance.



(a)





(b) (c)

Figure 3.5: *Illustrations for how the new set of features are extracted. (a) The intersection between the projection line for each image point with the estimated road plane. (b) The extraction of a 3D point belonging to the predicted path of the vehicle. (c) The condition that determines if an image point* **q** *is close to the predicted wheel path.*

Algorithm 1: Feature Point Refinement Algorithm

Input: Image points $\mathbf{y}_{prev}$, $\mathbf{y}_{curr}$, Images $\mathbf{I}_{prev}$, $\mathbf{I}_{curr}$, Patch Size S, Max Step τ
Output: Matching Image point $\mathbf{y}_r$ in Image $\mathbf{I}_{prev}$.

```
Initiate h_0 = 0
Extract image patches F, G around image points y_prev, y_curr from images I_prev, I_curr with size SxS.
Compute Initial Residual ε_i according to Equation (3.10) for patches F, G.
for i = 0 to 5 do
    Estimate disparity vector h_i by Equation (3.12)
    Update Total disparity h = h_{i-1} + h_i.
    Compute new Image Point y_r = y_prev + h
    if New Image Point y_r ∈ I_prev then
        Compute new image patch F with size SxS around image point y_r from image I_prev
        Compute Residual ε according to Equation (3.10) for patches F, G.
    else
        Abort Refinement
if ε < ε_i and Process not Aborted and |h| < τ then
    Return y_r
else
    Return y_prev
```

3.4 Experiments

In this section the experiments performed to evaluate the performance of the proposed method are presented.

3.4.1 Experiment Setup

The dataset used in the experiments is described in Section 2.4.1. The experiment is conducted over this entire dataset and performance is evaluated using the metric defined in Section 2.4.2. A maximum step size $\tau = 1$ and patch size $S = 11$ is used for the patch based refinement.

3.4.2 Quantitative Results

The quantitative evaluation of the method's performance was performed on the evaluation dataset described in Section 2.4.1. The quantitative results with respect to the metric defined in Section 2.4.2 is presented in Table 3.1. Our proposed method significantly outperforms the baseline, mainly due to the reduced amount of noise in the output.

Table 3.1: *Quantitative results of Road Surface Preview performance on the evaluation set described in Section 2.4.1 using our Structure from Motion approach. The MSE is measured in meters.*

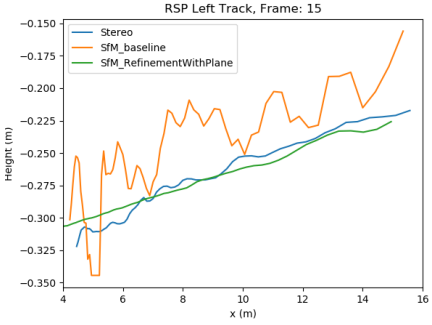
Method	MSE
The Baseline	7.807
Our Method	0.023

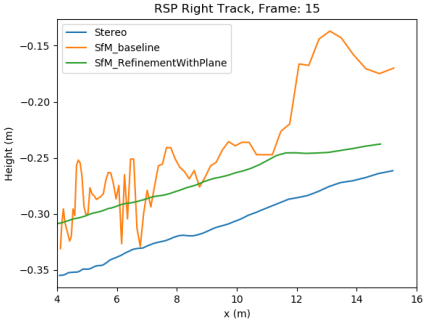
3.4.3 Qualitative Results

The qualitative results are presented in Figure 3.6. Results for the method not performing as well are presented in Figure 3.7 and in Figure 3.8. The figures compare the output between the Structure from Motion baseline defined in Section 3.2, our Structure from motion-based method and the existing Stereo RSP algorithm. More qualitative results for a more rural environment, as well as in the rain and at night, are presented in Figure A.1, A.2 and A.3 in Appendix A.

Figure 3.6 shows that our method gives a good estimation of the road profile. The noise is reduced compared to the structure-from-motion baseline and the resulting road profile is more similar to the stereo output. However, Figure 3.7 and Figure 3.8 show weaker performance for similar conditions. The results are therefore inconclusive, but the method shows good potential for these conditions. Note here that the road is not curved and the weather is clear. The more difficult case of a more curved road is discussed in Section 3.5.1. Figure A.1 shows that our method has similar performance for a more rural environment with similar road characteristics, while Figure A.2 and Figure A.3 show that our method gives a weaker estimate of the road profile in more difficult weather and light conditions.

(a)

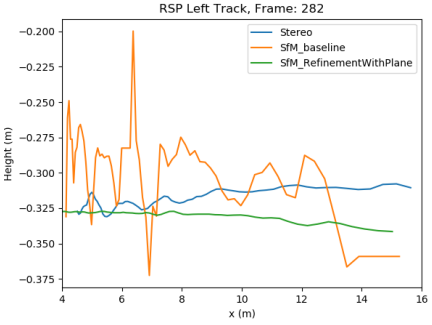


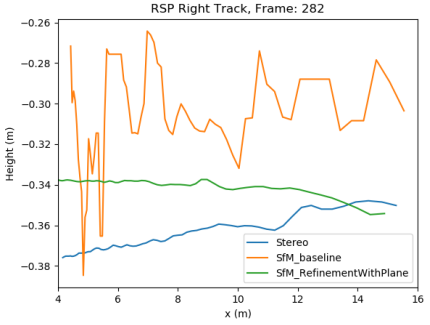


(b) *(c)*

Figure 3.6: *Examples from the evaluation dataset for the Structure from Motion Approach compared to its baseline. (a) The Camera Image. (b) Left Wheel. (c) Right Wheel.*

(a)

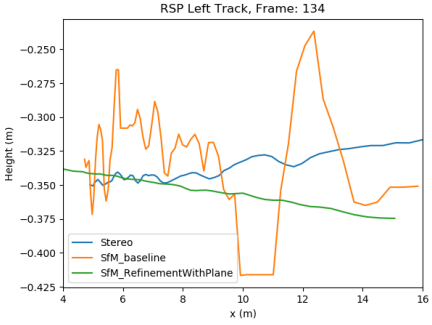


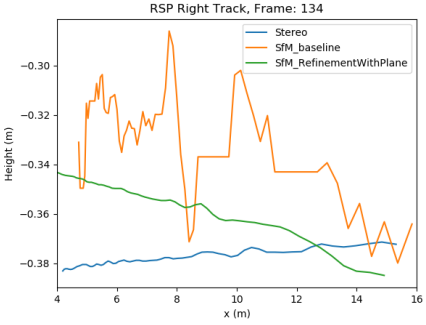


(b) *(c)*

Figure 3.7: *Poor examples from the evaluation dataset for the Structure from Motion Approach compared to its baseline. (a) The Camera Image. (b) Left Wheel. (c) Right Wheel.*

(a)





(b) *(c)*

Figure 3.8: *Failure cases from the evaluation dataset for the Structure from Motion Approach compared to its baseline. (a) The Camera Image. (b) Left Wheel. (c) Right Wheel.*

3.5 Discussion

The results for the Structure from motion-based method, as well as problems encountered during the development of the method are discussed within this section.

3.5.1 Method

The Structure from Motion framework is highly dependent on feature detection and tracking. The road surface in general may lack sufficient texture for good tracking of the features. Therefore, finding distinct features on the textureless road surface and tracking them is a challenging problem. Unreliable features would lead to unstable tracking of features as shown in Figure 3.6 for the baseline method. In our proposed method, we mainly focus on finding distinct feature points to track and tracking is performed in the same manner as described in Section 3.1.2. An assumption is made that only a small movement for each feature is expected in order to improve the stability and performance. In case of using larger steps in the patch based refinement, the overall performance deteriorated as seen in Figure 3.9. This is probably due to the unreliability of the features being tracked. The decrease in performance is small and future work could remove the need for this assumption. In the patch based refinement, the estimated road plane is used to try to find the structure of the road surface. The small step size does improve the performance, but a larger step size could improve the method's ability to detect larger bumps and irregularities on the road surface and produce an output less similar to a plane. Further work is required to investigate the full effect of this parameter in our proposed method.

The proposed Structure from motion-based method is highly dependent on the Road Plane Estimation. The initial guess for the road profile is the estimated road plane with the height determined by a point on the front bumper defined by the camera calibration. The strong dependency is clearly seen in Figure 3.8, where the road profile follows a plane, i.e. if the road surface is unfeasible to approximate with a plane in a certain frame, then the performance for that frame is not as good. Furthermore, the error from frame to frame can be quite different largely due to how the height of the road plane depends on the camera calibration. A difference of a few centimetres for the translation in the camera calibration increases the error in the results.

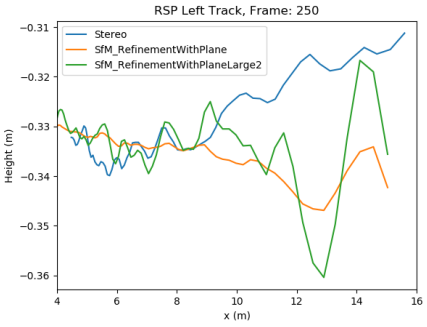
3.5.2 Results

Our method was shown to perform well as seen in Figure 3.6 and does describe the road surface accurately showing good potential. The accuracy of the road surface preview output is greatly improved with our method compared to the baseline as shown in Figure 3.6 and in the quantitative results, indicating that we have made progress towards a solution for a monocular camera using a structure from motion-based approach with required performance, e.g. that can be used in an active suspension control unit.

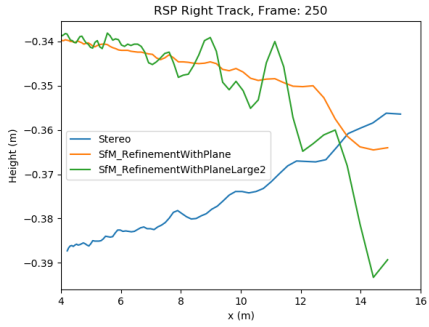
However, when the road is curved or more uneven (i.e. larger bumps) our method fail to perform as well due to the likeliness that the output still resembles a plane, as seen in Figure 3.7. As seen in the figure, a few centimeters difference in the height of the plane gives a quite large error or displacement in the resulting road profile considering centimeter accuracy is desired. The road plane estimation depends on many factors including the motion estimation and the feature detection and tracking. Both these components use features on the road surface and these features may be unreliable and hard to track accurately resulting in an inaccurate plane estimation and a decrease in the method's performance.

We also note that the method fail to perform well in night and rain conditions, as seen in Figure A.2 and Figure A.3. The visibility in the camera is greatly reduced in these conditions making the tracking of features unreliable, resulting in a weaker performance. Further investigations are required to improve the performance for these more difficult scenarios.

