Christoph Herzog

Improving Stereo Matching under Difficult Conditions with CNN-based Confidence Learning

MASTER'S THESIS

to achieve the university degree of
Diplom-Ingenieur

Master's degree programme
Computer and Information Engineering

submitted to

Graz University of Technology

Supervisor

Ass.Prof. Dipl.-Ing. Dr.techn. Friedrich Fraundorfer
Institute for Computer Graphics and Vision

Dr. Christian Mostegel
Institute for Computer Graphics and Vision

Graz, Austria, June 2018

Abstract

Almost all stereo matching systems are trained and evaluated under good conditions. Stereo datasets with ground-truth have images under good lighting conditions and no disturbances like raindrops, bright lights with lens flares or reflections. There are a few datasets available that are trying to overcome this problem, but unfortunately they have either no ground-truth at all or the ground-truth is missing for the more severe effects.

We propose a method to generate datasets with pseudo ground-truth and disturbances in the images. We also propose a system to learn confidence over these disturbances and incorporate them into the standard stereo pipeline. Additionally, we record our own dataset with raindrops on windshields as disturbance to evaluate the confidence system.

We show that our system can significantly reduce the error on this dataset.

Kurzfassung

Fast alle Stereo-Matching-Systeme werden unter guten Bedingungen trainiert und bewertet. Stereodatensätze mit ground-truth bestehen aus Bilder mit guten Lichtverhältnissen und beinhalten keine Störungen wie Regentropfen, helle Lichter mit Lens Flare oder Reflexionen. Es gibt einige Datensätze die versuchen dieses Problem zu lösen, aber leider haben sie entweder überhaupt keine ground-truth oder die ground-truth fehlt für die Bereiche mit Störungen.

Wir stellen eine Methode vor, um Datensätze mit pseudo ground-truth und Störungen in den Bildern zu erzeugen. Wir stellen auch ein System vor, um diese Störungen zu lernen und Unsicherheiten in die Standard-Stereo-Pipeline zu integrieren. Zusätzlich nehmen wir unseren eigenen Datensatz mit Regentropfen auf Windschutzscheiben als Störung auf um unser System mit Hilfe dieses Datensatzes zu testen.

Diese Tests zeigen, dass unser System den Fehler auf unserem Datensatz signifikant reduzieren kann.

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used.

The text document uploaded to TUGRAZonline is identical to the present master's thesis dissertation.

Date	Signature

Acknowledgments

First and foremost I would like to thank Ass.Prof. Fraundorfer and Dr. Mostegel for supervising my thesis and for the many suggestions during the creation of this work. I also want to thank my parents and fiancée Christina for the support during the time it took me to write this thesis.

Contents

List of Figures

List of Tables

1 Introduction

Contents

1.1 Motivation

Most modern stereo matching systems are designed for scenes in good conditions and commonly available datasets with ground-truth are either indoor scenes or real-world scenes under good lighting conditions. The most widely used indoor dataset is the Middlebury dataset [18, 20], where a structured light scanner was used to obtain ground-truth depth. The standard dataset for outdoor scenes is the KITTI dataset [3, 12], which used a laser scanner mounted on top of a car for depth sensing. Therefore, image capturing was only possible in good weather. The Middlebury dataset tries to address the limitations of an indoor dataset by changing lights and exposure, but these solutions are still very limited.

Real world applications of stereo matching have to work under much more challenging conditions like dealing with lens flares, rain drops or snow. Meister et al. [11] addressed these problems and captured a stereo dataset in multiple complex situations. Unfortunately, they were not able to capture any form of ground-truth. Kondermann et al. [8] recorded a real world dataset with ground-truth and under challenging conditions by premapping the environment with a LIDAR system. This allowed them to include disturbances like lens flares and raindrops in their dataset. Unfortunately this method is limited to a small area for recording, which makes this dataset unsuitable for machine learning. Figure 1.1 shows example images from these different datasets.

(a) KITTI dataset [18, 20]

(b) Middlebury [3, 12]

(c) HCI Benchmark dataset [8]

(d) Challenging dataset [11]

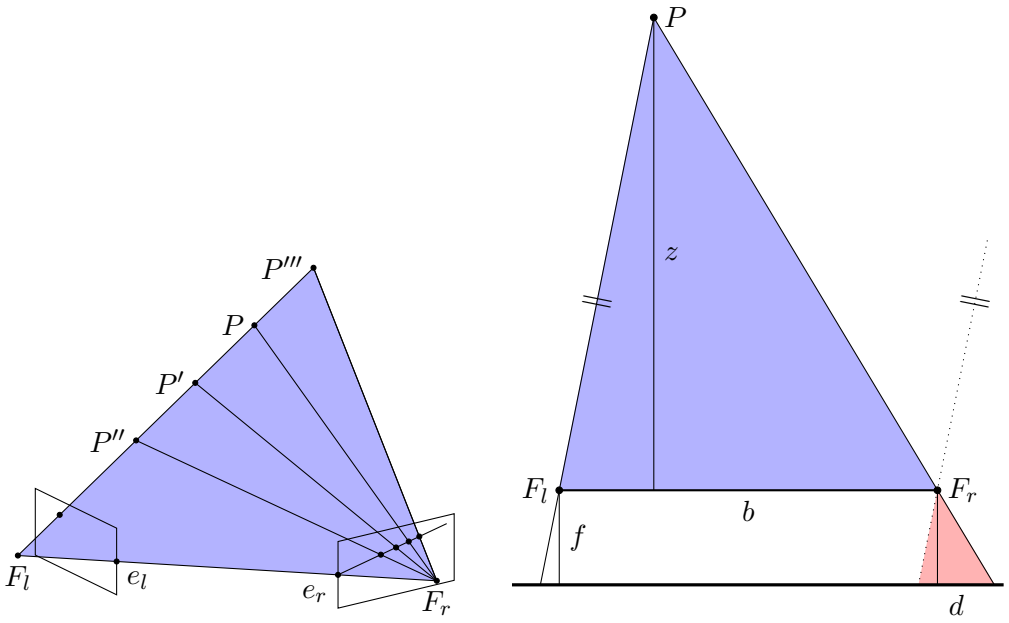
Figure 1.1: Example images from different available stereo datasets. The KITTI dataset contains real world street scenes with ground-truth under good conditions whereas the Middlebury dataset consists of indoor scenes also with good lighting. To highlight problems in real world scenes the Challenging dataset contains images with difficult conditions like snow and night but no ground-truth. The HCI Benchmark dataset also contains difficult conditions like the raindrops in this image and provide the ground-truth for the static elements of the scene.

In this thesis we trade ground-truth accuracy for the ability to record a dataset suitable for machine learning.

Park and Yoon [13] showed that modifying the cost volume based on hand crafted confidence metrics combined with machine learning improved the stereo method significantly. The two main differences to our work are that we extract a confidence measure directly from the input images, and that we incorporate our confidence earlier into the stereo pipeline.

1.2 Stereo Matching

Stereo matching is the process of finding corresponding points in two images of the same scene taken from different positions. The most important use of these corresponding points is the calculation of a depth map.



(a) A point on the image plane of the left image defines a ray into the world. This ray is projected onto the right image. Therefore, one must search only along this line in order to find corresponding points – if the relative position of the cameras is known.

(b) If the two cameras are normal to the baseline b and looking into the same direction the depth z of the Point P can be related to the known disparity d via similar triangles: $z/b = f/d$

Figure 1.2: The relation between disparity and depth in the context of epipolar geometry.

1.2.1 Depth Map

A depth map assigns each pixel a distance value consisting of how far away the point is from the camera. To calculate a depth map we must find the corresponding point in one image for each pixel in the other image.

For efficient matching both cameras have to be calibrated and the images rectified. When the cameras are calibrated and the images are undistorted we can use pinhole camera models and epipolar geometry. This reduces our search space from the whole image to only one line in the image as shown in Figure 1.2a. To simplify the search even further we can virtually rotate our cameras and use the same focal length so that the search line is a horizontal line on the same height as the point in the first image. Only a single value d is needed to locate the corresponding point: $x_r = x_l + d$. d is called the disparity.

For this rectified epipolar geometry we can calculate the depth directly from the disparity using similar triangles as shown in Figure 1.2b:

$$z = \frac{bf}{d}. \tag{1.1}$$

In the rest of this thesis we will only talk about the calculation of the disparity map (one disparity value for each pixel in one image).

1.2.2 Stereo Method

Žbonta and LeCun [24] based on [5, 19] structured the calculation of a disparity map into 2 main parts:

- The computation of a cost volume, which assigns a cost for each pixel and each disparity (up to a maximum disparity).
- Finding an optimal cut through the cost volume. This optimization is called the stereo method.

To calculate the cost volume in this thesis the fast method in [24] is used. A convolutional neural network generates feature maps for the left and right image. These 64-dimensional feature vectors are then compared with the cosine similarity. This part of the algorithm is explained in more detail in Section 1.2.3

1.2.2.1 Semiglobal Matching

To calculate a depth image from the cost volume an optimal cut through the volume has to be found.

The simplest solution for this problem is to use the lowest cost depth for each pixel. For real world images this results in very noisy depth images with a huge number of depth jumps, because the cost volume itself is very noisy. Therefore, additional constraints have to be introduced. Real world disparity maps are normally made out of two parts: objects and jumps between objects. Disparity maps of objects are either constant, if the object is flat and normal to the camera, or smoothly changing. Between objects there can be big jumps in the disparity map if one object is in front of the other.

Hirschmüller [5] proposed this energy function $E(D)$ for an optimal disparity map:

$$E(D) = \sum_{\boldsymbol{p}} \left(C(\boldsymbol{p}, D_{\boldsymbol{p}}) + \sum_{\boldsymbol{q} \in N_{\boldsymbol{p}}} P_1 \, \mathrm{T}\left[|D_{\boldsymbol{p}} - D_{\boldsymbol{q}}| = 1\right] + \sum_{\boldsymbol{q} \in N_{\boldsymbol{p}}} P_2 \, \mathrm{T}\left[|D_{\boldsymbol{p}} - D_{\boldsymbol{q}}| > 1\right] \right). \tag{1.2}$$

$C(\boldsymbol{p}, D_{\boldsymbol{p}})$ is the cost of pixel $\boldsymbol{p}$ and the disparity D of this pixel. T[] is an operator that returns 1 if the expression in the argument is true and 0 otherwise. P_1 and P_2 are penalty constants and $N_{\boldsymbol{p}}$ is the neighbourhood of pixel $\boldsymbol{p}$.

This energy function is a sum over all pixels and consists of three parts. The first part consists solely of the cost for the current pixel – with only this part the energy function would equal the simple solution. The second and third part are penalties if the disparity changes around this pixel. The penalty P_1 is lower than P_2 and corresponds to curved or tilted objects, where P_2 corresponds to discontinuities between objects. Since

discontinuities in the depth are often visible as intensity changes in the images, P_2 can be scaled indirectly proportional to the intensity change.

The optimal disparity map D minimizes the energy function. Solving this two-dimensional optimization problem is NP-complete and therefore not feasibly solvable. On the other hand, the 1D version of this optimization can be efficiently solved with dynamic programming. Because solving only for one row at a time results in streaking Hirschmüller used an approach where he aggregated the costs for different directions. He defined a recursive 1D equivalent to Equation 1.2 for a direction $\boldsymbol{r}$:

$$\begin{aligned} L_{\boldsymbol{r}}(\boldsymbol{p}, d) = C(\boldsymbol{p}, d) + \min(&L_{\boldsymbol{r}}(\boldsymbol{p} - \boldsymbol{r}, d), \\ &L_{\boldsymbol{r}}(\boldsymbol{p} - \boldsymbol{r}, d-1) + P_1, \\ &L_{\boldsymbol{r}}(\boldsymbol{p} - \boldsymbol{r}, d+1) + P_1, \\ &\min_i (L_{\boldsymbol{r}}(\boldsymbol{p} - \boldsymbol{r}, i)) + P_2) - \min_i (L_{\boldsymbol{r}}(\boldsymbol{p} - \boldsymbol{r}, i)). \end{aligned} \tag{1.3}$$

This equation consist of a summation of three parts. The first part is the cost of the current pixel and depth combination while the second part contains the transition cost along the path. This follows the scheme from Equation 1.2, where jumps are more penalized than continuous changes.

Without the third part the cost would only accumulate along the path. To guarantee a fixed upper bound for the cost Hirschmüller introduced a regularization term for easier implementation. The minimum cost of the previous pixel is constant for the current pixel and can therefore be subtracted without changing the optimization result.

To approximate $E(D)$ the cost $L_{\boldsymbol{r}}(\boldsymbol{p}, d)$ is calculated for many paths r and then summed:

$$S(\boldsymbol{p}, d) = \sum_{\boldsymbol{r}} L_{\boldsymbol{r}}(\boldsymbol{p}, d). \tag{1.4}$$

Hirschmüller suggested to use either 8 or 16 straight paths. Up, down, left, right, the diagonals and 8 directions in between. Žbontar and LeCun [24] reduced this to the 4 main directions.

To obtain the disparity map the minimum for each pixel is determined as follows: $D_{l\boldsymbol{p}} = \arg\min_d (S(\boldsymbol{p}, d))$. For an increase in accuracy a subpixel interpolation is then performed, where a quadratic curve is fitted to the minimum and the neighbouring depth points. The minimum of this curve equals the depth of this pixel. The subpixel interpolation allows for non integer results of the depth map.

After the generation of the depth map, it can be further refined by post processing steps. To remove small outliers a 3×3 median filter is applied.