(a)



(b)



(c)

Figure 3.9: *Example from the evaluation dataset for the Structure from Motion Approach. Comparing the method's output for a maximum refinement step $\tau = 1$ (orange) and $\tau = 2$ (green). (a) The Camera Image. (b) Our Method Left Wheel. (c) Our Method Right Wheel.*

4

Convolutional Neural Networks (CNNs) Approach

The second approach that is proposed in this thesis is to predict the pixel-wise depth or disparity from monocular images instead of stereo pairs using a CNN. This predicted disparity can then be used in a similar way as traditional disparity maps from a stereo vision system to extract a 3D representation of the scene. In the existing RSP algorithm, disparity images in stereo is a key component, and therefore the quality of the predicted disparity image is crucial to the performance of this approach. An overview of the complete system is presented in Figure 4.1.

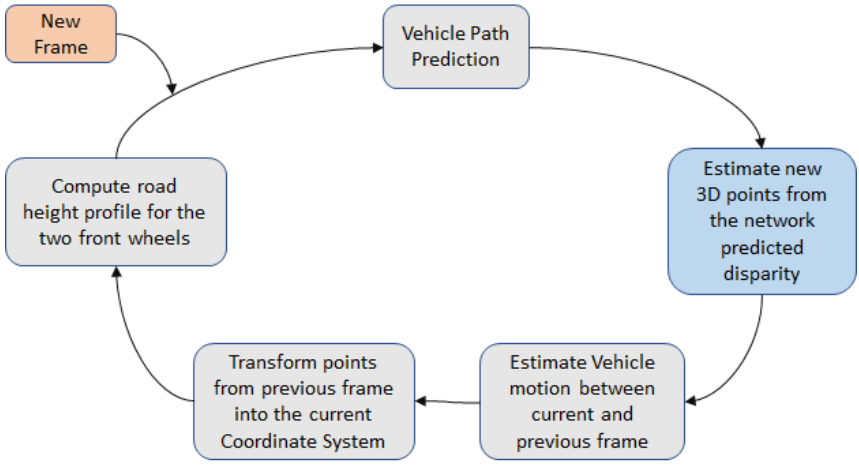


Figure 4.1: *Overview of the complete RSP system using the CNN-based approach.*

4.1 Theory

In this section, theory related to Convolutional Neural Networks and the proposed method is presented.

4.1.1 Artificial Neural Networks

The basic idea behind a CNN comes from Artificial Neural Networks (ANNs). ANNs are inspired by biological neural networks in the brain. An ANN consists of a set of layers, where each layer contains a set of neurons. A neuron receives a number of input signals and generates an output signal, similar to how a nerve cell in the brain operates. The significance for each input is described by an assigned weight. In the neuron, all the inputs are summed together followed by an activation function. The value of the activation function is the output or activation for the neuron. The neuron is illustrated in Figure 4.2. For more details regarding Artifical Neural Networks, see [1].

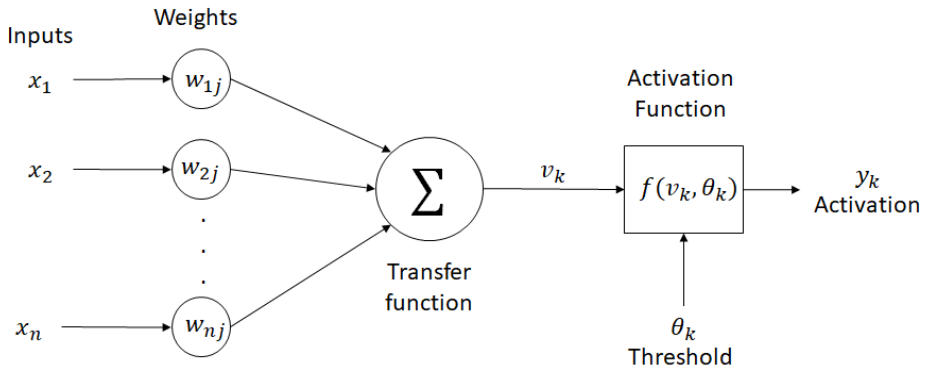


Figure 4.2: *A description of the neuron in an Artificial Neural Network.*

4.1.2 Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNN) is a method from the field of machine learning, that belongs to deep learning methods. The name deep learning comes from the often high number of hidden or internal layers in the network. A Convolutional neural network is a neural network that incorporates convolutional operations into its layers. This kind of network has proven very useful for high dimensional data related classification and vision problems. Krizhevsky *et. al* [21] demonstrated the power of CNNs on ImageNet image classification challenge by achieving superior results. CNNs have shown the same promising results for problems such as semantic segmentation [2, 26] and depth estimation [6, 7, 11, 12]. CNNs learn low level feature representations for different objects in the input image. These features are learned automatically during training and they are not hand-crafted, which is the major benefit and difference compared to

classical computer vision methods.

In the following sections the key concepts and components of CNNS are described as well as the components utilized in our proposed approach.

4.1.3 Convolutional Layers

In this section different kinds of convolutional layers are presented. All of the layers implement the convolution operation according to the description for the regular convolutional layer, but with different filter representations. All the layers presented here are utilized in the network architecture used in the thesis.

Regular convolutional layer

The convolutional layer is the main building block of a CNN architecture, and the primary learnable layer. These kind of layers are defined by their filter size $F \times F$ and stride S and for an input of size $X \times Y \times D$, where D is the number of color channels, produce a feature map of size $X_{conv} \times Y_{conv}$. The spatial size F of the filter determines the number of weights in the filter and also controls the receptive field of the filter. The size of the receptive field is the number of elements in the input volume that contributes to the convolutional output in each position. The input volume contains the D layers of spatial information each with dimensions $X \times Y$. The stride is the step length that the filter uses when it traverses the image to perform the convolution. The spatial size of the feature map is determined by the filter stride S, spatial size F and potential zero padding P.

The output feature map dimensions is defined as

$$\begin{aligned} X_{out} &= (X_{in} - F + 2P)/S + 1 \\ Y_{out} &= (Y_{in} - F + 2P)/S + 1 \end{aligned}. \tag{4.1}$$

With appropriate zero padding P and stride $S = 1$ the output will have the same spatial dimensions as the input, as seen in Figure 4.3.

The produced feature map is then sent through an activation function (described in Section 4.1.4) that applies non-linearity to the feature values. The number of filters D_{conv} in the convolution layer determines the size of the final output volume of activation maps, as the output has the dimensions $X_{out} \times Y_{out} \times D_{conv}$. The number of learnable parameters in a convolutional layer with input dimensions $X_{in} \times Y_{in} \times D$ and using D_{conv} filters, each with spatial size $F \times F$ and bias term b, is described by

$$L = (F \times F \times D) \times D_{conv} + D_{conv}. \tag{4.2}$$

Dilated convolutional layer

A large receptive field for the network is a desired property. However, in a conventional CNN architecture, the methods used for this purpose involve enlarging the filters. The dilated convolutional layer was introduced by [4] to increase the receptive field without increasing the number of weights in the convolutional

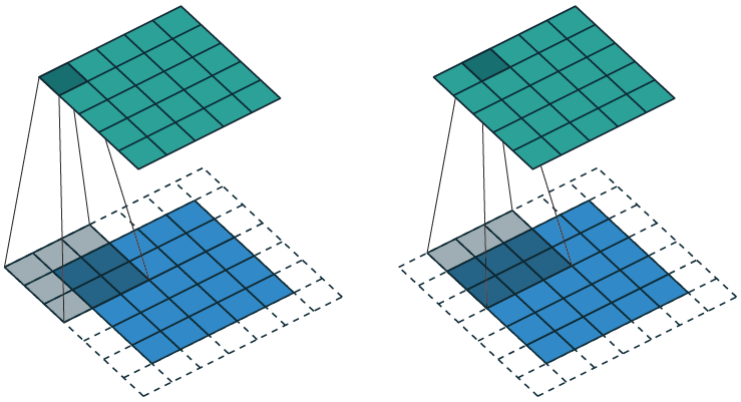


Figure 4.3: *The regular convolution of a zero padded input, the lower grid, with filter size 3 x 3 and stride 1 resulting in a feature map, the upper grid, with equal spatial resolution. Figure provided by [5].*

layer. Usually downsampling is performed by a pooling layer before the convolution layer, effectively reducing the spatial dimensions and removing redundant information. However, if the network is used for a dense prediction task such as in pixel-wise depth prediction, the feature maps have to be mapped to the spatial dimensions of the input image. The number of feature responses is smaller than the full image resolution resulting in a sparse feature response map. This could in turn be interpolated to a dense representation, but interpolation alters pixel values that could distort the feature map. The dilated convolutional layer avoids these issues while still increasing the receptive field of the network. A full or more complete motivation for the dilated convolutional layer is presented by Schennings [29] and Chen *et. al* [4].

The dilated convolutional layer increases the receptive field by upsampling the filter, e.g. inserting zeros between each weight in the filter kernel. A *rate* parameter r directly controls how many zeros, i.e. if $r = 2$ the filter is upsampled by a factor 2 and one zero is inserted between each filter weight (see Figure 4.4). The new larger filter is then used to perform the convolution operation, giving a larger receptive field. Even though the filter is larger, the number of numerical operations and filter parameters are unchanged due to that the additional zeros can be disregarded when performing the convolution operation. This gives the dilated convolution layer its main advantage, an increase in the receptive field without adding any more parameters or complexity to the network. Further, the dilated convolution layer in comparison to the more traditional method of spatial downsampling using a pooling layer, preserve information in the image while still increasing the receptive field.

The output size $X_{dilated} \times Y_{dilated}$ for the dilated convolutional layer with filter spatial size F, stride S, padding P and dilation rate r applied to input with size

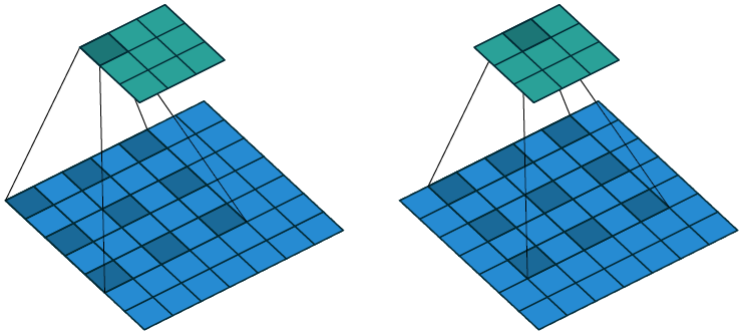


Figure 4.4: *The dilated convolution filter with size 3 x 3 and r = 2 giving a receptive field of 5 x 5. The parameter r increases the receptive field of the filter while keeping the number of parameters in the filter constant. Figure provided by [5].*

$X \times Y$ is described by [5] to be expressed by

$$\begin{aligned} X_{dilated} &= \frac{X + 2P - F - (F-1)(r-1)}{S} + 1 \\ Y_{dilated} &= \frac{Y + 2P - F - (F-1)(r-1)}{S} + 1 \end{aligned} . \tag{4.3}$$

Asymmetric convolutional layer

The regular convolutional layer uses a spatial size of F x F. In a network designed for vision problems it is expected that the output of nearby activations are strongly correlated. Szegedy *et. al* [31] proposed, based on this concept, The Asymmetric convolutional layer. This new layer factorize the regular symmetric convolution with spatial size F x F into a series of two asymmetric convolutions with spatial sizes F x 1 and 1 x F. This is effective at reducing the total number of parameters in the convolution from 3 x 3 = 9 to 3 x 1 + 1 x 3 = 6, for a filter size of $F = 3$. The gain in reduced complexity increases rapidly with a larger filter size F. This reduction in complexity is achieved while maintaining the receptive field of the original symmetric filter. Since the computational complexity of a CNN due to the large amount of parameters is a limitation this is a highly desired property. Furthermore, this makes it possible to use larger filters in the asymmetric convolution layer compared to the regular, which in turn increases the receptive field of the network, without increasing the complexity.

Deconvolution layer

The convolutional neural network traditionally downsamples the image and extract feature maps with smaller spatial dimensions than the input. This is suitable for classification tasks as the output is a global decision, but for regression tasks, the spatial size of the output should correspond to the size of the input.

This could be done by inserting zeros between the existing values and using standard interpolation methods. Another approach that was introduced by Long *et. al* [24] is performing deconvolution. A deconvolution layer tries to learn a suitable interpolation function based on the training data. The functionality of the layer is simply a reversed convolution layer, where the feature maps in the input is dilated with zeros and then convolved. This produces the effect of normal interpolation with a spatially larger output. When combined with activation functions this kind of layer even has the potential to learn non-linear upsampling. By using the deconvolutional layer, the network is able to produce higher detailed and spatially larger output predictions instead of using interpolation to make the outputs larger outside of the network. This also means that the upsampling and interpolation scheme is affected by learning. The deconvolutional layer is illustrated in Figure 4.5.

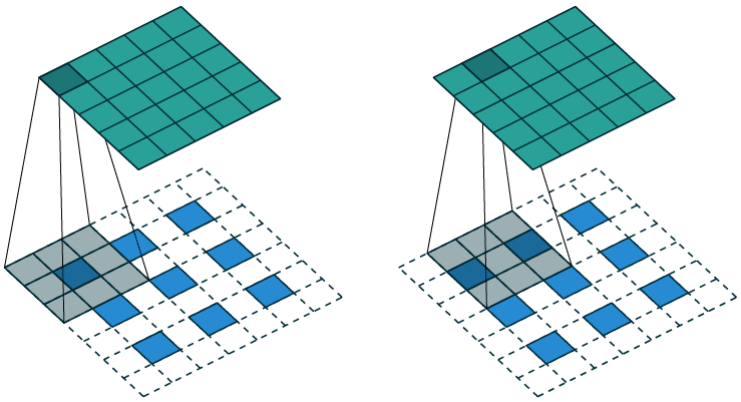


Figure 4.5: *The deconvolutional layer applied to an input with spatial dimensions 3 x 3, lower grid, produce an output with size 5 x 5, upper grid, with filter spatial size 3 x 3, stride 1 and rate r = 1. Figure provided by [5].*

4.1.4 Activation Functions

Activation functions add non-linearity between network layers which helps in fitting the network model to the data. The function transforms the output from each neuron, or in other words position in a feature map. Sometimes the network is described as a set of neurons, where the connections between neurons contain the weights in the network. A common and popular choice of activation function is the Parametric Rectified Linear Unit (PReLU) [22]. It was introduced by He *et. al* [17]. A major reason for this choice is that the function has a simple gradient and gives a high convergence rate for the network during training, which both are desired properties for an activation function in a CNN.

Parametric Rectified Linear Unit

The Parametric Rectified Linear Unit (PReLU) is a modification of the ReLU [14], first introduced by He *et. al* [17] to address the absence of gradients for negative feature values in the ReLU. Formally, the PReLU function is defined as

$$f(y_i) = \begin{cases} y_i, & \text{if } y_i > 0 \\ a_i y_i, & \text{if } y_i \leq 0 \end{cases}. \tag{4.4}$$

Here, y_i is the input to the activation function f on channel i, and a_i is a coefficient controlling the slope of the negative part. The subscript i indicate that the slope can vary across different channels. In the PReLU function a_i is a learnable parameter, which is a part of the network training. The motivation behind the PReLU function is to avoid zero gradients for negative input, which is the case for the standard ReLU function. He *et al.* [17] show a significant increase in performance when replacing all ReLUs with PReLU activation functions throughout their network. The modification of the ReLU only introduces a very small number of extra parameters. The number of extra parameters is equal to the amount of channels, which is very small compared to the total number of weights in all the convolutional layers. This is good since no significant extra complexity is added and no further risk of overfitting during training. Too many parameters in a network greatly increase the risk of overfitting, which is the term for adapting too much to the training data and failing to generalize to unseen samples. The PReLU activation function is illustrated in Figure 4.6.

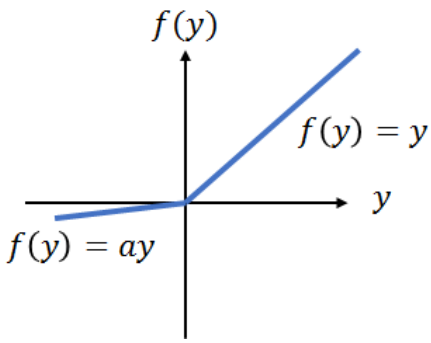


Figure 4.6: *The Parametric Rectified Linear Unit (PReLU) activation function.*

4.1.5 Pooling Layers

The pooling layer is used to modify the spatial dimensions of its input, downsampling the image and the extracted features. A pooling layer uses a filter kernel that is moved over the entire input feature map, applying a max or average operation to the feature values. The filter kernel is usually a square with a small size compared to the image, typically 2x2 or 3x3. The downsampling factor is

controlled by the stride S and the spatial size F of the kernel. For an input of size $X_{in} \times Y_{in} \times D_{in}$ the output dimensions can be described as

$$\begin{aligned} X_{out} &= (X_{in} - F)/S + 1 \\ Y_{out} &= (Y_{in} - F)/S + 1. \\ D_{out} &= D_{in} \end{aligned} \tag{4.5}$$

We apply the max pooling operation on the sliding window, extracting only the highest pixel value within each sliding window. An illustration of this kind of pooling layer, called Max Pooling, is presented in Figure 4.7.

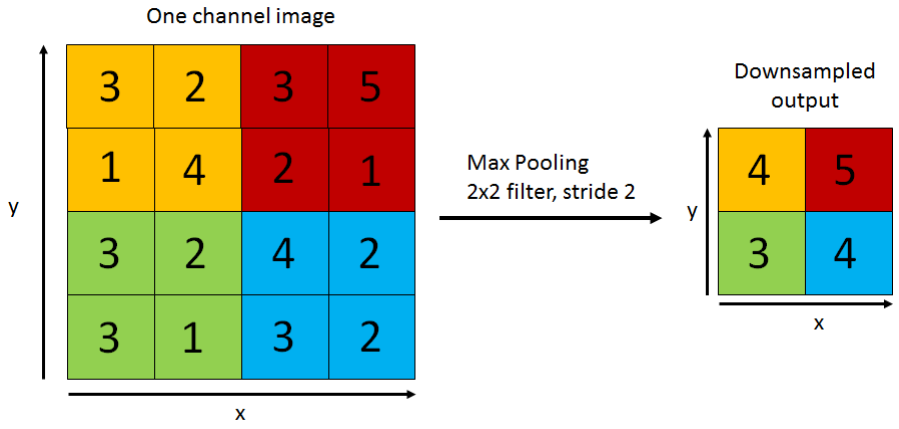


Figure 4.7: *The max pooling operation with kernel size 2x2 and stride 2 on a small 4x4 one channel input.*

4.1.6 Loss Function

The loss function is used to evaluate the performance or accuracy of the network prediction during training. The prediction $\hat{\mathbf{y}}$ is evaluated against the ground truth $\mathbf{y}$ in the loss function that computes the prediction error. Different tasks may require different loss functions. For the task of depth prediction, it is common to use either $\mathcal{L}_1$ or $\mathcal{L}_2$ norm as the loss function. We use the $\mathcal{L}_1$ loss defined as:

$$\mathcal{L}(p(\mathbf{x}), Z(\mathbf{x})) = \frac{1}{n} \sum_{i=1}^{n} |p(\mathbf{x}_i) - Z(\mathbf{x}_i)| = \|p(\mathbf{x}) - Z(\mathbf{x})\|_1, \tag{4.6}$$

where n are the number of pixels in an image, $p(\mathbf{x})$ is the prediction from the network and $Z(\mathbf{x})$ is the ground truth. We chose to use the $\mathcal{L}_1$ loss due to its frequent use in research and based on the results from the analysis of different loss functions presented by Schennings [29].

4.1.7 Training of Convolutional Neural Networks

As mentioned earlier, CNNs have different types of layers. Some of them have learnable parameters, e.g. the diffrent kind of convolutional layers and the PReLU activation function. The network iteratively updates these parameters during the training process to increase the prediction accuracy. Each iteration is also referred to as a paramter update since the parameters in the network are updated for every iteration. The training is usually run for a fixed number of epochs or until a certain stop condition is satisfied, for example, the improvement of the loss function is significantly small. The non-learnable layers on the other hand process the data as intermediate steps inside the network and can not be learned or optimized.

The network predicts an output $\hat{\mathbf{y}}$ for an unknown input $\hat{\mathbf{x}}$ by training the network with known input $\mathbf{x}$ and output $\mathbf{y}$. The network for input $\mathbf{x}$ predicts output $\hat{\mathbf{y}}$ according to

$$\hat{\mathbf{y}} = \mathbf{W}\mathbf{x} + \mathbf{b}, \tag{4.7}$$

where $\mathbf{W}$ is the weights and b the bias term. The goal in training is to learn the mapping from known input $\mathbf{x}$ to the output $\mathbf{y}$, by minimizing the error between the prediction $\hat{\mathbf{y}}$ and the true output $\mathbf{y}$. The algorithm is defined in Algorithm 2.