A second disparity map for the other image can be computed from the same cost S by taking the baseline into account: $D_{r\boldsymbol{p}} = \min_d (S(\boldsymbol{p}_d, d))$ where $\boldsymbol{p}_d$ means shifting the point along the baseline by d pixels (a horizontal line when the images are rectified).

This second disparity map allows for a left / right disparity check. Disparities where the values do not match are set to invalid:

$$D'_{\boldsymbol{p}} = \begin{cases} D_{l\boldsymbol{p}} & \text{if } |D_{l\boldsymbol{p}} - D_{r\boldsymbol{p}_d}| \leq 1, \\ D_{inv} & \text{otherwise.} \end{cases} \tag{1.5}$$

To remove outliers the resulting disparity image is segmented and segments that are too small are removed. To refine untextured parts they also segment the intensity image and fit a plane into the disparity image of large enough segments. This works simply because most untextured parts are man made and flat e.g. walls.

After the left / right consistency check and the outlier filtering parts of the disparity image are set to invalid. To interpolate these invalid values they are classified into 2 categories: occlusions and mismatches. Occlusions have to be interpolated from the background and can be identified by the left / right check. Mismatches are interpolated from neighbouring valid values using a median filter to preserve discontinuities due to edges.

This interpolated disparity map is the final result of the algorithm.

We use a modified version including pre- and heavy post processing used by Žbontar and LeCun [24]. We describe this version in more detail in Section 1.2.3.

1.2.3 Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches

In this paper Žbonta and LeCun [24] introduce a sophisticated stereo algorithm using machine learning and semiglobal matching. The algorithm and code from this paper is used as basis for the confidence stereo method of this thesis. The paper consists of two main parts: the generation of a cost volume with convolutional neural networks and the subsequent calculation of the disparity image from this cost volume.

For the generation of the cost volume they propose 2 different architectures.

The first one is called the fast architecture, where a siamese convolutional network is used to calculate a similarity score for each pixel of the input images as seen in Figure 1.3 These scores are then compared with the cosine similarity to build the cost volume. To train this network Žbonta and LeCun use pairs of matching and non-matching patches and maximised the difference with the hinge loss.

We decided to use this architecture because we could modify the matching of the similarity scores without retraining the whole network. Our modifications are described in Chapter 2.

The second network is called the accurate architecture. For this architecture they still use the same network for the generation of the similarity scores as in the fast network, but appended an additional network for the matching. They trained this network similar to the fast architecture but changed the loss to the binary cross-entropy.

After generating the cost volume they use different filter and refinement methods.

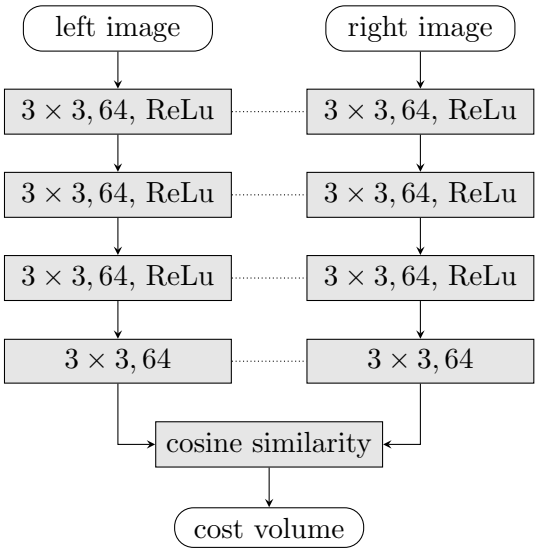


Figure 1.3: The fast architecture: a convolutional neural network to generate a cost volume. The dotted lines indicate shared weights.

Cross-based Cost Aggregation is a method to average the cost volume over a local neighbourhood. The cross-based neighbourhood was introduced by Zhang et al. [25]. This neighbourhood is designed to only include pixels with similar intensities, because these pixels are mostly from the same object.

First a horizontal and a vertical segment is defined for a pixel $\boldsymbol{p}$. The segment is extended from $\boldsymbol{p}$ as long as the difference of the intensities is smaller as a threshold. Additionally a maximum length for each direction of the segment is defined.

The cross-based neighbourhood is defined as the union of all horizontal arms of the pixels in the vertical arm of $\boldsymbol{p}$.

Žbonta and LeCun average the cost volume over this neighbourhood multiple times as the intensities and the neighbourhood changes each iteration.

They apply semiglobal matching in four directions to the cost volume filtered by the cross-based cost aggregation as described in Section 1.2.2.1.

After the semiglobal matching they average the cost volume again with the cross-based cost aggregation and then compute the disparity image $D((p))$ using the the winner takes all strategy:

$$D(\boldsymbol{p}) = \arg\max_{d}(C(\boldsymbol{p}, d)) \tag{1.6}$$

They expand on the left / right consistency check by introducing the differentiation between a mismatch and occlusions where the interpolation value for an occlusion should come from the background. Therefore, they use the next correct value left of the interpolation pixel. For mismatches they use the median of the nearest correct pixels found in 16

directions.

They also use a quadratic curve fitting as subpixel enhancement to increase the accuracy of the disparity map.

As a final step they apply a 5×5 median and a bilateral filter to preserve the edges while smoothing the rest of the disparity map.

1.3 Convolutional Neural Networks

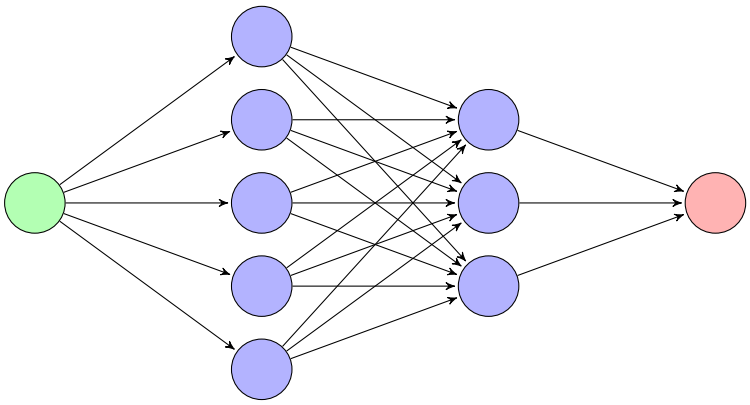


Figure 1.4: The shape of a Neural Network with 2 Hidden Layers. The green node represents the input and the red node the output.

Neural networks are a widely used machine learning method. The idea is to combine many simple operations like summations and multiplications with nonlinearities in between. Information is passed on in a strict feed forward manner. Each step is called a layer and in order to calculate the result of one layer only the preceding layer is needed. Therefore, each individual layer can be calculated in parallel. The topology of a standard network can be seen in Figure 1.4. The equation for each layer is defined as follows:

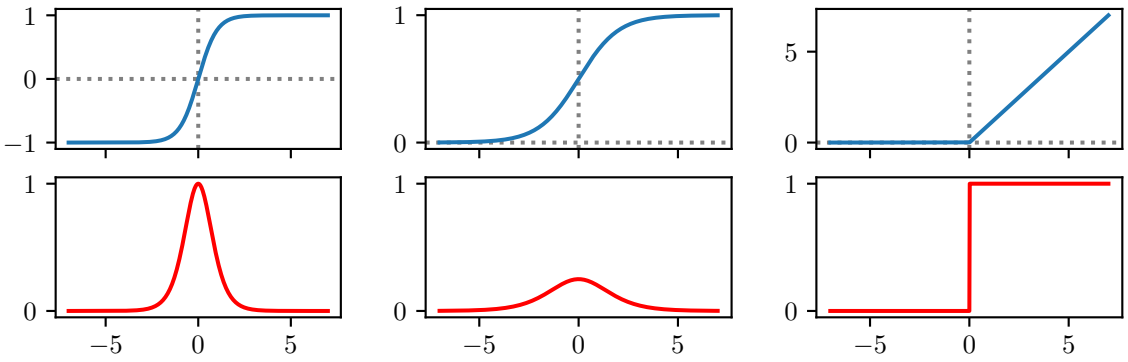
$$g_l^{(a)} = h_l \left(\sum_{b=1}^{B_{l-1}} w_l^{(a,b)} \cdot g_{l-1}^{(b)} + t_l^{(a)} \right), \tag{1.7}$$

where $g_l^{(a)}$ is the output of the a^{th} neuron in layer l, h_l is the nonlinearity of layer l and w and t are the weights that are learned.

The most commonly used nonlinearities are shown in Figure 1.5. In conventional neural networks $\tanh(x)$ is the most widely used function, whereas for convolutional networks the rectified linear units:

$$\mathrm{ReLu}(x) = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{1.8}$$

proofed more powerful and faster to compute and are therefore used in most deep networks,



(a) $\tanh(x)$ **(b)** $\sigma(x)$ **(c)** $\mathrm{ReLu}(x)$

Figure 1.5: Different nonlinearities used in neural networks. The functions are plotted in blue and their derivatives in red.

for example in [4, 10, 16].

The sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ is used if the output of the network should be in the range $[0, 1]$ and is similar to $\tanh(x)$.

Convolutional neural networks replace the single value for each neuron with a 2-dimensional matrix, typically an image. They also change the multiplication by a weight to a convolution with a 2-dimensional kernel. In order to keep the same image dimensions before and after the convolution the edges are padded with zeros.

Often it is useful to reduce the spatial dimensions of the images while increasing the number of neurons in a layer. This gives the network a wider receptive field without increasing the kernel sizes. To reduce the image size by a factor k a stride of k is introduced so that the convolution is only calculated on every k^{th} pixel.

In convolutional neural networks the nonlinearities are often separated into an extra layer type. Apart from the two standard layers convolution and nonlinearity, there are various other types of layers for example batch normalization (BN), deconvolution and max-pooling.

Max-pooling is a layer where the maximum of an input window is used as the output. This, like a stride factor, reduces the spatial dimension.

A technique to learn deeper networks is to introduce a layer that normalizes the values between layers. This special layer is called batch normalization introduced by Ioffe and Szegedy [7]. During learning each mini-batch is whitened so that the mean is zero and the variance is one. Additionally, a scale and shift parameter is learned to guarantee that this extra layer can represent the identity transform and does not change what the network is capable to represent.

Another important layer for our thesis is the deconvolution or upsampling layer introduced by Long, Shelhamer and Darell [10]. This layer is a normal convolutional layer with the forward and backward passes reversed so that the stride of a deconvolution layer

can be used for an increase in image dimensions. This layer can be used in two variants: Either it is learned like any other convolutional layer or the weights are fixed to implement conventional upsampling like the bilinear upsampling.

1.3.1 Deep Residual Learning for Image Recognition

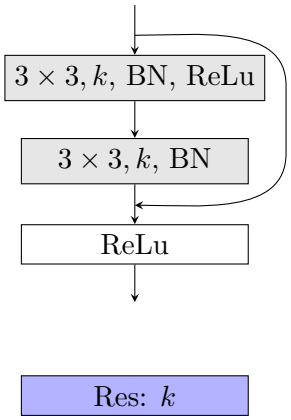
After overcoming the common problems for deep networks like overfitting and vanishing gradients with methods like the batch normalization, the main reason very deep networks are hard to train is the problem of degrading accuracy.

He et al. [4] observed that when adding a layer to a network, the performance of the network decreases. This behavior is not due to the network becoming less capable, since using the identity mapping for the new layer yields the same result as before. They attributed this degradation to the solver and the increased difficulty to train the network.

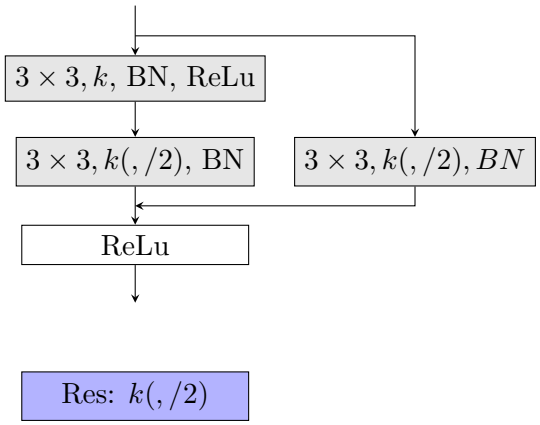
The proposed solution is using shortcut connections and letting the solver only learn residuals, the difference to the identity mapping. They hypothesize this makes training the network easier, because pushing all weights towards zero is easier than learning the identity mapping.

Figure 1.6 shows the proposed architectures and how we denote them in our thesis. A "$3 \times 3, 64, /2$, BN, ReLu" Layer means that we use a 3×3 kernel and 64 channels with a stride of 2. After that we use batch normalization and include a nonlinearity layer, with a rectified linear unit as the nonlinearity.

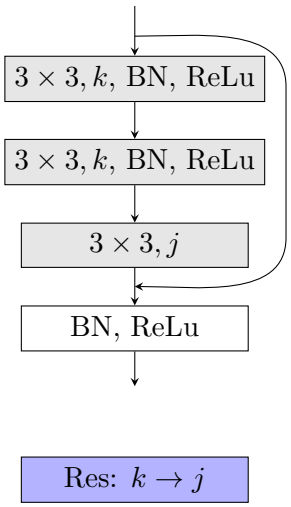
He et al. tested their hypothesis on the ImageNet 2012 [17], an image classification dataset with 1000 classes. First they showed that while going from 18 to 34 layers when using a conventional network the training error increases, while it decreases when a residual network is used. They then introduced 18, 34, 50, 101 and 152 layered networks based on the architecture of [22]. The deeper networks clearly outperformed the shallower ones and the 152-layer network performed better than all other state-of-the-art methods.