Algorithm 2: Neural Network Training Algorithm

Input: Input $\mathbf{x}$, Loss function $\mathcal{L}$, Ground Truth $\mathbf{y}$, Epochs K, Weights $\mathbf{W}$
Output: Prediction $\hat{\mathbf{y}}$
Dataset Ω**:** Training set Ω_{train}, Validation set Ω_{val}, Test set $\Omega_{test} \subset \Omega$ with $\Omega_{train} \cap \Omega_{val} \cap \Omega_{test} = \emptyset$
1 **Initialization:** init_weights($\mathbf{W}$), $i = 0$
while $i \leq K$ **do**
2 Generate prediction $\hat{\mathbf{y}}$ for $\mathbf{x} \in \Omega_{train}$
3 Calculate the loss function $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$
4 Calculate loss gradient $\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{W}}$
5 Backpropagation of loss gradient
6 Update the weights $\mathbf{W}$ and $\mathbf{b}$ with gradient based optimization method
7 Evaluate network on $\mathbf{x} \in \Omega_{val}$
end
8 Generate predictions of test set Ω_{test} using the network with best performance on validation set Ω_{val}
9 Calculate the loss function for network predictions from Ω_{test}

Stochastic Gradient Descent

In the weight update step in Algorithm 2 a gradient based optimization method, called Stochastic Gradient Descent (SGD), is used to find the minimum of the loss function $\mathcal{L}$. It is usually an iterative method where the incremental step towards the minimum is taken in the negative gradient direction. The SGD follows

the negative gradient computed after only observing one or a few training examples. In general, the parameter update in SGD is computed with respect to a few training examples or a mini-batch instead of only one sample. This decreases the variance in the parameter update and often leads to a more stable convergence towards the solution. By using minibatches it is also possible to benefit from efficient matrix operations and parallelization in modern GPUs. A typical mini-batch size is 256, but it is somewhat limited by the GPU. A batch should be small enough to fit in the GPUs memory, and still be large enough to reduce the variance and reduce the computational complexity by maximizing use of matrix operations. The stochastic gradient descent can be defined as

$$W_{i+1} = W_i - \alpha \nabla \sum_{j=1}^{n} \mathcal{L}(W_i, x_j, y_j), \tag{4.8}$$

where W_i is the network weights at epoch i, n is the batch size, and the loss function and its gradient is computed for n pairs of input x_j and output y_j. Supplying the data in a poor order can lead to poor convergence, so to avoid this a good method is generally to chose n random samples for each mini batch and shuffle all the data between epochs. The entire training set is traversed in one epoch, in a set of mini-batches. The parameters are updated once for every mini-batch. The learning rate α is used to control how fast the network learns, i.e. how quickly the weights change.

Backpropagation

Backpropagation is one of the key methods in neural network development and the training of deep networks. In the parameter update each weight should be updated with respect to its impact on the loss function $\mathcal{L}$. This is described by the partial derivative of the loss function with respect to a specific weight. Backpropagation uses the chain rule to decompose the gradient of the loss function into local gradients for the neurons. These local gradients express how a weight impacts the loss function and is what is used in the parameter update. First the network predicts an output $\hat{y}$ for an input x by forward propagation through the network. This prediction is evaluated towards the ground truth y using the chosen loss function. The gradient of the loss function with respect to the network weights is computed at the output and then sent through the network. This is what is referred to as backpropagation, sending the gradient of the loss function back through the network from the output. The gradient is during the propagation decomposed into local gradient components that are used to update the weights. For more details on backpropagation, see [28].

4.1.8 Training Improvements

In this section, we discuss a few improvements to speed up the convergence rate, reduce the training time and reduce the risk of overfitting.

Batch Normalization

The Batch Normalization layer was first introduced by Ioffe *et. al* in 2015 [20]. The training of deep neural networks is complicated by how the distribution of each layer's input changes during training as a side effect of the parameters being updated. Training therefore requires lower learning rates and in general becomes harder. Using the Batch Normalization layer normalizes the input to each layer to have mean zero and variance one. This is done for each scalar feature in the input independently. This means that for a k-dimensional input $x = (x^{(1)} \dots x^{(k)})$, each dimension k is normalized as

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{\text{Var}[x^{(k)}]}}, \tag{4.9}$$

where the expectation and variance is computed over the whole training data set. This is impractical when combined with the Stochastic Gradient Descent described in Section 4.1.7 where the optimization in each step is done over a mini-batch of size n. Therefore the expectation and variance is instead computed over each such mini-batch. As an extension to the normalization operation, two learnable parameters for each dimension in the input is introduced. This is done since the normalization may alter or reduce the representative power of the layer. As an example, Ioffe *et. al* mention that normalizing the inputs of a sigmoid (a non-linear activation function) would constrain them to the linear aspect of the non-linearity. The new parameters γ and β are used to scale and shift the normalized value $\hat{x}^{(k)}$ for each dimension as

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}. \tag{4.10}$$

These parameters are learned along with the other parameters in the model. The full algorithm of the batch normalization (BN) layer for one activation $x^{(k)}$, where k is omitted for clarity, is presented in Algorithm 3. In the algorithm ϵ is a constant added to the variance for a mini-batch to improve the numerical stability.

Spatial Dropout

Spatial Dropout is a method to reduce overfitting introduced to the field of Convolutional Neural networks by Tompson *et al.* [32]. The general dropout layer that is more commonly used in neural networks sets network activations to zero during training. This is done typically to prevent activations from becoming too correlated, which leads to overtraining or overfitting. Tompson argued that also feature map activations and not only local activations could be strongly correlated during training. Therefore they proposed the Spatial Dropout layer, that cancels out entire feature maps during training instead of single neurons or activations. The strength of the effect is regulated with a probability parameter p. This layer reinforces the learning and forces the network to learn more robust features and not depend on a single activation or feature map to always be present.

Algorithm 3: The internal algorithm executed in the Batch Normalization layer, applied to activation x over a mini-batch. $BN_{\gamma,\beta}(x_i)$ denotes the Batch Normalization Transform BN with parameters γ and β applied to activation x_i. Provided by [20].

Input: Values of activation x over a mini-Batch: $\mathcal{B} = x_{1...m}$
Parameters to be learned: γ, β
Output: $y_i = BN_{\gamma,\beta}(x_i)$

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{(mini-batch mean)}$$

$$\sigma^2_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{(mini-batch variance)}$$

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{(normalize)}$$

$$y_i = \gamma\hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i) \qquad \text{(scale and shift)}$$

4.2 The Baseline

This section defines the baseline for development of the method using a Convolutional Neural Network. It is based on the work of Schennings [29] and modifies the existing Road Surface Preview Algorithm originally developed for a stereo camera system.

4.2.1 CNN Network Architecture

The version of The Efficient Neural Network, or ENet, proposed by Schennings [29] is the architecture that is used in our work. We chose this architecture since Schennings demonstrated state-of-the-art performance for depth estimation using this architecture, while keeping the complexity of the network low. A solution to the RSP task should run in the vehicle on an embedded system and therefore computational resources are limited. The original ENet architecture was originally proposed by Paszke *et al.* [26] and is an efficient convolutional neural network for semantic segmentation. To adapt this network to the depth estimation problem the last fully connected layer and softmax activation layer were removed. This is described in more detail by Schennings [29]. The chosen architecture is based on the encoder-decoder layout. ENet and other networks for semantic segmentation such as SegNet [2] implement this network structure. The network is divided into two parts, an encoder that downsamples the image and a decoder module that upsamples the image back to the original size. The encoder layers extract a high level feature representation of the input. The decoder extracts from these features a pixel-wise output. In this application, the output

is an estimation of the disparity in each pixel. The final layer in the network architecture is a deconvolution layer forming the pixel-wise disparity map with the same spatial size as the input. The architecture is presented in Table 4.1. The different modules in the architecture is presented in its respective subsection.

Initial block

The initial block contains one branch with a max pooling layer with size $F = 2$, stride $S = 2$ and another branch with a regular convolution layer with filter size $F = 3$ and stride $S = 2$. The output from these two branches are then concatenated to form the output from the initial block. The initial block is illustrated in Figure 4.8.

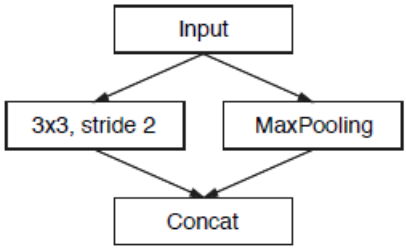


Figure 4.8: *The initial block in the network architecture. Figure provided by [26].*

Bottleneck module

The bottleneck module was introduced by He *et al.* [18] and its main advantage is the possibility to increase network depth without increasing network complexity. This is made possible by parallelization in the network. In the ENet architecture, the bottleneck modules consists of a main branch with pooling operations and a side branch with convolutional layers. The main convolution operation in the bottleneck is either a regular, dilated, asymmetric or deconvolutional layer. Between all convolutions, the network has a Batch Normalization and PReLU layer. If a bottleneck is downsampling its input, the main branch contains a max pooling layer. Further, the first 1x1 convolution in the bottleneck is replaced with a 2x2 convolution with stride $S = 2$ in both dimensions. An upsampling bottleneck instead contains a deconvolution layer in the side branch and the main branch performs upsampling by resizing the input using standard interpolation methods. The regularizer in each bottleneck module is a Spatial Dropout layer (see Section 4.1.8) with $p = 0.01$ before bottleneck2.0 and $p = 0.1$ afterwards. The bottleneck module is illustrated in Figure 4.9.