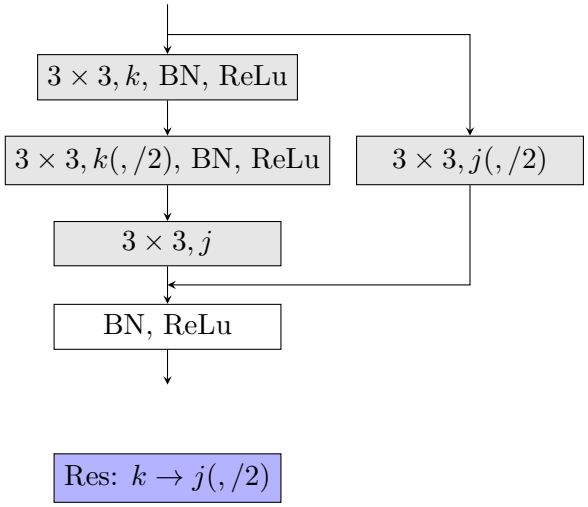
(a) Normal residual layer, where the input and output dimensions are the same.



(b) Normal residual layer, where the number of channels changes or spatial dimensions change. This change is realized with a stride of 2 (, /2)



(c) Bottleneck residual layer with k internal channels, where the input and output dimensions are the same.



(d) Bottleneck residual layer with k internal channels, where the number of layers changes or spatial dimensions change. This change is again realized with a stride of 2 (, /2)

Figure 1.6: Different kinds of residual layers used by He et.al [4] and our more complex networks.

1.3.2 Convolutional Neural Networks for Image Segmentation

Segmenting images is in many ways a similar task to our binary confidence calculation. Therefore, we can use similar network architectures. Instead of having one output channel per segmentation class and assigning a high confidence to one specific class, we have one output channel for each feature channel and assign uncertainties. The main difference is that our confidences are not mutually exclusive and one pixel can have high uncertainties for multiple channels, whereas in the segmentation case one pixel can only belong to one class.

Transfer learning [1] is a method where a fully trained convolutional neural network is retrained for another task. Since new tasks need a different output type, the last layers are often discarded and replaced with new layers suitable for the new task. This retraining has the advantage that fewer training samples are needed to train in comparison with training a completely new network. One explanation why this method works is that the main part of a network works as a feature extractor and these extracted features are universal enough to work for many tasks.

There have been many successful advances with neural networks in the field of image classification [4, 9, 22, 23].

Long et al. [10] utilized this fact and adapted these networks for pixelwise image segmentation. Most networks for an image classification have a similar structure. First they use convolutional layers with strides of 2 to reduce the spatial dimensions further and further. After that a max-pool layer eliminates the spatial dimensions completely and a conventional neural network is used for the final classification.

Long et al. replaced the max-pool layer and the following network with a deconvolution layer to increase the spatial dimension to the original image sizes. They also added 2 additional deconvolution layers in parallel from earlier in the network and combined the output of all three paths in the end. Because the spatial resolution of the earlier layers is higher, these additional paths increased the fine details in the output of the network. With their network they were able to achieve a 20% improvement over the state-of-the-art for the PASCAL VOC 2011 and 2012 datasets.

Ronneberger et al. [16] built upon this idea for the task of segmenting neuronal structures in microscopic images. They changed the architecture by splitting the single upscaling layer into multiple deconvolution layers and adding convolutional layers inbetween. Similar to Long et al. they used the results of earlier layers with higher spatial resolution. However, instead of combining them at the end they combined them when the upscaling path had the appropriate spatial resolution. This enabled the network to combine the more accurate semantic information from the upscaling path with the more accurate spatial information from these shortcut connections.

With their architecture they were able get the top rank on the ISBI EM segmentation

challenge in term of the warping error and also the top rank on the ISBI cell tracking challenge 2015.

Our best performing network uses a similar architecture. We describe it in more detail in Section 4.2.3.

1.4 Confidence Measures in Stereo Matching

Most works about confidence measures for stereo matching do not try to improve the matching result. They instead work to evaluate the quality of the disparity map on a pixel level. One exception is the paper from Park and Yoon [13]. They used learned confidences to modulate the cost volume and managed to improve the matching results. We use a similar approach, with the difference that we use the confidence for the generation of the cost volume and not afterwards. A more detailed overview of the method used by Park and Yoon is available in Section 1.4.1.

There are two different approaches to get confidence values. Traditional ones based on handcrafted measures like the peak ratio or the winner margin and machine learning based ones. A good overview over different traditional methods can be found in [6]. Even most machine learning approaches like [13–15, 21] use handcrafted features, some of which are based on the disparity, or use the disparity directly for their learning algorithms.

A drawback of these methods is that many measures are based on the disparity map itself. Therefore, a preliminary disparity image has to be generated to calculate the confidence values if they are to be used as input for the cost volume.

1.4.1 Leveraging Stereo Matching with Learning-based Confidence Measures

In their paper [13] Min-Gyu Park and Kuk-Jin Yoon introduced a method to incorporate confidence values into the stereo pipeline. They showed that this makes stereo algorithms more robust in difficult situations. An overview of this algorithm is shown in Figure 1.7. They did this by modulating the cost volume for each pixel with the confidence value of this pixel:

$$\hat{C}(\boldsymbol{p}, d) = \hat{Q}(\boldsymbol{p}) C(\boldsymbol{p}, d) + (1 - \hat{Q}(\boldsymbol{p})) \sum_{k \in \mathcal{D}} \frac{C(\boldsymbol{p}, k)}{|\mathcal{D}|}. \tag{1.9}$$

C is the cost volume and $\hat{Q}$ is their confidence value. The sum is simply the mean cost of this pixel $\boldsymbol{p}$. Therefore, this equation scales the cost of one pixel from the original cost if the confidence is 1 to the mean of the cost if the confidence is 0.

In our thesis we expand on this idea and apply our confidence values during the cost volume generation. We do this because not all features that generate the cost volume are equally susceptible to different interferences. Applying our confidence earlier allows us to preserve more information and use features that are undisturbed.

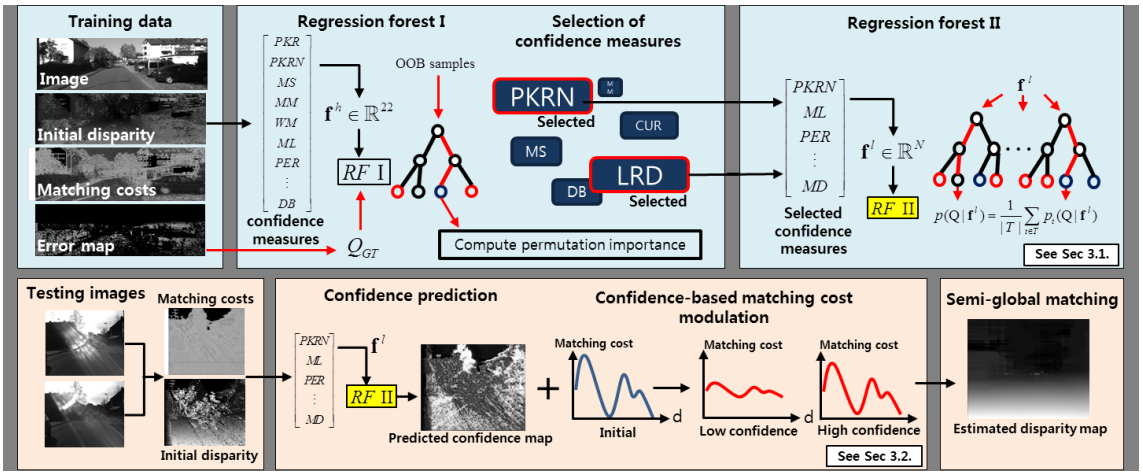


Figure 1.7: A overview of the algorithm used by Park and Yoon. This image is directly from [13]

To generate the confidence values they used a machine learning based multi-step process. Initially they used 22 hand crafted confidence measures using the input images, an initial disparity and the unmodified matching cost. They then constructed a regression forest and based on the results selected a subset of important confidence measures. Finally, they learned a second regression forest with those selected measures which generates the final confidence.

We also use machine learning to generate our confidence, but we omit the step of hand crafting confidence measures and directly learn from the left and right input image. This makes our approach more versatile, but also much more dependent on training data. Therefore, we also propose a method to generate training data and record our own dataset. Instead of regression forests we use convolutional neural networks, because they are designed for working with images.

2 Confidence Values for the Generation of Stereo Images

Contents

Normal feature generation for stereo matching does not distinguish between unique points that result in good matching and points which are similar to many other points or disturbances in the picture that are not part of the real scene like raindrops on the windshield. Park and Yoon [13] showed that modifying the matching cost volume with a confidence value improves matching for scenes under difficult conditions like rain, fog and lens flares.

Different feature channels have different robustness for different disturbances. Therefore, our systems incorporates confidence values during the generation of the matching cost volume on a feature channel level. Figure 2.1 shows the structure of our proposed system.

For the following chapters we omit the x and y position in our equations for better readability. We use these symbols in our equations:

$\boldsymbol{C}$ is the cost volume for the stereo method and C_d denotes the matching cost for a single point (x, y) and one disparity value d.

$\boldsymbol{f}_l$ is the feature vector for the point (x, y) in the left image.

$\boldsymbol{f}_{r,d}$ is the feature vector for the point $(x - d, y)$ in the right image.

$\boldsymbol{u}$ denotes the uncertainty for one point. We use the same index conventions as with $\boldsymbol{f}$.

- $\boldsymbol{u}_d$ is the combined uncertainty of both images and disparity d.
- $\boldsymbol{u}_{ref}$ is the ground-truth uncertainty used for learning.

$\boldsymbol{c}$ is the confidence value corresponding to $\boldsymbol{u}$.

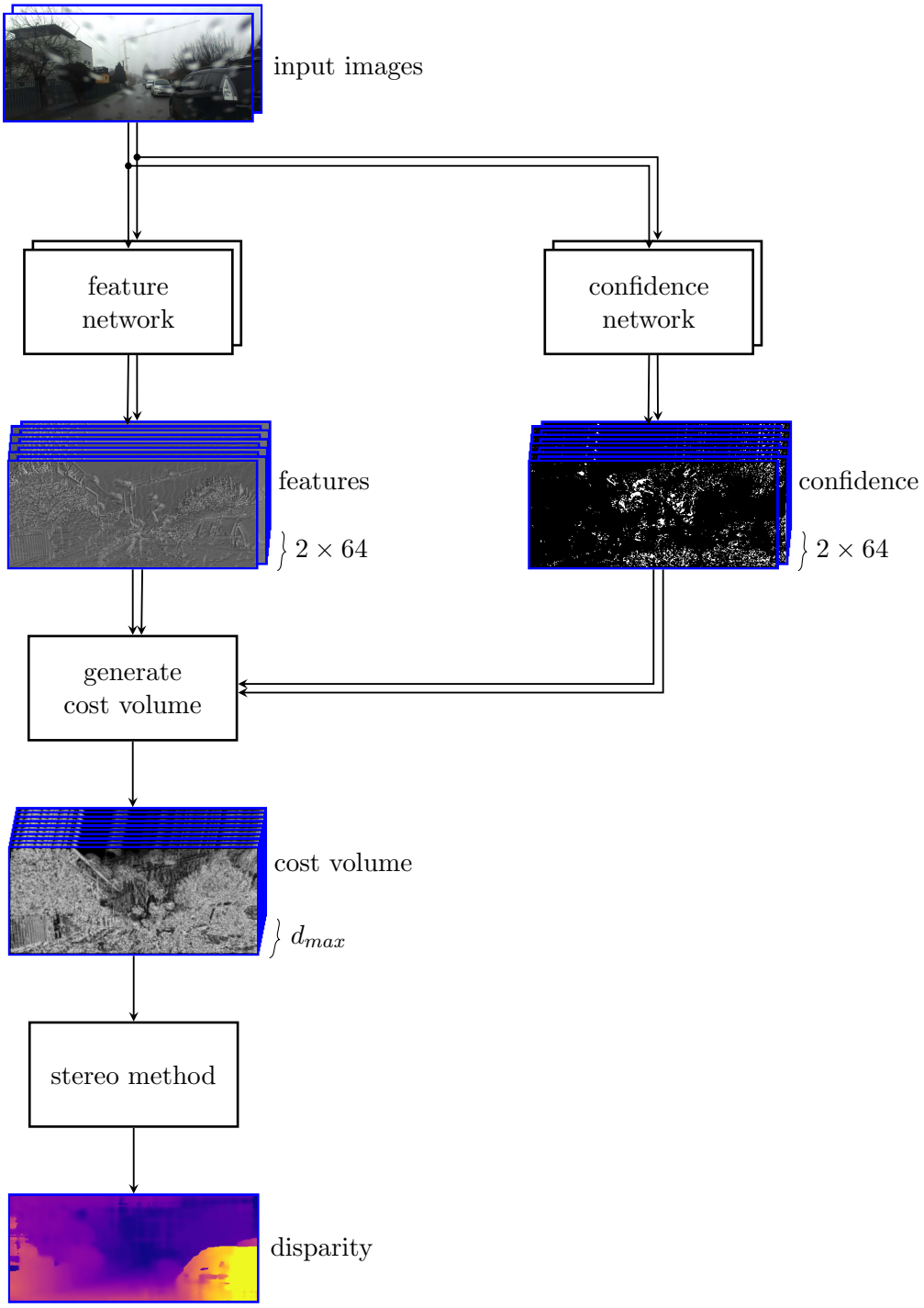


Figure 2.1: The main flow diagram of our proposed stereo algorithm. The left part represents the conventional path and the right part our confidence network.

d_{gt} denotes the ground-truth disparity for one point. We need this ground-truth for learning.