4.2.2 Dataset

The dataset from Veoneer consists of a large collection of stereo video recordings. The sequences were captured in different countries, at different time of day with

Table 4.1: *The network architecture used in this thesis. It is based on the work of Schennings [29] and the ENet architecture [26]. Type describes the convolutional component in the bottleneck module. The convolutions are performed with a filter size $F = 3$ and stride $S = 1$ in both dimensions except for the asymmetric layers that use a filter size $F = 5$. The dilated convolutions are also described with their respective dilation rate r.*

Name	**Type**
initial	
bottleneck1.0	downsampling
4x bottleneck1.x	regular
bottleneck2.0	downsampling
bottleneck2.1	regular
bottleneck2.2	dilated 2
bottleneck2.3	assymetric
bottleneck2.4	dilated 4
bottleneck2.5	regular
bottleneck2.6	dilated 8
bottleneck2.7	assymetric
bottleneck2.8	dilated 16
Repeat section 2, without bottleneck2.0	
bottleneck4.0	upsampling
bottleneck4.1	regular
bottleneck4.2	regular
bottleneck5.0	upsampling
bottleneck5.1	regular
Final Deconvolutional Layer	

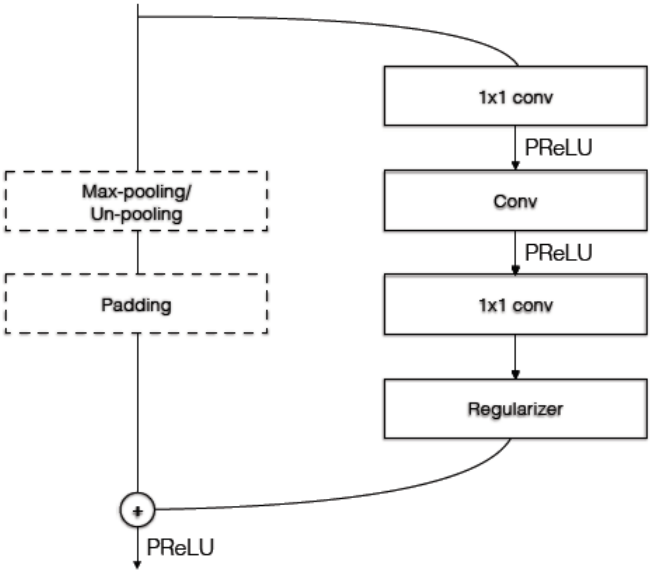


Figure 4.9: *The bottleneck module. Figure is provided by [29].*

different lighting conditions and weather scenarios. The input images used in the network training and evaluation was one of the images from the stereo camera. This involved extracting camera images from each sequence, because frames close to each other in the sequence are very similar in appearance, which would not contribute any significant new data to the training but rather increase the risk of over-fitting.

A disparity image computed from the stereo pair is used as the ground truth in the network training. The disparity image is computed by measuring the displacement of pixels between the right and left image in the stereo camera. The disparity image directly gives information about the depth, since the disparity $D(\mathbf{x})$ at pixel $\mathbf{x}$ is inversely proportional to the depth $Z(\mathbf{x})$.

The dataset split was performed by choosing different sequences from those available at Veoneer, focusing on selecting sequences from different scenarios for a more general dataset. Frames from the same recorded sequence is kept separate between the training and test set, since frames in the same sequence can be quite similar.

4.3 Our Method

In the proposed method the performance of the Road Surface Preview Algorithm depends on the quality of the disparity or depth estimation. The improvements

and modifications to the baseline therefore focuses on improving the performance of the network. Improvements are limited to not altering the network architecture due to the time constraint and to reduce the complexity of the changes. The different proposed modifications are presented in their respective section, and will be evaluated together with some exceptions. This is due to the significant training time that limited the ability to evaluate the performance for all combinations.

4.3.1 Improved Disparity Estimation for the Ground Truth

The results and discussion presented by Schennings [29] indicated that the training data could be one way to improve the estimated depth. In general, the performance of a neural network is only as good as the training data. Trying to improve the training data was therefore investigated as an initial step to improve the depth estimation compared to Schennings. Details in the depth estimation is important for RSP performance, however the disparity image used by Schennings and in the baseline during training is computed with low computational cost and therefore may lack some details. Since the training data is generated before training, a method with higher computational complexity could be used to improve the level of details in the ground truth disparity images that were used during training.

To improve the baseline, we therefore calculate the disparity image for each chosen frame pair used in training with a refined disparity algorithm, originally developed specifically for the RSP application at Veoneer. For performance reasons this algorithm was initially only used in a small region of the image in the RSP application. The algorithm is specialised on calculating the disparity on the road surface and assumptions in the calculations make it unsuitable for estimating the disparity in the entire image. The ground truth was therefore only generated for the lower and centre parts of the scene captured by the camera. Therefore all the images in the dataset have the rather odd shape of 360x808. Important to note here is that an arbitrary input size can not be used. The chosen network architecture includes a downsampling followed by upsampling with a factor eight, meaning that any image size where both dimensions does not contain the factor eight will not work.

4.3.2 Incorporating Deep Motion Features

In the baseline approach the network predicts the depth or disparity from a single camera image. To improve the accuracy of the prediction the network is supplied additional input. For this part of the method a few different options are proposed, namely

- Previous Camera Frame
- Optical Flow
- Previous Camera Frame and Optical Flow

The idea is that the previous camera frame will supply the network with additional information related to the motion of the vehicle relative its surroundings, which should also give information related to the depth in the scene. With the new input data the network can learn additional features and therefore hopefully also predict a more accurate depth. The Optical Flow is mostly another version of the same idea, but now two pre-processed input images with the movement in the x and y direction for each pixel is supplied to the network. This estimation will contain errors, but the idea is that the added information will aid the network in predicting a more accurate depth. The optical flow is computed using the Farneback algorithm [9] implemented in *OpenCV*. An illustration of how we modify the network input in practice is presented in Figure 4.10.

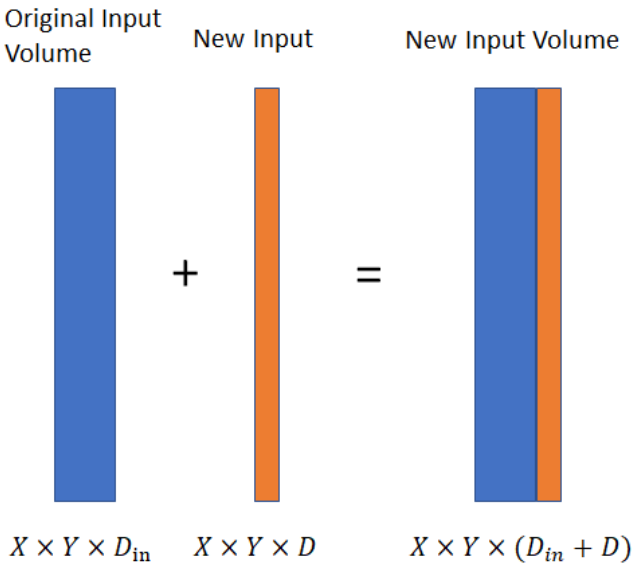


Figure 4.10: *Illustration of how we modify the network input to incorporate deep motion features, e.g. the previous camera image and the optical flow into the network. D_{in} is the number of channels in the input volume before we add the new inputs and D is the number of channels in the new input.*

4.4 Experiments

In this section the experiments performed to evaluate the performance of the proposed method is presented.

4.4.1 Implementation tools

The convolutional neural network was implemented using the Keras API [30] with Tensorflow as backend in Python. Tensorflow provide the tools necessary to construct, compile, train and evaluate the network. The Keras API simplifies

the implementation by providing an extra layer on top of Tensorflow to facilitate implementation. Any image processing needed for the creation of the dataset was performed by OpenCV and ImageMagick [23] and all arithmetic operations were performed by Numpy.

4.4.2 Experiment Setup

Here, the different experiments conducted to evaluate the performance of the proposed method is presented. All the networks were trained for 60 epochs with the $\mathcal{L}_1$ loss function, learning rate 0.001 and batch size 6. The networks were trained using a training dataset with 34592 samples, with image resolution 1024x400 for the baseline and 808x360 for our method. For more details regarding the image resolution used in our method, see Section 4.3.1. Training of the networks for our method involved computing the optical flow and cropping all the input data to the correct size. We give each version of the network a name for future reference according to

- *Baseline* - The baseline implementation.
- *ImpDisp* - Our proposed method without any deep motion features.
- *ImpDispFlow* - Our proposed method with Optical Flow as additional input features
- *ImpDispPrev* - Our proposed method with the previous camera image as additional input features
- *ImpDispPrevFlow* - Our proposed method with Optical Flow and the previous camera image as additional input features

Depth Estimation Performance

To evaluate the performance of the network with respect to the depth estimation the set of evaluation metrics proposed by [7] and also used by Schennings [29] were used. This set of evaluation metrics is commonly used in literature when evaluating the performance of a depth estimation method. The used metrics measured in meters were

- Root Mean Square Error (RMSE)
- RMSE of the logarithm of the error
- Absolute Relative Difference (ARD)
- Squared Relative Difference (SRD)
- Prediction Accuracy

defined as

$$
\begin{aligned}
\text{RMSE: } & \sqrt{\frac{1}{T}\sum_{i=1}^{T} \|p(\mathbf{x}_i) - Z(\mathbf{x}_i)\|_2^2}, \\
\text{RMSE log: } & \sqrt{\frac{1}{T}\sum_{i=1}^{T} \|\log(p(\mathbf{x}_i)) - \log(Z(\mathbf{x}_i))\|_2^2}, \\
\text{ARD: } & \frac{1}{T}\sum_{i=1}^{T} \frac{|p(\mathbf{x}_i) - Z(\mathbf{x}_i)|}{Z(\mathbf{x}_i)}, \\
\text{SRD: } & \frac{1}{T}\sum_{i=1}^{T} \frac{|p(\mathbf{x}_i) - Z(\mathbf{x}_i)|}{Z(\mathbf{x}_i}, \\
\text{Accuracy: } & \frac{1}{T}\sum_{i=1}^{T} \left[\max\left(\frac{p(\mathbf{x}_i)}{Z(\mathbf{x}_i)}, \frac{Z(\mathbf{x}_i)}{p(\mathbf{x}_i)} \right) < thr_j \right], thr = \left[1.05, 1.25, 1.25^2, 1.25^3 \right],
\end{aligned}
\tag{4.11}
$$

where T is the total number of pixels to be evaluated, $p(\mathbf{x}_i)$ is the predicted depth for pixel i and $Z(\mathbf{x}_i)$ is the ground truth depth for pixel i. The accuracy measurement divides the pixels into error intervals regulated by the thresholds. Each error interval describes the portion of pixels that fall below the error threshold. The chosen threshold values were proposed by [7], and reused by [29] and correspond to a prediction accuracy of $\{\pm 5\%, \pm 25\%, \pm 56.25\%, \pm 95.31\%\}$ compared to the ground truth depth. For the RSP application studied in this thesis and its high demand for precision, an evaluation of depth estimation performance only using an accuracy of ±25% is not detailed enough. Therefore we have chosen to add an additional accuracy threshold of ±5% compared to [29] to get a more detailed evaluation.

The dataset used to train the networks also contained a test dataset with 1573 sequences. The properties of this dataset was the same as for the evaluation dataset described in Section 2.4.1. This test dataset was used to evaluate the depth estimation performance with respect to the chosen metrics.

Road Surface Preview Performance

The dataset used in the experiments is described in Section 2.4.1. This evaluation dataset is a subset of the testset, previously used to evaluate the depth estimation performance. This is largely due to time constraints forcing the use of a smaller evaluation dataset for this part. The experiment is conducted over this evaluation dataset to give a more robust measurement of the methods performance applied to a general sequence. The performance with respect to the Road Surface Preview output is evaluated using the metric defined in Section 2.4.2.

4.4.3 Depth Estimation Results

Due to the change in training data between our method and the baseline a quantitative evaluation and comparison between the methods with respect to the depth estimation is not relevant. The quantitative results displayed here regarding the depth estimation will therefore focus on the different aspects in our method and how they affect the depth estimation performance. The quantitative results with respect to the evaluation metrics in Section 4.4.2 are presented in Table 4.2 for our four methods. Our two methods with deep motion features, named *ImpDispPrev* and *ImpDispPrevFlow*, performed the best with respect to the selected metrics with an increase in performance compared to the baseline. The qualitative results for the depth estimation are presented in Figure 4.11 for the baseline and in Figure 4.12 for our method.

The qualitative results show that the baseline and our method predicts a reasonable estimate of the depth in the scene. For our proposed method, objects such as the guard rail is not detected since the algorithm that produced the training data approximated the whole scene as belonging to the road surface, therefore the network prediction shows the same properties.

Table 4.2: *Quantitative results of CNN network depth estimation performance on the evaluation dataset, measured with the metrics described in Section 4.4.2. The best results are marked in bold. The RMSE, RMSE log, ARD and SRD are all measured in meters.*

	Lower is better				Higher is better			
Method	RMSE	RMSE log	ARD	SRD	$thr = 1.05$	$thr = 1.25$	$thr = 1.25^2$	$thr = 1.25^3$
ImpDisp	1.683	0.1287	0.0833	0.309	0.557	0.9173	**0.9800**	**0.9934**
ImpDispFlow	1.686	0.1289	0.0843	0.325	0.561	0.9180	0.9790	0.9923
ImpDispPrev	1.692	0.1278	0.0836	0.334	**0.564**	**0.9198**	**0.9800**	0.9931
ImpDispPrevFlow	**1.666**	**0.1277**	**0.0831**	**0.307**	0.558	0.9195	0.9799	0.9932

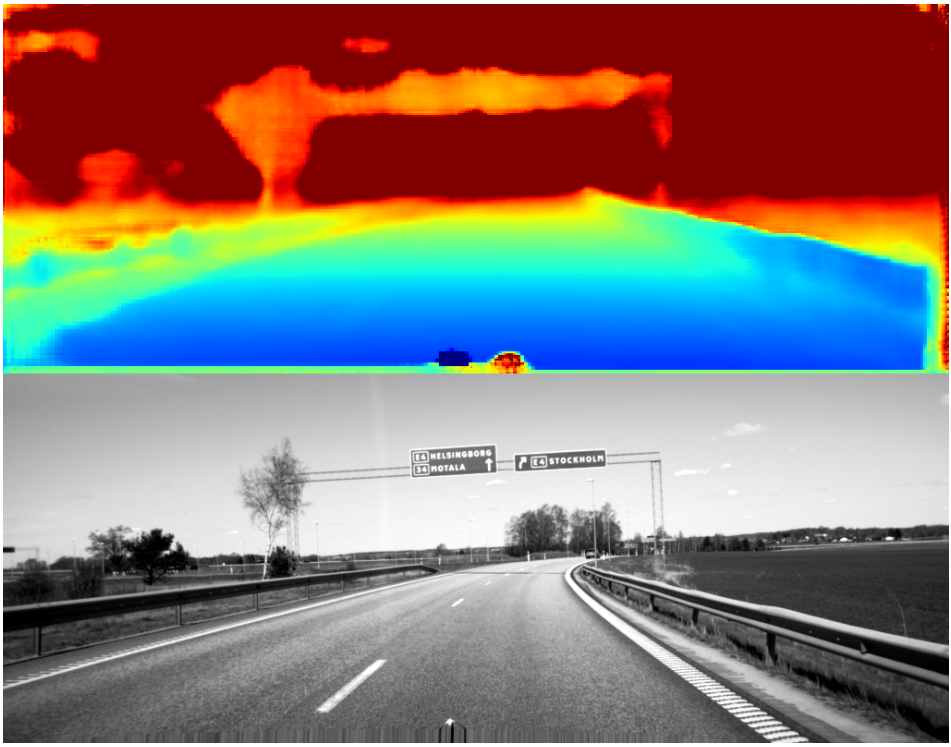


Figure 4.11: *The qualitative results for the depth estimation for the baseline. Prediction of the depth for the entire image.*

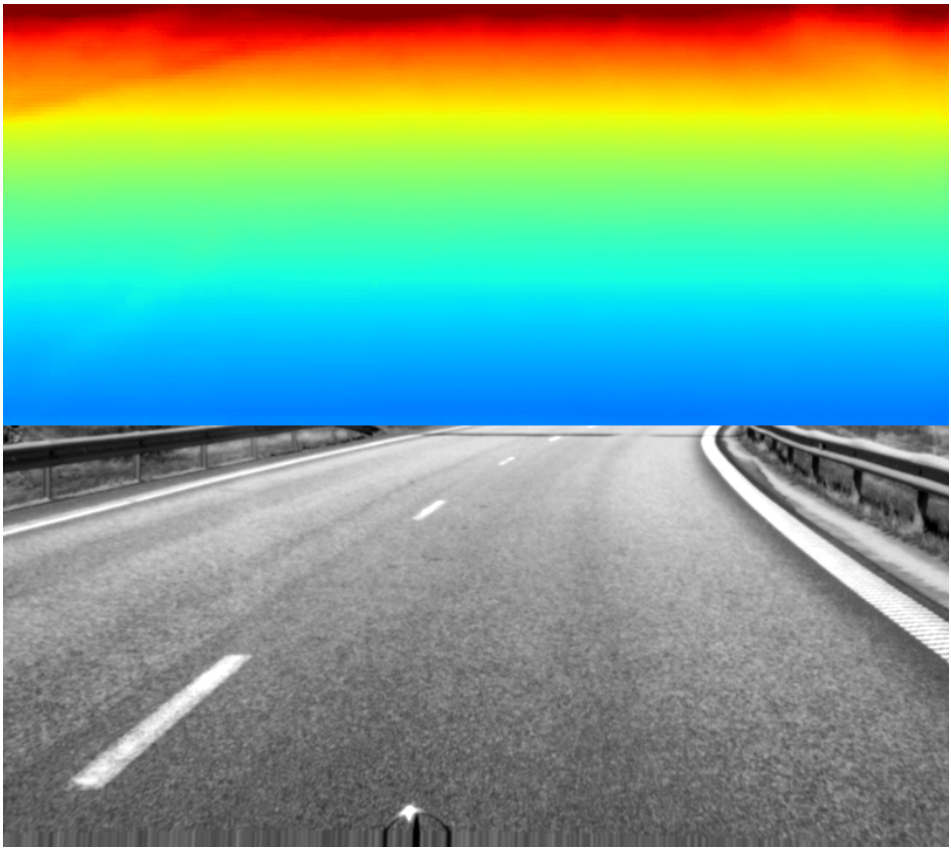


Figure 4.12: *The qualitative results for the depth estimation for our proposed method using optical flow as deep motion features. Prediction of the depth for the lower and centre part of the image. Objects such as the guard rail is not detected since everything in the scene is approximated as belonging to the road surface.*

4.4.4 Road Surface Preview Results

In this section the qualitative and quantitative results with respect to Road Surface Preview performance are presented for the CNN approach. The results for our proposed method is compared with the CNN baseline defined in Section 4.2.

Quantitative Results

The quantitative evaluation of the method's performance on the evaluation dataset. The results with respect to the metric defined in Section 2.4.2 comparing the different versions of the network and their performance on the Road Surface Preview problem is presented in Table 4.3. The quantitative results show that all the different versions of the proposed method demonstrate similar performance, with significant performance increase compared to the baseline.

Table 4.3: *Quantitative results of Road Surface Preview performance on the evaluation set described in Section 2.4.1 using our CNN-based approach. The MSE is measured in meters.*

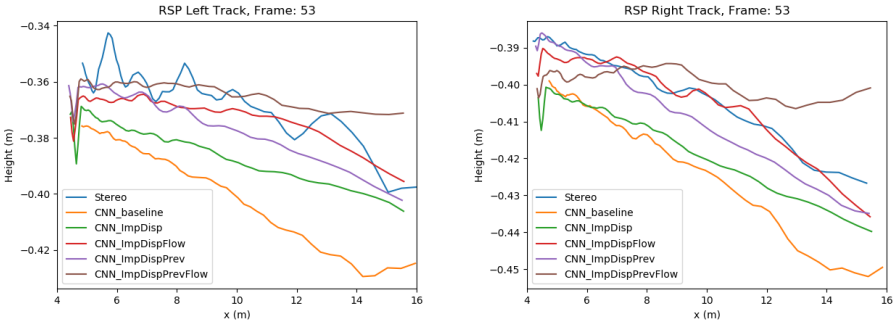
Method	MSE
The Baseline	0.059
ImpDisp	0.021
ImpDispFlow	0.021
ImpDispPrev	0.021
ImpDispPrevFlow	0.021

Qualitative Results

The qualitative results are presented in Figure 4.13. Results for the method not performing as well is presented in Figure 4.14. The figures compare the output between the CNN baseline, our CNN-based methods and the existing Stereo RSP algorithm. More qualitative results for a more rural environment, as well as in the rain and at night, are presented in Figure A.4, A.5 and A.6 in Appendix A.