Our general approach for defining uncertainty is to compare one feature channel in one image with the whole corresponding line in the other image. This generates 2 noteworthy points. The first point is the value and position of the minimum corresponding to the disparity that would be calculated using only this feature dimension. The second point is the value and position of the ground-truth.

These points can be compared either by value or by position. For our system we chose the value as basis, because it generate better results and is less dependent on exact thresholds. The drawback of the value comparison is that repeating structures cannot be recognized and accounted for in the uncertainty calculation.

A complete explanation of how we calculate the confidence ground-truth can be found in Section 2.1.1

2.1 Uncertainty with the L2 Norm

Our first idea was to reduce the matching cost of a feature where the uncertainty is high. The value of the L2 uncertainty is dependent on the scale of the corresponding feature channel, which makes a normalization to the range $[0, 1]$ difficult. We avoid this problem by directly comparing the uncertainty with the corresponding feature channel.

To implement this we have to change the feature comparison from the cosine similarity to the L2 norm as seen in Equation 2.1:

$$\boldsymbol{u}_d = \max\left(\boldsymbol{u}_l, \boldsymbol{u}_{r,d}\right), \tag{2.1a}$$

$$C_d = \|\max\left(|\boldsymbol{f}_l - \boldsymbol{f}_{r,d}| - \boldsymbol{u}_d, 0\right)\|_2^2. \tag{2.1b}$$

We use the maximum of the left / right uncertainty as the combined uncertainty $\boldsymbol{u}_d$ as seen in Equation 2.1a which is then used to reduce the matching cost for feature channels with high uncertainty (Equation 2.1b). The maximum operator is needed to prevent negative values due to the uncertainties.

2.1.1 Training

To train a neural network to output the uncertainty we need to generate a training set. We use the feature network to gain feature vectors for the training images. By comparing the optimal match $\boldsymbol{t}_m$ with the true match $\boldsymbol{t}$ we estimate the uncertainty for a point:

$$\boldsymbol{t} = |\boldsymbol{f}_l - \boldsymbol{f}_{r,d_{gt}}|, \tag{2.2a}$$

$$\boldsymbol{t}_m = \min_d\left(|\boldsymbol{f}_l - \boldsymbol{f}_{r,d}|\right), \tag{2.2b}$$

$$\boldsymbol{u}_{ref} = |\boldsymbol{t}_m - \boldsymbol{t}|. \tag{2.2c}$$

The disparity d_{gt} is obtained from the ground-truth disparity of the dataset. A visual representation of these equations is shown in Figure 2.2.

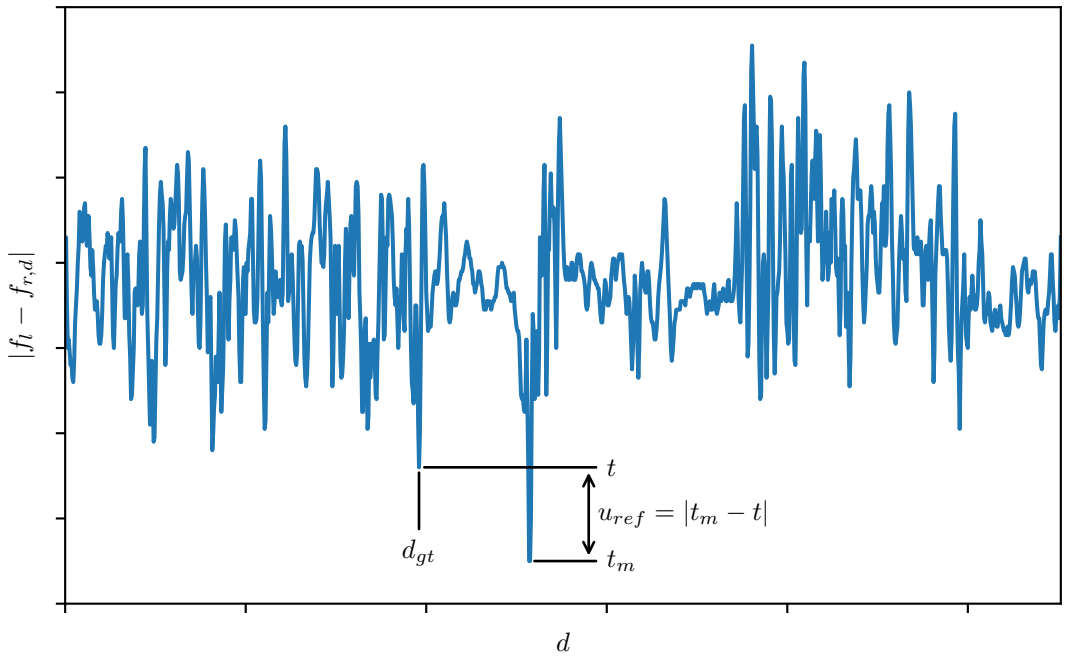


Figure 2.2: A visual representation of Equation 2.2. The blue line is the comparison of one feature of one point in the left image with the features of the whole corresponding line in the right image.

We used this uncertainty $\boldsymbol{u}_{ref}$ as the target for training the neural networks.

2.2 Binary Confidence with the Cosine Similarity

The feature network trained with the L2 norm performed significantly worse than a network trained with the cosine similarity. We therefore chose to keep the cosine similarity and had to overcome the problem of normalizing the uncertainty.

We modified the cosine similarity by modifying both vectors beforehand in the same way:

$$\boldsymbol{c}_d = \boldsymbol{c}_l \circ \boldsymbol{c}_{r,d}, \tag{2.3a}$$

$$\tilde{\boldsymbol{f}}_l = \boldsymbol{f}_l \circ \boldsymbol{c}_d, \tag{2.3b}$$

$$\tilde{\boldsymbol{f}}_{r,d} = \boldsymbol{f}_{r,d} \circ \boldsymbol{c}_d, \tag{2.3c}$$

$$C_d = \frac{\tilde{\boldsymbol{f}}_l \cdot \tilde{\boldsymbol{f}}_{r,d}}{\|\tilde{\boldsymbol{f}}_l\|_2 \|\tilde{\boldsymbol{f}}_{r,d}\|_2}. \tag{2.3d}$$

$\boldsymbol{a} \circ \boldsymbol{b}$ denotes the hadamard product (element-wise multiplication) of two vectors.

For this to work the uncertainty has to be inverted to represent a confidence value. This is problematic, because our in Equation 2.2 generated uncertainty is not bounded to a specific range. The range of $\boldsymbol{u}_{ref}$ is heavily dependent on the range of $\boldsymbol{f}$, which can change for each feature channel.

We tried different approaches for this inversion:

The obvious method for inversion $1/u$ does not work, since an uncertainty of 0 would result in an infinitely high confidence. This is a theoretically reasonable assumption, but hard to use in practice.

Our next approach was to find the maximum and scaling to the range $[0, 1]$: $1 - \boldsymbol{u}/u_{max}$. This resulted in an overall very high confidence and was therefore not practical. We also tried using a different u_{max} for each feature channel, but the effect was the same.

To counter the few points with very high uncertainty we introduced an arbitrary u_{max} and clamped all higher values of $\boldsymbol{u}$ to u_{max}. This was the first approach that outperformed the L2 approach in our sanity check as described in Section 4.1. We empirically determined good values of u_{max}. Very low values performed much better, and we found that most of the features with any uncertainty at all were mapped to zero confidence.

This effect is due to the fact that features with uncertainty have no valid information at all. Instead, they would result in completely wrong disparity values if used. Therefore, our previous approach to use continuous values for the uncertainty was not ideal.

As a result of these findings we used an empirically determined value u_{thr} as threshold and all values of $\boldsymbol{u}$ higher than u_{thr} were assigned a confidence of 0 and all other values a confidence of 1. Because of the different ranges of the feature channels we used a different threshold for each channel. 64 feature channels were to much for a exhaustive grid search and gradients are difficult do determine, therefore we implemented a genetic algorithm to find reasonable thresholds. The population was chosen as 10 and as new population we chose the 2 best thresholds, the average and random variations of these 2 best thresholds. After 1000 generations good values for the thresholds were found.

With these binary confidence values we selectively disable single feature dimensions during the matching. In case that the complete feature vector is set to 0, the smoothness constraints of the semi global matching are used to generate a sensible disparity value.

2.2.1 Training

We generated the target in the same way as the L2 uncertainty using an additional threshold at the end to obtain binary targets. To decrease training time we pre-generated the targets and saved them to the disk.

Generating clear uncertainties using only images with disturbances is very hard, because it is unclear if the left or the right image caused high uncertainty $\boldsymbol{u}_{ref}$. To ease learning we generated the uncertainties with one clean and one disturbed image. This resulted in much faster network convergence and better output.

As the training method we used stochastic gradient descent with momentum and

updated the gradient after each image. For most networks we used a momentum of 0.9, 300 epochs and a learning rate of 0.001. After 2/3 of the epochs, we reduced the learning rate by a factor of 100 to fine tune the network.

We trained all our networks on a CUDA server with 16 Nvidia Tesla K80 graphics cards with 12 GB of ram, 4 Intel Xeon CPU E5-2630 v3 CPUs and 126 GB RAM.

We trained multiple different network architectures on our own dataset. The results of these networks are shown in Chapter 4.

3 Rain Dataset

Contents

3.1 Goal of the Dataset

To test our method we needed a dataset of images with disturbances. We decided to focus on a single disturbance for this dataset and chose raindrops on the windshield, because many stereo rigs are mounted behind the windshield of a car. Meister et al. [11] also used this approach.

We propose a method to record a dataset with disturbances and a possibility to generate a pseudo ground-truth. We capture two image pairs from the same scene, one with and one without disturbances. The pseudo ground-truth can be generated from the clean image and the confidence values can be trained with the other image pair.

With our method it is possible to record a reasonable sized dataset in a relatively short time using only a car, a stereo camera and a recording device.

Figure 3.1 shows an example image from our dataset. The top row contains the clean images without disturbances and the bottom row shows the images with raindrops on the windshield. The middle row contains the disparity maps, where warmer colors denote nearer objects and cooler colors more distant pixels. The left disparity map is generated from the clean images and the right image from the images with raindrops. Both images are generated with the unmodified algorithm from Žbonta and LeCun [24]. The right disparity map contains blatant errors introduced from the raindrops.

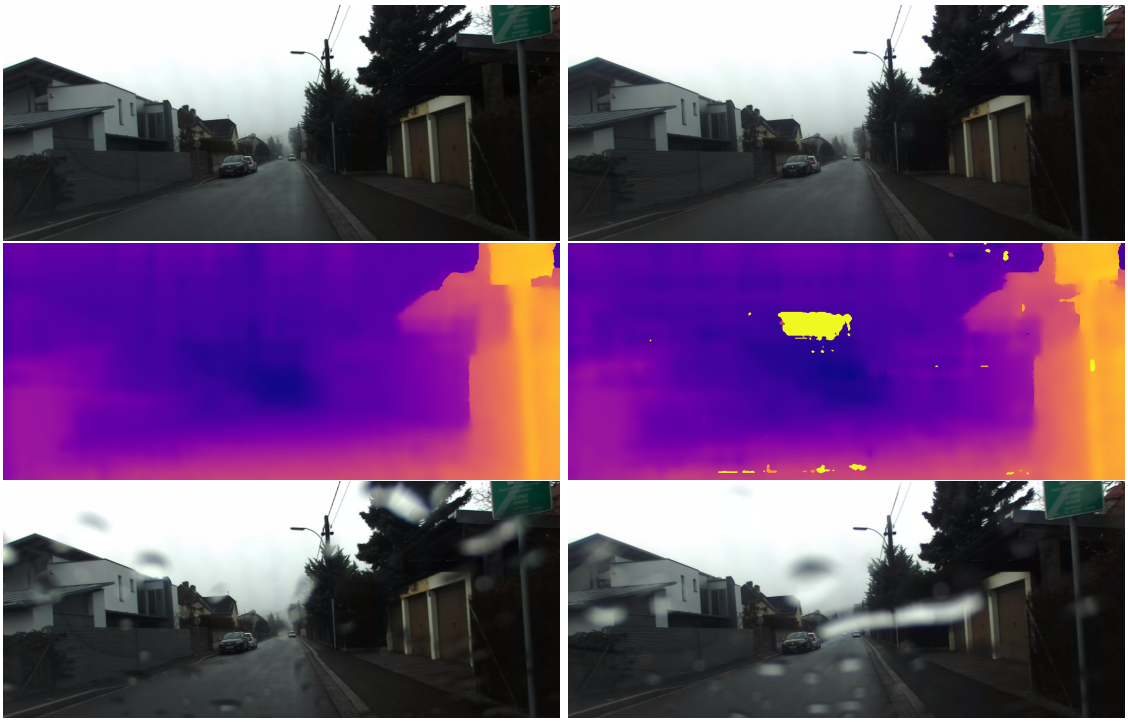


Figure 3.1: A comparison between images and corresponding disparity maps with and without disturbances.

3.2 Dataset Acquisition

Figure 3.2: The tools used for the image acquisition.

Before the actual image acquisition we recorded a series of calibration images as seen in Figure 3.3. We used a calibration toolbox from the institute provided by Ferstl et al. [2] based on the Camera Calibration Toolbox for Matlab[1] from Caltech. These images are necessary to calibrate the camera pair and rectify the images of the dataset.

[1] `http://www.vision.caltech.edu/bouguetj/calib_doc/`

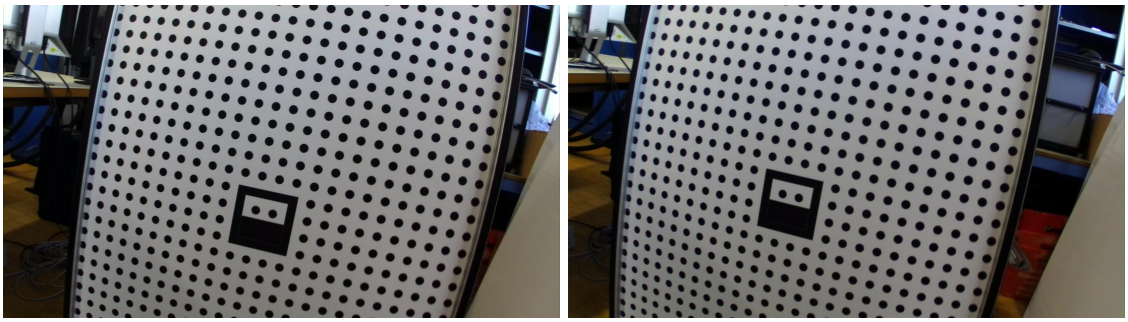


Figure 3.3: Calibration target we used for camera calibration as seen from the left and right camera.

To acquire a pseudo ground-truth we had to record an image pair without the raindrops visible to run the unmodified stereo algorithm [24] on. Therefore, we had to stop the car to record multiple images with the same scene. We used two different recording sequences depending on the weather. For rainy weather we used:

1. Wait until raindrops have gathered on the windshield and record one image pair with raindrops.
2. Record multiple images with as few raindrops as possible, by running the windshield wiper.

When it was not raining our sequence was as follows:

1. Record a clean image pair for the pseudo ground-truth.
2. Use a squirt gun to add drops to the windshield and record one image pair with raindrops.

For the images with real rain we had to fuse the multiple images with minimal raindrops to get one clean image using a combination of the median filter and averaging. For each pixel and color we sorted the values and used the average of the middle third as the clean value.

This resulted in good clean images for the generation of the pseudo ground-truth.

To generate the disparity maps we used the unmodified stereo algorithm from Žbonta and LeCun [24] with the clean versions of our image. We disabled post processing steps in which new data would be generated. This was necessary to only generate data on which we could rely on. This included the interpolation step, subpixel enhancement and the median and bilateral filters. Afterwards we discarded pixels where the left-right consistency check or the occlusion test failed.

We used two different cameras for the acquisition: a stereo camera rig available at the institute with an baseline of 50 cm and a ZED camera from Stereolabs[2] with a baseline of 12 cm.

[2] `https://www.stereolabs.com/zed/`

Figure 3.4: Comparison between details of clean and dirty images for an image on a rainy day. Left is one image with as few raindrops as possible to generate a clean image. The middle image is the generated clean image. On the right the dirty image with raindrops present.

Figure 3.2 shows an image of our capture setup with the ZED camera. We captured 251 stereo image pairs over a period of multiple days. Non of these days had rain, therefore we decided to also include the 45 pairs with the other camera setup in our final dataset.

The locations where we captured images had to fulfil a few criteria: They had to be relatively quiet to capture 2 images without changing fore- or background. We also had to be careful not to obstruct traffic. Another criterion was the distance to drive. Our Laptop battery only lasted 2 - 3 hours therefore it was important that the next charging point was not too far away. We managed only two acquisition runs per day because of the long charging time of the laptop.

Based on these criteria we chose quieter neighbourhoods around Graz. This introduced of course a bias into our dataset, but we believe the impact on the results is minimal.

We only got one day with real rain and it was only a light drizzle. Therefore, the raindrops on our windshield were not particular big. We tried to emulate this light rain with our squirt bottle as well as heavier rain. Figure 3.5 shows the comparison of real and simulated rain in our images.

(a) real rain

(b) simulated light rain

(c) simulated heavier rain

Figure 3.5: The difference between real rain and simulating it by squirting drops onto the windshield.

3.3 Dataset

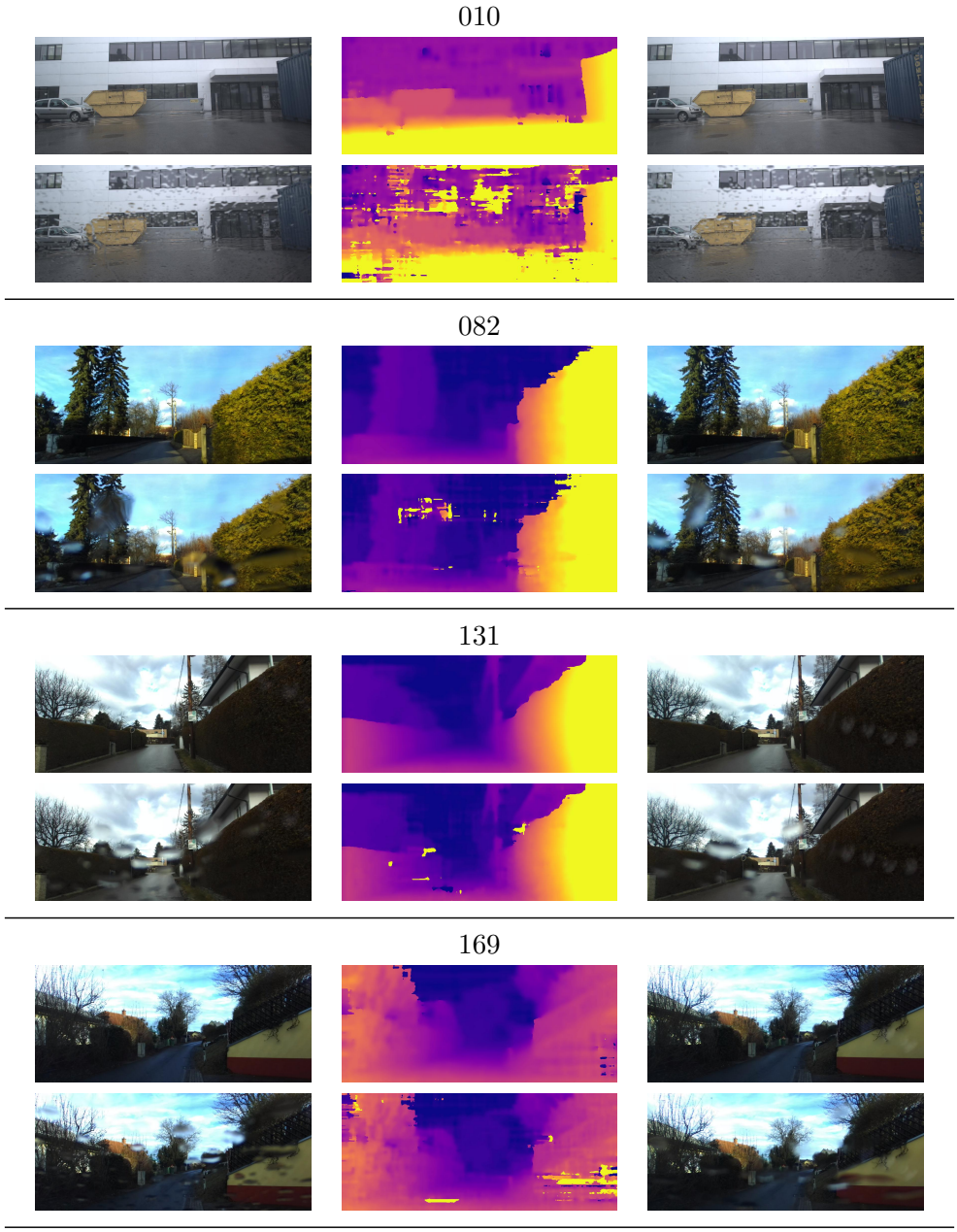
The complete dataset contains 278 scenes, each containing 4 images and one disparity map. The images have a resolution of 1104×468 and contain the left and right image with and without raindrops. The maximum disparity is 150 pixels.

We randomly split the dataset into a training and a validation set, but we ensured that both sets contained images with real rain. Our training set contains 242 scenes and the remaining 55 scenes are for validation. The exact indices are shown in Table 3.1

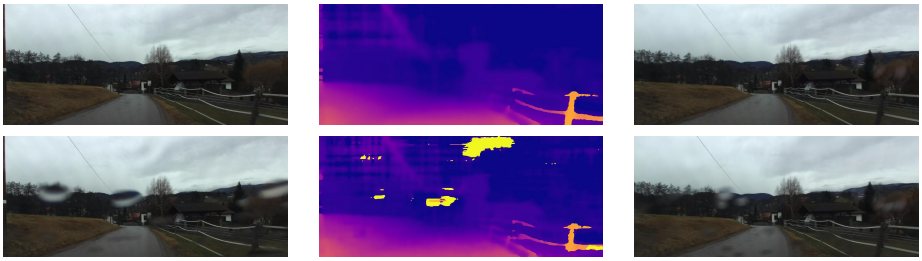
training	22 39 12 34 3 11 18 9 36 29 44 25 38 43 6 45 21 35 13 31 26 40 8 10 28 19 33 0 16 5 37 17 27 7 24 15 4 32 1 30 84 182 101 250 262 193 190 266 239 295 216 172 143 204 125 293 284 244 158 119 115 165 121 159 164 166 285 130 75 214 213 279 51 102 222 273 154 111 183 173 261 196 77 58 81 74 88 267 68 139 230 150 97 160 141 286 247 122 87 282 200 124 277 72 289 243 236 185 46 48 123 92 146 291 198 227 136 131 223 144 82 107 210 187 57 292 246 73 201 50 168 78 192 226 184 108 181 174 274 270 116 240 110 90 296 86 169 199 69 258 231 127 85 257 93 140 238 207 89 191 219 49 151 99 179 275 241 280 95 209 126 80 254 53 217 156 137 129 271 276 135 54 59 105 263 177 63 212 118 268 180 252 278 109 100 153 96 255 220 287 235 294 253 269 215 104 94 134 67 103 249 206 290 233 237 175 83 203 283 47 98 195 176 197 149 145 162 133 248 120 260 256 167 297 66 234 117 152 60 138 225 148
validation	23 14 41 42 2 20 211 52 157 218 161 229 245 281 55 76 106 264 272 205 114 194 242 61 147 70 132 65 163 170 79 288 232 265 56 208 186 142 171 128 259 71 155 64 251 228 113 178 202 189 91 224 62 112 188

Table 3.1: Indices of our dataset split into training and validation set.

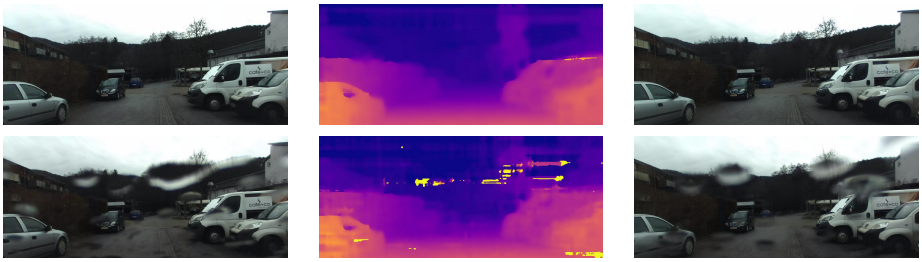
3.3.1 Examples of Images in the Training Set