Figure 4.13 show that our proposed method gives a good estimation of the road profile with potential to detect details on the road surface. The method shows potential as the estimation of the height is good, but more work is required to improve the accuracy in the details. The results are too inconclusive to determine which version of features, or network input, performs the best. Figure 4.14 show that our method performs weaker in some cases even though the weather and light conditions remain the same, meaning that the network has failed to generalize. Figure A.4 indicate that the network performs weaker in a more rural environment, probably because the training dataset did not contain enough examples of rural environment. As for Figure A.5 and Figure A.6 the performance of our method shows good potential. Especially at night, our method show signs

(a)

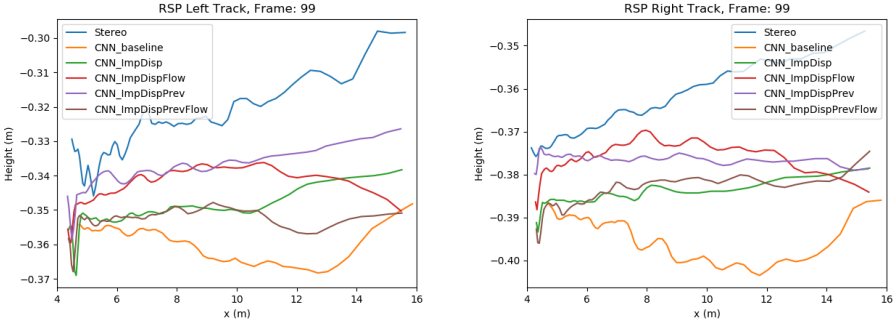


(b) *(c)*

Figure 4.13: *Examples from the evaluation dataset for the CNN Approach and its baseline. (a) The Camera Image. (b) Left Wheel. (c) Right Wheel.*

of being more robust than perhaps even the stereo RSP. Further investigation and a more thorough evaluation against sensor measured data is required to properly analyze these results.

(a)



(b) *(c)*

Figure 4.14: *Poor examples from the evaluation dataset for the CNN Approach and its baseline. (a) The Camera Image. (b) Left Wheel. (c) Right Wheel.*

4.5 Discussion

The results for the proposed CNN-based method, as well as strengths and weaknesses of the method itself is discussed within this section.

4.5.1 Method

Strengths

The performance of the proposed method when compared to the Stereo system is only limited by the CNN network performance. In the proposed method the network is trained on data extracted directly from the Stereo system, meaning that the networks learns to predict a disparity image similar to the one used in the Stereo system for RSP. Furthermore, the model used to describe the problem is not rigid, but instead it is constantly updated and can therefore adapt to the current situation. The set of features that the network learns does not depend on any fixed universal threshold that should apply at all times, and is not chosen based on any fixed method such as the Harris measure [15]. This means that the method should be good at adapting to different lighting,weather and road conditions as long as the network is strong at generalization.

Another strength is that the complete system is very similar to the already developed Stereo system, with only one new component. Since the other components are well tested and in commercial products, we can have more confidence and focus specifically on this new component in trying to increase the performance further.

Weaknesses

In general, the estimation of the depth from the CNN network will always be a prediction or approximation to the true depth as the geometric model of the scene is implicit and only learned by the network. Therefore, other methods that use explicit geometric constraints to find or estimate the depth could potentially reach a higher performance.

The performance of a CNN network is largely limited by the quality of the dataset and even the size of the dataset. To improve the size of the training dataset can be expensive. Further, the networks ability to predict the depth for a general image or scene highly depends on the distribution of the training dataset with respect to road types, weather conditions and time of day. For example, if the proposed method should be able to detect speed bumps the training dataset needs to be large enough with respect to speed bump frames that the network learns to predict the depth good enough. However, this is generally not the case since most of the recorded data of the kind used in this thesis and in similar work is recorded in city, rural or highway settings where speed bumps are quite uncommon. To collect enough data for the specific situations could therefore be even more costly and time consuming.

4.5.2 Results

Depth Estimation Results

The proposed network architecture and training scheme is able to predict an estimate of the depth on the road surface, as indicated by the qualitative results in Figure 4.12. A comparison between the baseline and our method with respect to the depth estimation performance is not feasible since the dataset and output are different. A more quantitative evaluation of the network performance of our method relative to other methods is not included here, since the dataset is not public. It would be possible to train other methods on the used dataset, or even train our network on for example the KITTI dataset. However, since depth estimation is not the main focus of this thesis, but rather the RSP performance this has not been investigated more in this thesis.

The quantitative evaluation is instead focused on the effect of deep motion features in the network training with respect to the network performance. The results in Table 4.2 indicate that the deep motion features, e.g optical flow and the previous camera image, has a relative small but positive impact on the depth estimation performance.

Road Surface Preview Results

The proposed method is able to compute an estimate of the road profile that shows good potential. The results seen in Figure 4.13 show that the estimated road profile is quite accurate in height, but the method is less successful at detecting the details of the road surface. The estimation of the disparity is the component that has been replaced in our method, indicating that the poor result is due to a poor disparity image. Our proposed method of improving the performance by including deep motion features as input to the network improved the results. This indicates and confirms that deep motion features, such as the ones used in this thesis, does help a CNN network to learn features suitable for depth estimation in monocular images. We note that even though the CNN depth estimation performance was not significantly improved by adding the deep motion features, the proposed method with deep motion features shows a noticeable increase in performance for the Road Surface Preview task. The estimated depth information seems to be more suitable for solving the Road Surface Preview task. However, since the increase in RSP performance is still relatively small (see Table 4.3) further investigations are required to find the most suitable combination of input images to the network. Further, a more extensive evaluation could give more insight to the impact of adding the deep motion features to the network.

For the scenes in Figure 4.14 and A.4, the proposed method is less successful in computing the road profile. As discussed in Section 4.5.1, this is probably related to the depth estimation performed by the network in this frame. In other words, the network has failed to generalize for this frame. Exactly what may have caused this is hard to know, since it is tightly related to the features learnt by the network. A first suggestion to improve the performance for this frame, and other weak performing cases, is to focus on improving the depth estimation

performance with specific focus on expanding and improving the dataset.

Interesting to note are the qualitative results presented in Figure A.5 and A.6 , were the scenes are captured at night and in the rain respectively. The proposed CNN-based method generates an output quite different from the Stereo RSP and a more detailed evaluation is needed to determine if our method fail to perform as well for these cases or if our proposed CNN-based method performs better than the stereo RSP. Note, better in the sense of more accurate with respect to the true height of the road. There is no guarantee that the stereo algorithm finds a correct description of the road surface and the results look a bit unstable even for the stereo RSP for these more difficult cases.

5

Conclusion and Future Work

This chapter presents the conclusion related to the formulated problem, see Section 1.2. It also contains some direction for possible future work related to the development of an algorithm for Road Surface Preview using a monocular camera.

5.1 Conclusion

Our proposed methods have shown good potential to solve the Road Surface Preview problem using a monocular vision system. However, the achieved performance is not good enough to be used in an active suspension control unit. A Structure from Motion based approach demonstrated some potential, but was sensitive to texture on the road surface in the images. A method based on Convolutional Neural Networks performs slightly better, producing an estimate of the road profile that in average height is relatively accurate. The achieved performance during this thesis work is not good enough to be used in a vehicle implementation, but show potential and should inspire future work in development of a Road Surface Preview Algorithm using a monocular camera.

5.2 Future Work

In this section we present some suggestions for future work, with respect to each proposed method. However, first we discuss future work that is more general to both methods.

5.2.1 Evaluation against sensor measurements

We note that the evaluation could greatly benefit from comparing the output from our methods with sensor measurements of the road height near the wheels. The results in rain and at night for both the methods are quite different compared to the stereo, but it is not possible with our evaluation method to determine if our method's are closer to the true height of the road surface. A more extensive evaluation is needed to further analyze the performance of our methods. This was not focused on in this work, but the focus was rather to investigate the two different methods to find a promising way going forward for solving the RSP task in a mono system with the required accuracy. For future work and as our methods are refined, a more extensive evaluation is needed to give a more reliable measurement of their performance.

5.2.2 Structure from Motion Approach

To improve the performance and continue the development of a Structure from Motion based approach to the Road Surface Preview problem we have a few suggestions.

New Binary Descriptor

The BRIEF descriptor is used in the current implementation for describing and tracking all features. After the development of the BRIEF descriptor several new rotation invariant binary descriptors has been proposed in research that show greater performance compared to BRIEF, see [19]. Using another one of these binary descriptors could improve the performance without adding a significant increase to the complexity.

Road Plane Estimation

The estimated road plane contains valuable information that should be possible to use to aid in solving the problem. The proposed method is an attempt at this, but more work is desired to find a more suitable way to use the model of the road plane. The proposed method have known weaknesses that should be investigated further. These weaknesses were discussed in Section 3.5.1.

5.2.3 Convolutional Neural Network Approach

Future improvements to the CNN-based approach is reviewed in this section.

Data Augmentation

The size of the training dataset is a bottleneck for all deep learning methods. Data augmentation could be used in the network training to expand the size of the dataset, which in turn could lead to an increase in network depth estimation performance. The accuracy of the estimated road profile depends primarily on the accuracy of the depth estimation.

Investigate Additional Input

Adding additional input to the Convolutional Neural Network proved successful. More input data such as several camera images could further improve the performance.

Improved Training Scheme

The learning rate could be updated in a more sophisticated way during training to enable faster convergence and potentially a better performance. It would help to avoid points of local minimum in the solution.

Optical Flow Estimation

Our proposed method used Farnebäck's method [9] implemented in *OpenCV*. Since this method for computing the optical flow is based on image feature points, the result may be weak on the road surface due to low texture increasing the difficulty to track the features accurately. Farneback is a general purpose algorithm for computing the optical flow and potentially a more specific method could be used for this application resulting in a higher performance.

The estimated motion for the vehicle could be used as a prior or a condition in the optical flow computation, since the calculated optical flow should correspond well to the estimated motion. Research has also been done lately on estimating the optical flow using Convolutional Neural Networks. A recent successful attempt by Fischer *et. al* [10], showed promising results. Instead of using the Farneback computed optical flow, a CNN network could learn to estimate the optical flow from the previous image in a similar fashion as Fisher *et. al* [10]. The Optical Flow estimation could be done by a separate network as a preprocessing step to the existing network, or the current network could be expanded to also predict the optical flow internally directly from the previous camera image. The idea is that an improved estimation of the optical flow could give an increase in performance and accuracy in the depth estimation, and therefore also an increase in the method's performance applied to the Road Surface Preview problem.

Investigate other approaches for CNN depth estimation

The work by Godard *et. al* [13] use an unsupervised learning method were they exploit epipolar geometry constraints to generate disparity images. This enables training of the network without access to any ground truth depth data. As discussed, to collect data for a supervised training method is expensive. In this thesis, the ground truth data is estimated by an algorithm and not measured by a sensor. To pose and solve the depth estimation problem as an unsupervised learning problem could therefore be interesting to investigate.

Appendices

A

Additional Qualitative Results

For the interested reader additional examples of the qualitative results for both of the proposed methods are presented in this appendix in their respective sections.