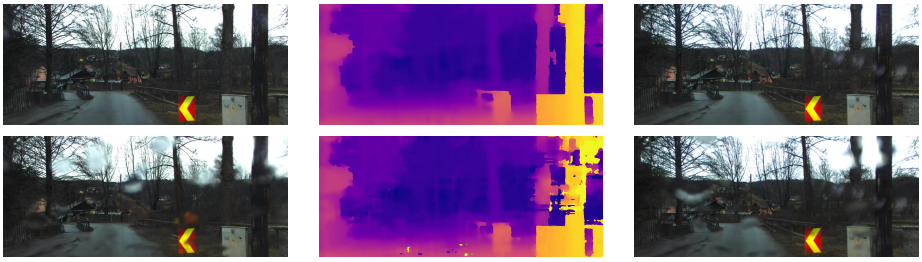
181



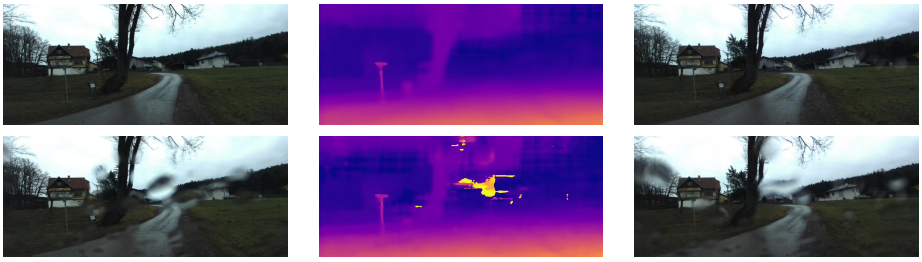
204



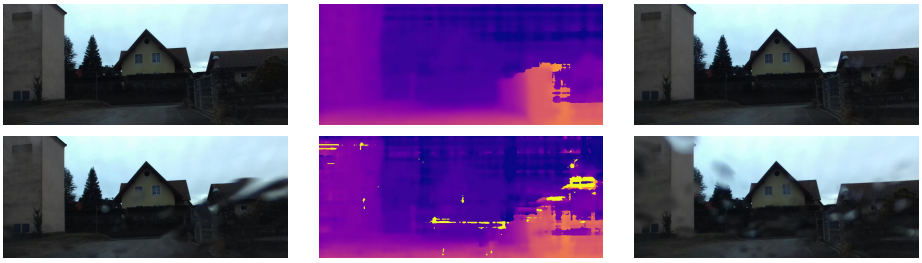
207



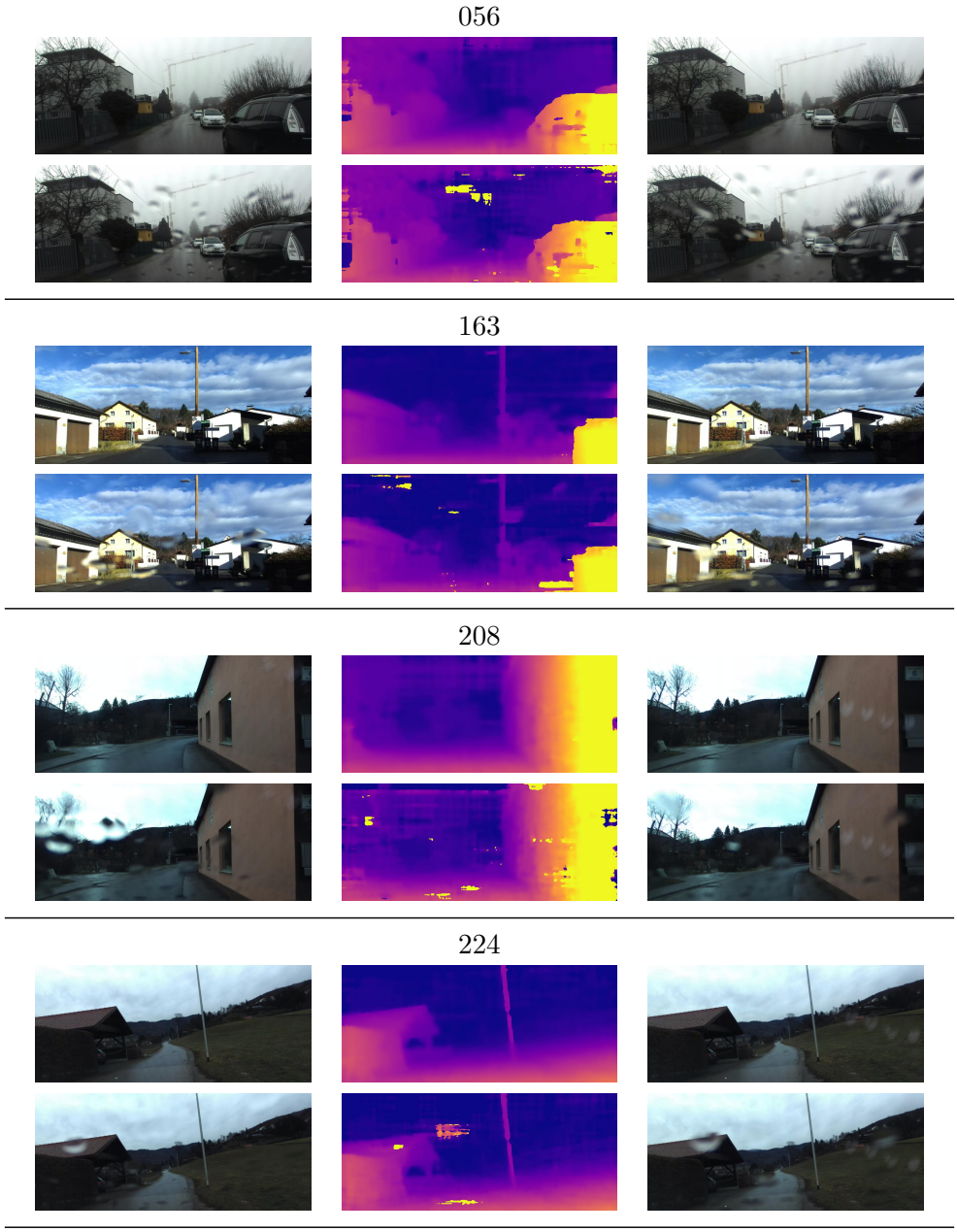
213



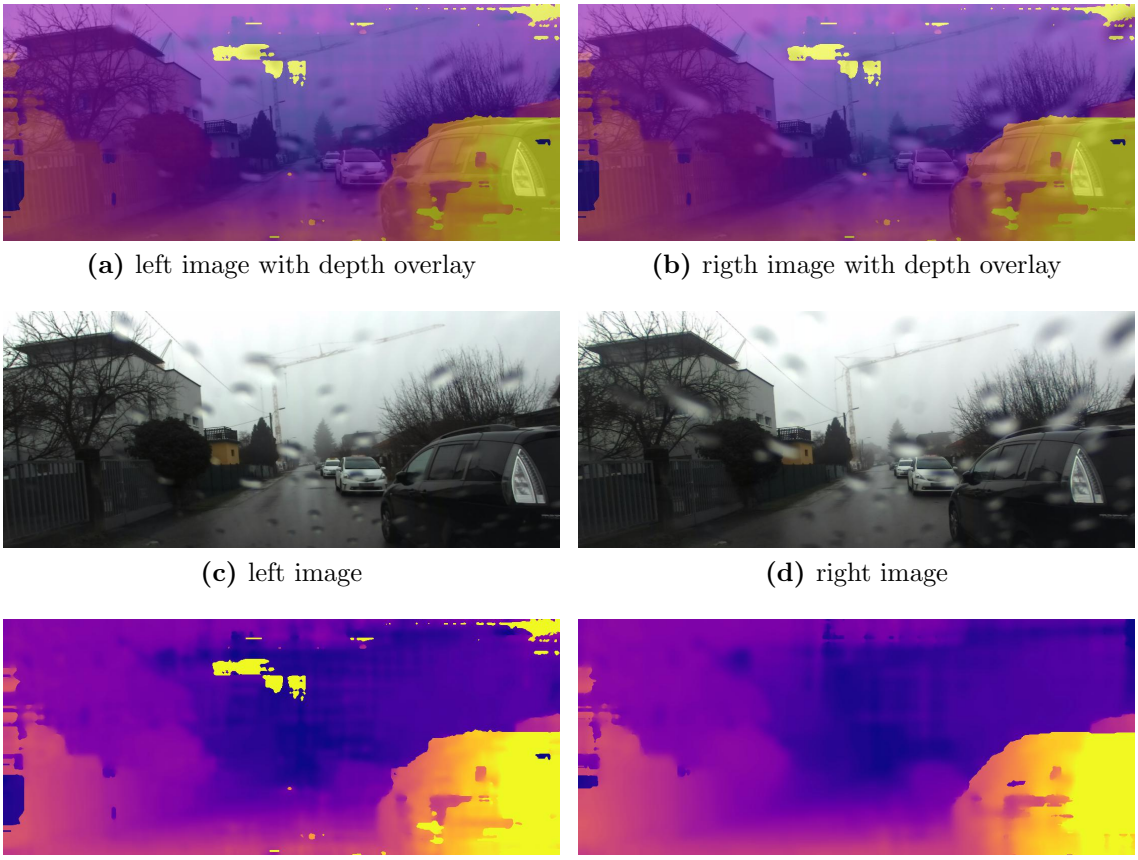
235



3.3.2 Examples of Images in the Test Set



3.4 Errors Introduced by Raindrops



(a) left image with depth overlay

(b) rigth image with depth overlay

(c) left image

(d) right image

(e) depth map with errors

(f) clean depth map

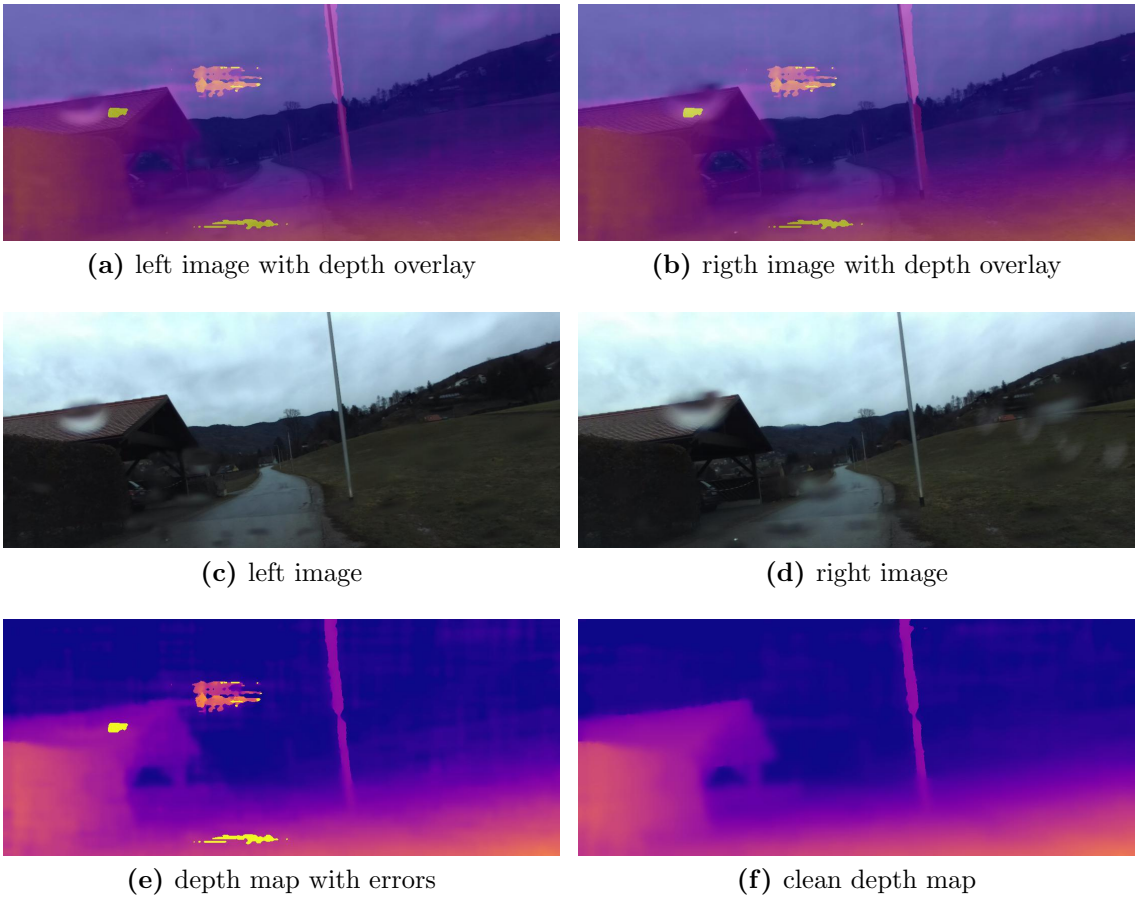
Figure 3.6: Comparison between the clean and the erroneous disparity map for image 56. Most errors occur if raindrops are in similar positions on both images.

There are two problems associated with raindrops.

The first one is that the raindrops occlude or at least blur the real scene. This makes accurate matching of fine details very hard. Lower frequency features can however be detected through the blurring of the raindrops.

The second problem is that two different drops can look very similar and can therefore be matched. This introduces error in the form of blobs with very high disparity values. In real world applications these errors could be interpreted as obstacles that are right in front of the cameras. In the case of an automotive application this could lead to an emergency braking with no real obstacle in the way.

An example of this type of error can be seen in Figure 3.6e and 3.7e. Our confidence network can reduce these errors – for example the errors in the middle have been eliminated by our system (Figure 4.9).



(a) left image with depth overlay

(b) rigth image with depth overlay

(c) left image

(d) right image

(e) depth map with errors

(f) clean depth map

Figure 3.7: Comparison between the clean and the erroneous disparity map for image 224. Small differences due to raindrops can lead to big errors on flat surfaces, in this case the sky.

Another side effect is that the location of the error is not always consistent with the location of the raindrops in one image, because the error is caused by a raindrop in the other image. This effect can be seen in Figure 3.7a.

4 Experimental Results

4.1 Sanity Check

To verify if the generation of the confidence training data and our system to modify the features can improve the disparity we made a simple sanity check. We used the confidence generated from the disparity pseudo ground-truth instead of the output of the neural network. This way we could quickly estimate the best possible performance of different confidence algorithms without training a neural network.

	training		test	
	error	reduction	error	reduction
Csim no confidence	4.75%	0%	4.65%	0%
L2 no uncertainty	73.59%		75.45%	
L2 uncertainty	12.41%		10.49%	
Csim confidence $1 - \boldsymbol{u}/u_{max}$	1.13%	76.2%	1.14%	75.5%
Csim confidence with trained threshold	0.87%	81.7%	0.86%	81.5%

Table 4.1: Results of different confidence methods with the sanity check on the rain dataset.

We used this method to evaluate the different methods in Chapter 2 and as seen in Table 4.1 the cosine similarity significantly outperforms the L2 norm. As error percentage we used the number of disparity pixel that are more than 3 steps different than the ground-truth over the number of pixel overall. To visualize the improvements better we calculated the error reduction over the basis stereo method as reduction $= {}^{(\text{error}_{ref} - \text{error})}/_{4.75\%}$. We got the best results with the trained threshold described in Section 2.2 and therefore only used this confidence to train our neural networks.

Even with the best confidence generation it is not possible to generate perfect disparity maps, because we use the confidence only to disable features and not to generate new data. Therefore, when rain drops completely occlude the scene we have no information about the scene in this location and have to use the smoothness constraints of the stereo method.

4.2 Network Architectures

While designing the confidence network architecture we had to consider the following constraints:

- The output of our network had to be binary.
- The image dimensions of the input and output had to be the same.
- The number of output channels had to match the number of channels of the feature network.
- The network had to be small enough to be trained with the available hardware.

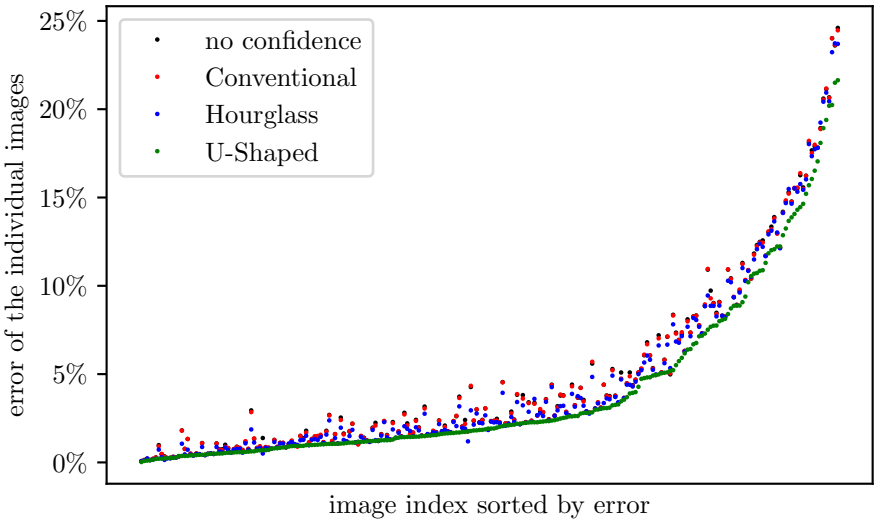
We solved the binary output on all our networks by using a sigmoid layer during training and thresholding the output of this layer during testing.

For the second requirement we used three different approaches, detailed in the following sections:

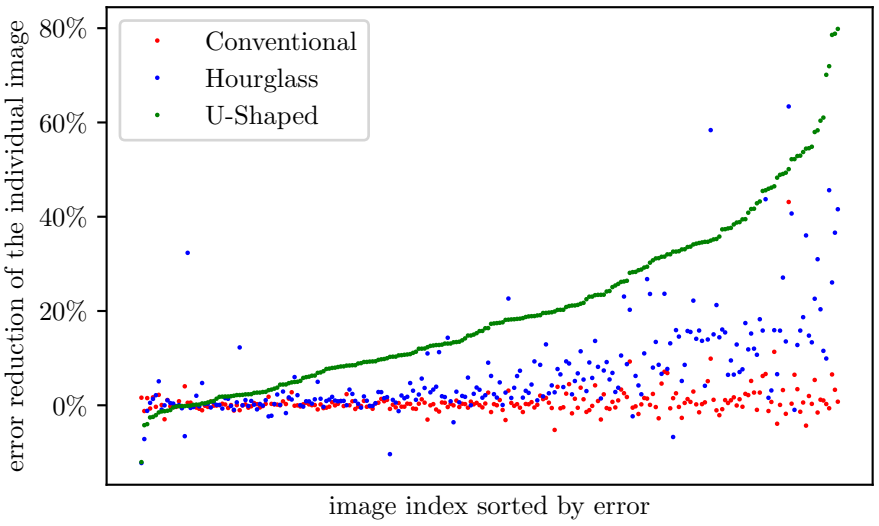
- Keeping the image dimensions constant over the whole network. We achieved this by padding each convolution and having a constant stride of 1.
- Gradually downscaling the image dimensions and then subsequently upscaling. This results in an hourglass structure.
- Using the hourglass network and introducing shortcut connections for each downscaling layer. These network architectures are called U-shaped.

We describe these architectures in more detail in the following sections.

Figures 4.1 and 4.2 show the percentage of wrong pixels in the disparity map for each image individually in the rain dataset. The image indices in each graph are sorted by the error of the U-shaped network since the U-shaped architecture outperforms the others by a large margin.

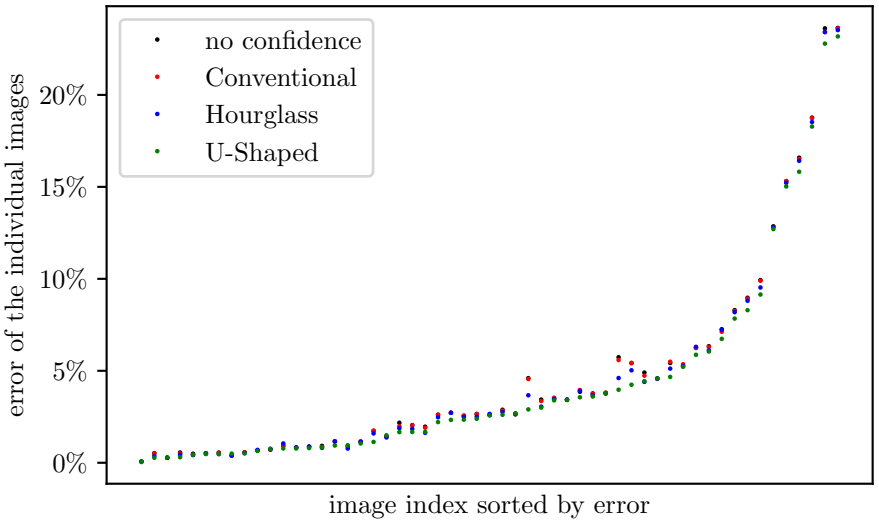


(a) Percentage of wrong pixels in the disparity map.

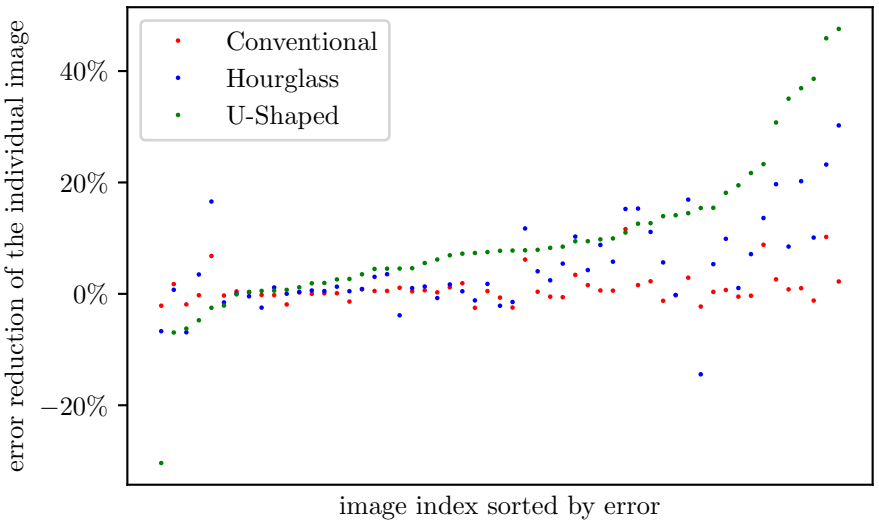


(b) Improvement of the error percentage over the result with no confidence.

Figure 4.1: Results of the different network architectures for each image in the training set. The indices are sorted by the performance of the U-shaped network.



(a) Percentage of wrong pixels in the disparity map.

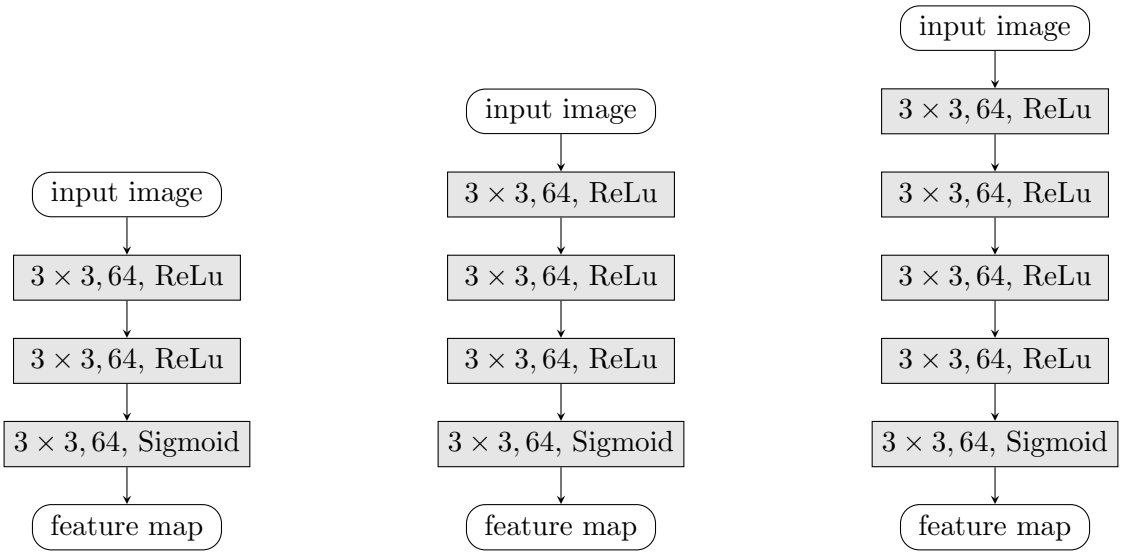


(b) Improvement of the error percentage over the result with no confidence.

Figure 4.2: Results of the different network architectures for each image in the test set. The indices are sorted by the performance of the U-shaped network.

4.2.1 Conventional Neural Networks

The first architecture we tried was the same as the feature network used by [24]. We tried using a different number of layers, as seen in Figure 4.3.



(a)

(b) The feature network.

(c)

Figure 4.3: Simple neural networks based on the network used for the feature generation.

This architecture proved to be way too simple for our task. As seen in Figure 4.4 the output is barely visible, because the networks are not able to represent the complex shapes.

	training		test	
	error	reduction	error	reduction
no confidence	4.75%	0%	4.65%	0%
3-layers	4.75%	0.00%	4.65%	0.00%
4-layers	4.73%	0.42%	4.62%	0.65%
5-layers	4.74%	0.21%	4.64%	0.22%

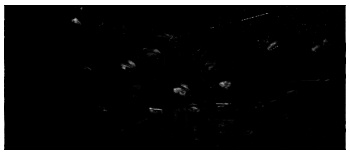
Table 4.2: Results of the trained conventional networks on our rain dataset.

However, we could show that the performance does not decrease with our extension even if the network is too basic to learn the confidences correctly. Figure 4.4 shows the averaged confidence output of this network compared with the ground-truth. Only the 4-Layer network was able learn some very high confidence values, but these were not enough to make a difference in the calculation of the disparities. These networks have 74, 111, and 148 thousand parameters (for the 3, 4, and 5 layer network).

Table 4.2 shows the results on the training and validation dataset. The results of the 5-layer network are worse than the 4-layer ones because the training parameters were not optimally chosen, since we decided to focus on the more promising complexer architectures.

(a) reference

(b) 3-layers



(c) 4-layers

(d) 5-layers

Figure 4.4: Mean images of the trained output-confidences of the conventional networks.

4.2.2 Hourglass Residual Networks

Previous works show that reducing the spacial dimensions and using pre-trained networks is advantageous to increasing the network performance [1, 4, 10].

We decided to use a Residual Network from [4]. Trained versions for Torch7 were replicated by Gross and Wilber[1][2]. Considering memory constraints and reduction in spacial accuracy we used the 50-layer version (ResNet-50) and truncated it right before the forth downscaling layer. Because of the requirement of the same spatial input and output dimensions we added an upscaling path to this pre-trained truncated network. Figure 4.5 shows the three different upscaling networks we used. For the first network (Figure 4.5a) we upscaled the output of ResNet-50 and adjusted the channels with a convolutional layer. While outperforming the standard networks in Section 4.2.1, this network itself was outperformed by the two other networks. With 1.5 million parameters this network has 10 times more parameters than the 5-layer one and is therefore much more complex.

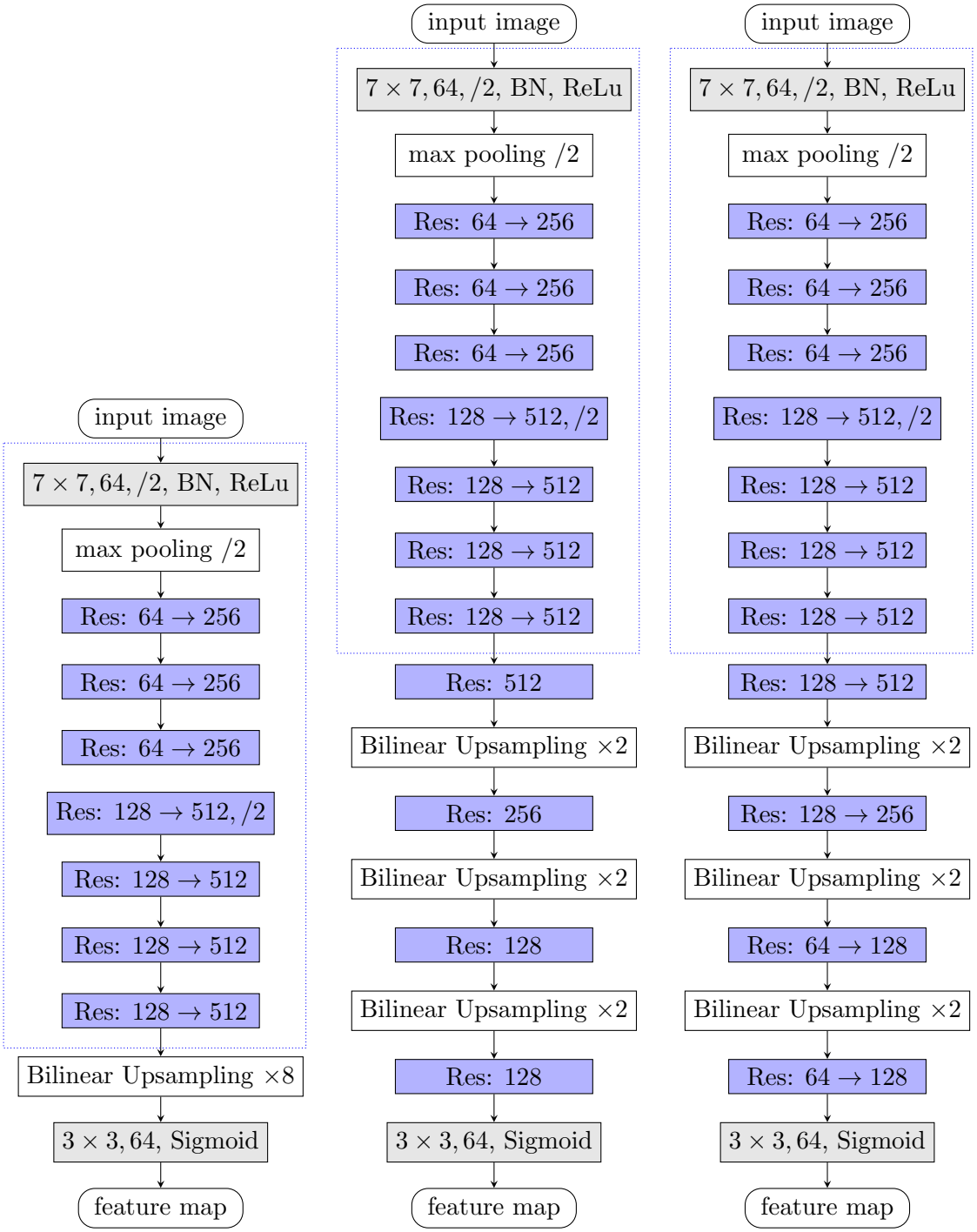
The second and third network (Figure 4.5b and Figure 4.5c) differ only in the type of residual layers used. For these networks we only upscale by a factor of two and use residual layers to reduce the number of channels gradually. The two networks performed very similar as seen in Table 4.3. While the numbers suggest similar performance, the output of the non-bottleneck architecture seems more detailed as can be seen in Figure 4.6. This can be explained by the difference of parameters: the non-bottleneck has 8.9 million parameters and the bottleneck only 2.3 million. Another difference is that while the