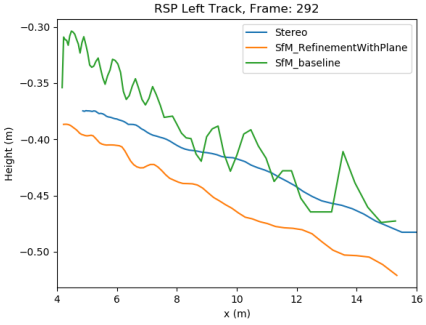
A.1 Structure from Motion Approach

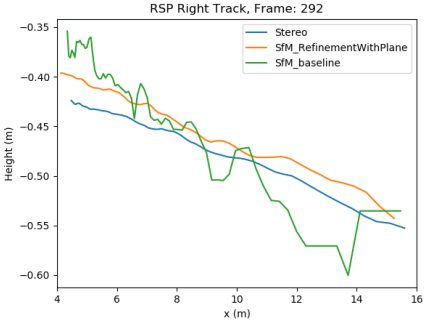
Additional qualitative results for the Structure from Motion based approach are presented in Figure A.1, A.2 and A.3.

A.2 Convolutional Neural Networks Approach

Additional qualitative results for the CNN-based approach are presented in Figure A.4, A.5 and A.6.

(a)

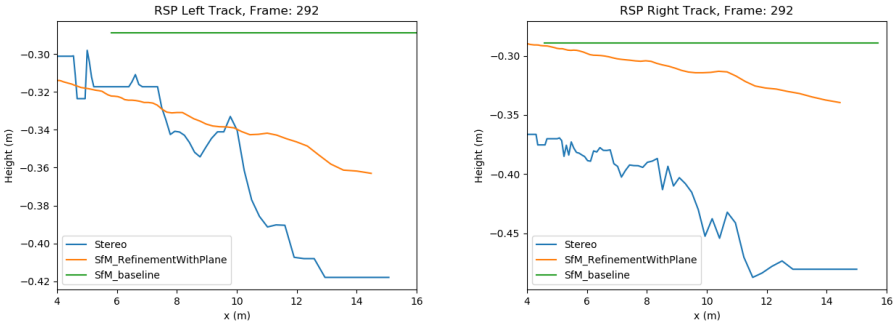




(b) *(c)*

Figure A.1: *Examples from the evaluation dataset for the Structure from Motion Approach compared to its baseline. (a) The Camera Image. (b) Left Wheel. (c) Right Wheel.*

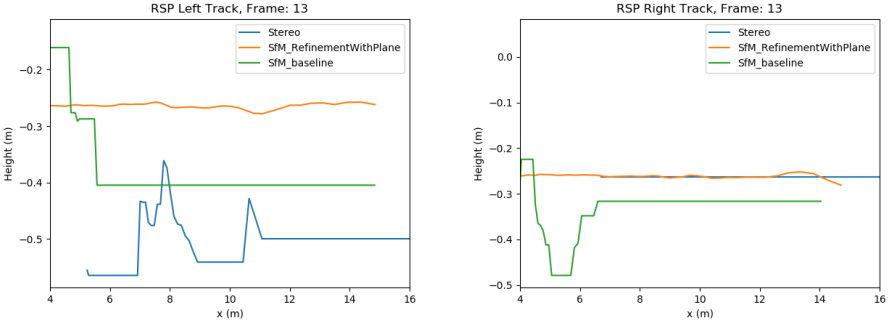
(a)



(b) *(c)*

Figure A.2: *Examples from the evaluation dataset for the Structure from Motion Approach compared to its baseline in a night scenario. (a) The Camera Image. (b) Left Wheel. (c) Right Wheel.*

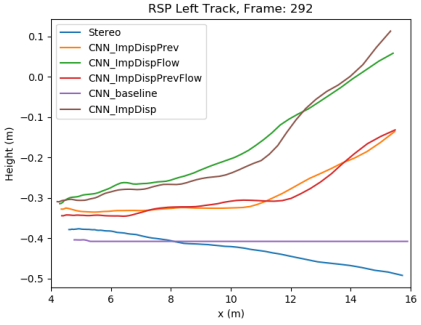
(a)



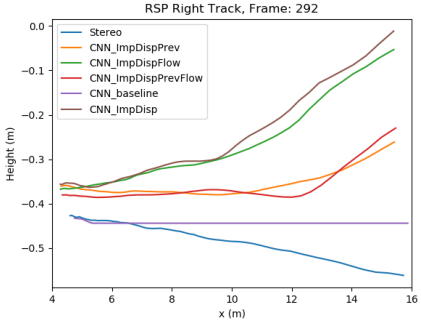
(b) *(c)*

Figure A.3: *Examples from the evaluation dataset for the Structure from Motion Approach compared to its baseline in rainy conditions. (a) The Camera Image. (b) Left Wheel. (c) Right Wheel.*

(a)



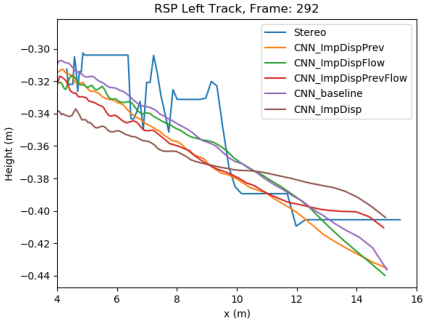
(b)



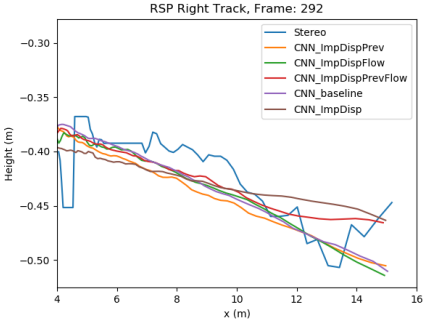
(c)

Figure A.4: *Examples from the evaluation dataset for the CNN Approach and its baseline in a more rural environment. (a) The Camera Image. (b) Left Wheel. (c) Right Wheel.*

(a)



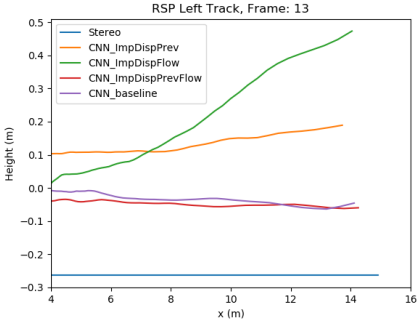
(b)



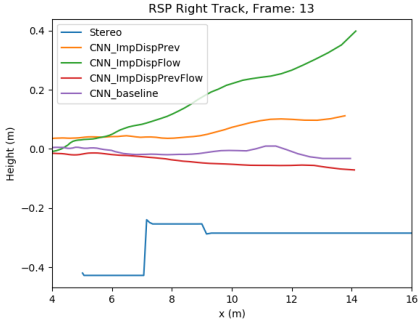
(c)

Figure A.5: *Examples from the evaluation dataset for the CNN Approach and its baseline in a night scenario. (a) The Camera Image. (b) Left Wheel. (c) Right Wheel.*

(a)



(b)



(c)

Figure A.6: *Examples from the evaluation dataset for the CNN Approach and its baseline in rainy conditions. (a) The Camera Image. (b) Left Wheel. (c) Right Wheel.*

Bibliography

[1] A.D.Dongare, R.R.Kharde, and Amit D.Kachare. Introduction to artifical neural network. *International Journal of Engineering and Innovative Technology (IJEIT)*, 2, July 2012. ISSN 2277-3754. Cited on page 30.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, 2015. Cited on pages 30 and 40.

[3] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. Brief: Computing a local binary descriptor very fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1281–1298, July 2012. ISSN 0162-8828. Cited on page 13.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, 2016. Cited on pages 31 and 32.

[5] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. Technical report, MILA, Université de Montréal, and AIRLab, Department of Electronics, Information and Bioengineering, Politecnico di Milano, Mar 2016. Cited on pages x, 32, 33, and 34.

[6] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *CoRR*, 2014. Cited on page 30.

[7] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *CoRR*, 2014. Cited on pages 30, 46, and 47.

[8] N. Fanani, A. Stürck, M. Barnada, and R. Mester. Multimodal scale estimation for monocular visual odometry. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1714–1721, June 2017. Cited on page 18.

[9] Gunnar Farnebäck. Two-frame motion estimation based on polynomial ex-

pansion. In *Proceedings of the 13th Scandinavian Conference on Image Analysis*, SCIA'03, pages 363–370, 2003. ISBN 3-540-40601-8. Cited on pages 45 and 59.

[10] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *CoRR*, 2015. Cited on page 59.

[11] Ravi Garg, Vijay Kumar B. G, and Ian D. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. *CoRR*, 2016. Cited on page 30.

[12] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. *CoRR*, 2016. Cited on page 30.

[13] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017. Cited on page 59.

[14] Richard H. R. Hahnloser, Rahul Sarpeshkar, Misha A. Mahowald, Rodney J. Douglas, and H. Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *nature*, 405:947–951, 2000. Cited on page 35.

[15] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988. Cited on pages 12 and 54.

[16] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. ISBN 052154051-8. Cited on pages 12, 14, and 18.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, 2015. Cited on pages 34 and 35.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, 2015. Cited on page 41.

[19] Jared Heinly, Enrique Dunn, editor="Fitzgibbon Andrew Frahm, Jan-Michael", Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid. Comparative evaluation of binary features. In *Computer Vision – ECCV 2012*, pages 759–773, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-33709-3. Cited on page 58.

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, 2015. Cited on pages 39 and 40.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. Cited on page 30.

[22] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521:436–444, 2015. Cited on page 34.

[23] ImageMagick Studio LLC. Image Magick. URL `http://imagemagick.org`. Cited on page 46.

[24] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, 2014. Cited on page 34.

[25] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, 1981. Cited on page 15.

[26] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *CoRR*, 2016. Cited on pages x, xii, 30, 40, 41, and 42.

[27] Srikumar Ramalingam, Suresh K. Lodha, and Peter Sturm. A generic structure-from-motion framework. *Computer Vision and Image Understanding*, 103(3):218 – 228, 2006. ISSN 1077-3142. Special issue on Omnidirectional Vision and Camera Networks. Cited on pages 16 and 18.

[28] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 1986. Cited on page 38.

[29] Jacob Schennings. Deep convolutional neural networks for real-time single frame monocular depth estimation. Master's thesis, Uppsala University, Division of Systems and Control, 2017. Cited on pages x, xii, 32, 36, 40, 42, 43, 44, 46, and 47.

[30] Open Source. Keras. URL `https://keras.io`. Cited on page 45.

[31] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, 2015. Cited on page 33.

[32] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. *CoRR*, 2014. Cited on page 39.