[1] `http://torch.ch/blog/2016/02/04/resnets.html`
[2] `https://github.com/facebook/fb.resnet.torch`

bottleneck architecture is faster to compute it uses more memory.

	training		test	
	error	reduction	error	reduction
no confidence	4.75%	0%	4.65%	0%
4-layers	4.73%	0.42%	4.62%	0.65%
direct	4.65%	2.10%	4.56%	1.94%
non-bottleneck	4.56%	4.00%	4.51%	3.01%
bottleneck	4.58%	3.78%	4.51%	3.01%

Table 4.3: Comparison of the results of the hourglass residual networks and the conventional network for the rain dataset.



(a) Hourglass network without layers in the upscaling path.

(b) Hourglass network without bottleneck layers in the upscaling path.

(c) Hourglass network with bottleneck layers in the upscaling path.

Figure 4.5: Hourglass networks. The blue bordered parts are taken from the trained ResNet-50.

(a) 4-layers **(b)** reference

(c) direct **(d)** non-bottleneck **(e)** bottleneck

Figure 4.6: Mean images of the trained output-confidences of the hourglass networks.

4.2.3 U-Shaped Residual Networks

The drawback of the hourglass architecture is the loss of small details. This can be partially circumvented by using shortcut connections. This architecture is based on the U-Net from Ronneberger et. al [16] that is explained in more detail in Section 1.3.2.

We can therefore use a larger part of the ResNet-50 and truncate it before the fifth downsampling layer. Figure 4.7 shows the complete layout of the network. The arrows meeting lines indicate stacking the two inputs in the channel dimension as the image dimensions are the same. This network uses large amounts of memory, therefore we chose to use non-bottleneck residual layers in the upscaling path. With 26.8 million parameters this was the largest network we could reasonable train, but training took about 6 days.

As seen in Table 4.4 this architecture significantly improves the results.

	training		test	
	error	reduction	error	reduction
no confidence	4.75%	0%	4.65%	0%
4-layers	4.73%	0.42%	4.62%	0.65%
non-bottleneck	4.56%	4.00%	4.51%	3.01%
U-shaped	4.02%	15.37%	4.33%	6.88%

Table 4.4: Results of the U-Net in comparison to the previous networks for our rain dataset.

The output of the network is far from perfect, but Figure 4.8 shows a great improvement over the other architectures.

In Figure 4.9 the disparity output of the U-Shaped network is shown. Even with only about 7% improvement many of the more severe errors are eliminated by this network. Our confidence eliminates all errors in the middle of the image caused by the raindrops.

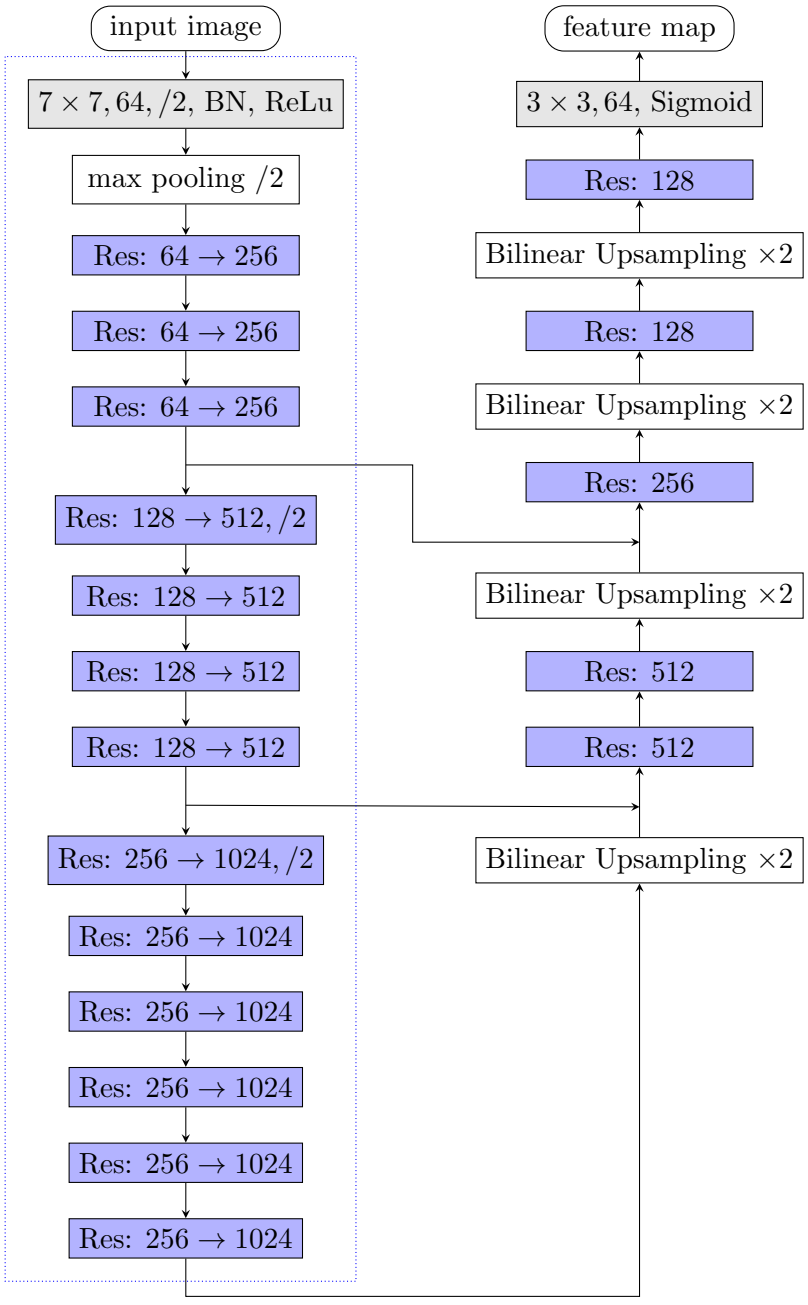
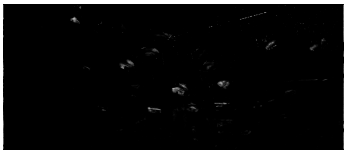
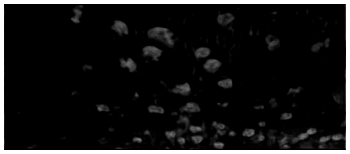


Figure 4.7: U-shaped network based on [16]. Similar to Figure 4.5 the blue bordered parts are taken from the trained ResNet-50.

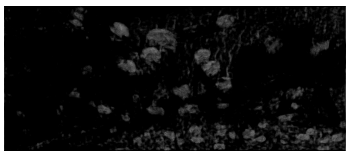


(a) 4-layers



(b) non-bottleneck

(c) reference



(d) U-shaped

Figure 4.8: Mean images of the trained output-confidences of the U-shaped network compared to the other networks.

(a) clean image

(b) no confidence

(c) U-shaped

Figure 4.9: Output of the stereo-method with our confidence extension.

5 Conclusion

We proposed a system to apply confidence values to the generation of the cost volume in a stereo pipeline that incorporates confidence values earlier than other systems like [13]. Therefore our confidence values can selectively disable features that are more susceptible to a certain disturbance and use the remaining features for a still accurate stereo matching.

We trained deep convolutional neural networks for the generation of these confidence values with the raw input images as input. To train these networks we proposed a way to generate confidence targets from the unmodified cost volumes and ground-truth disparities.

There are no datasets for stereo matching that match all criteria to train our networks. Either the datasets have almost no disturbances like KITTI [3, 12], no ground-truth disparities [11] or are too small and have to little variance to train neural networks [8]. Therefore we proposed a method to record a dataset with disturbances and pseudo ground-truth in a way that a large enough number of images for training neural networks can be achieved. We then recorded our own dataset with 278 street scenes with and without raindrops in suburban areas around Graz. We showed that with our method of generating confidence targets and using these confidence values in the generation of the cost volume we could reduce the error by 81%.

Our best trained neural network architecture was a combination of a residual network and U-shaped networks. We used stochastic gradient descent with momentum and utilized transfer learning to train our networks. The network was able to reduce the error by 7%. This indicates the possibility for much more improvement with better suited machine learning algorithms.

Even with this low percentage our network was able to eliminate many severe errors that were introduced by the rain drops.

Deep convolutional neural networks benefit greatly from more training data. Therefore, increasing the number of recorded training scenes would result in better confidence networks and increase the performance of our method. Also, more architectures of neural networks and training methods can be explored or different machine learning approaches

altogether can be tried. Different methods for generating the confidence targets and additional inputs for the network could be realized and tested

Furthermore our method to capture scenes with disturbances can be adapted for different forms of disturbances. Examples of real world disturbances are lens flares and glares, reflections on glass and wet surfaces, and snow. These datasets can then be used individually or combined to train a network that is able to predict confidences of a multitude of disturbances.

Bibliography

[1] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655. (page 12, 38)

[2] Ferstl, D., Reinbacher, C., Riegler, G., Ruether, M., and Bischof, H. (2015). Learning depth calibration of time-of-flight cameras. In *Proceedings of British Machine Vision Conference, (BMVC)*. (page 22)

[3] Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 1, 2, 45)

[4] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. (page 9, 10, 11, 12, 38)

[5] Hirschmuller, H. (2008). Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341. (page 4)

[6] Hu, X. and Mordohai, P. (2012). A quantitative evaluation of confidence measures for stereo vision. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2121–2133. (page 13)

[7] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*. (page 9)

[8] Kondermann, D., Nair, R., Honauer, K., Krispin, K., Andrulis, J., Brock, A., Gussefeld, B., Rahimimoghaddam, M., Hofmann, S., Brenner, C., et al. (2016). The hci benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 19–28. (page 1, 2, 45)

[9] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105. (page 12)

[10] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440. (page 9, 12, 38)

[11] Meister, S., Jähne, B., and Kondermann, D. (2012). Outdoor stereo camera system for the generation of real-world benchmark data sets. *Optical Engineering*, 51(2):021107. (page 1, 2, 21, 45)

[12] Menze, M. and Geiger, A. (2015). Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 1, 2, 45)

[13] Park, M.-G. and Yoon, K.-J. (2015). Leveraging stereo matching with learning-based confidence measures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 101–109. (page 2, 13, 14, 15, 45)

[14] Poggi, M. and Mattoccia, S. (2017). Learning to predict stereo reliability enforcing local consistency of confidence maps. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2. (page)

[15] Poggi, M., Tosi, F., and Mattoccia, S. (2017). Even more confident predictions with deep machine-learning. In *12th IEEE Embedded Vision Workshop (EVW2017) held in conjunction with IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2. (page 13)

[16] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer. (page 9, 12, 42, 43)

[17] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252. (page 10)

[18] Scharstein, D., Hirschmüller, H., Kitajima, Y., Krathwohl, G., Nešić, N., Wang, X., and Westling, P. (2014). High-resolution stereo datasets with subpixel-accurate ground truth. In *German Conference on Pattern Recognition*, pages 31–42. Springer. (page 1, 2)

[19] Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42. (page 4)

[20] Scharstein, D. and Szeliski, R. (2003). High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE. (page 1, 2)

[21] Seki, A. and Pollefeys, M. (2016). Patch based confidence prediction for dense disparity map. In *BMVC*. (page 13)

[22] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. (page 10, 12)

[23] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 7. (page 12)

[24] Zbontar, J. and LeCun, Y. (2016). Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17(1-32):2. (page 4, 5, 6, 21, 23, 37)

[25] Zhang, K., Lu, J., and Lafruit, G. (2009). Cross-based local stereo matching using orthogonal integral images. *IEEE transactions on circuits and systems for video technology*, 19(7):1073–1079. (page 7)