Monocular Depth Estimation of micro scale Images

Nick Stracke
s4771192

Bachelor's Thesis in Artificial Intelligence

Internal Supervisor:
Dr. Tim Kietzmann
Donders Institute
Radboud University

External Supervisor:
Dr. Faysal Boughorbel
ASM PT Competency Center
Netherlands

Radboud Universiteit

Artificial Intelligence
Radboud University
Netherlands
31/01/20

Abstract

Depth estimation based on a single image is a hot topic in current computer vision and deep learning research. Current work almost exclusively uses data depicting street or indoor scenes to train and benchmark their methods. However, there are plenty of other domains where depth information can be useful. I will elaborate how current state of the art methods can be applied to the domain of semiconductor devices where depth information can be useful to assess whether a device is damaged or faulty. I will showcase the entire process starting with a pipeline to obtain ground truth data and ending with bench marking several models. More precisely, I will train and evaluate a supervised, a self supervised and a semi supervised neural network. Ultimately, I demonstrate that it is possible to extract some depth information but that more research and data is needed to achieve dense and accurate disparity maps.

1 Introduction

Estimating depth from a single image or a stereo image pair is a classical field in computer vision. For many application it is crucial to have a 3D understanding of the scenes and to know how objects relate to each other and to the observer. Especially the recent advances in autonomous driving have led to a surge in interest of that topic.

Traditional methods focus on estimating the disparity (inverse depth) and are based on stereo correspondence algorithms which are well studied in the field (see [25]). Stereo correspondence algorithms work by comparing a left and a right image captured by two slightly displaced cameras. Based on the differences between these two images the depth can be triangulated. While these algorithms can produce high quality disparity maps, they face well known limitations.

First of all, they require manual tuning of hyper parameters. This might not be a problem if the camera positions remain constant and the general scene does not change much but will pose a significant bottleneck if this is not the case. Additionally, it has been notoriously difficult to estimate disparity in regions with repetitive patterns, little to no texture, or with different lighting conditions due to the very design of these algorithms. Lastly, most algorithms make the assumptions of an Lambertian world which often gets violated depending on the domain[23].

Due to the recent advances in machine learning and in particular of (fully) convolutional neural networks [18], deep learning models have been trained to predict depth with the goal to alleviate the most common limitations. Additionally, these models can also be trained to predict depth from a single, monocular image which eliminates the need of a dedicated camera rack to obtain stereo images. Initially, the field focused on supervised approaches [4] thus requiring the ground true depth as a target. Later, self supervised approaches gained attention due to the advancements of Garg et al. and Goddard et al. [6, 7].

While supervised approaches tend to perform better, they are limited by the fact that they require the ground truth depth map as a target for training. Obtaining these depth maps in the wild can be a challenge, e.g. due to sparse measurements of LIDAR sensors, but might be almost impossible depending on the domain. Thus, self supervised networks offer a promising alternative utilizing the geometric properties of a stereo image pair as a training signal.

In this work, I will assess how the aforementioned approaches can be applied to the domain of semiconductor devices. Semiconductor devices are small computer chips and have a size in the order of 1 millimeter with wires of around 0.5 - 1 micrometers in width. They get produced on a large scale and occasionally suffer from production faults such as cracks in the chip or loose wires. Missing these faults at production time can be costly and in combination with large scale production this necessitates automated solutions to spot production faults.

While there are many possible ways to do this, one assumption is that an accurate depth or disparity map can help to spot these faults. The challenge lies in the fact that these chips are very small and very different to the domains that depth estimation techniques are usually applied to. Furthermore, due to the micro scale, there is little ground truth data readily available. Thus, the first step will be to explore how ground truth data can be obtained since it is at the very least needed for evaluation purposes.

After that, three different neural network architectures will be trained to predict disparity maps. In the first stage, a convolutional neural network will be trained in a supervised manner based on partially manually generated disparity maps. Later, the focus will shift towards utilizing a self supervised approach and finally I will assess whether a combination of the two models can lead to better performance. My contributions are threefold:

- applying state of the art disparity estimation technology to a new domain that is significantly different to the original domain
- comparing three different architectures and exploring their strengths and weaknesses
- predicting relative disparity maps in a self supervised network as opposed to absolute disparity maps

More precisely, I will answer the question: How well can current state of the art disparity estimation technology be applied to the domain of semiconductor devices? After providing an overview of recent advances in depth prediction in section two, I will show how the ground truth data was generated in section three. In section four, the applied networks and network architectures will be discussed followed by implementation details in section five and a report of the experimental results in section six.

2 Related Work

Independent of how models are trained, the way they predict depth can be divided in two categories. Models of the first category make predictions based on a monocular image and are usually deep learning models. This is an ill-posed inverse problem because many 3D scenes can map to the same 2D image making it difficult to obtain depth maps that match the physical scale. Nevertheless, humans seem to be able to extract some depth information using a single eye and also neural networks have been shown to perform well. While these model infer depth based on a monocular image, it is important to realize that this does not mean that they were also trained using only a single image, e.g. self supervised models need a stereo pair when training but can make predictions based on a single image [7, 9].

The second category of models take multiple images as an input. This includes classical stereo correspondence algorithms but also deep learning models. These deep learning models are often an extension of the classical stereo correspondence paradigm as they use neural networks to perform some or all of the four steps as outlined by [25]. For instance, instead of computing the matching cost between two image section based on the image itself, a neural network can compute the matching costs based on the deep feature representation of the two images [14]. While models that take multiple input images as an input have more information available to them and thus should also perform better [26], they face the disadvantage of requiring a carefully calibrated camera rig to perform optimally which clearly contrasts the ease of taking a single image for monocular depth prediction. The following sections will refer to monocular depth estimation (the first category of models) with the exception of the last one.

Supervised The most common and straightforward approach are supervised neural networks. Supervised neural networks require ground truth data (a target) during training with the goal to make predictions as close to the ground truth as possible. One of the first attempts of supervised, monocular depth estimation was made by Saxena et al. [24] utilizing hand crafted features. Eigen et al. showed that neural networks can also be leveraged to learn depth features and subsequently estimate depth in combination with a scale-invariant error [4]. A lot of research has focused on improving the original architecture of [4] usually resulting in more complex models such as [5, 29]. On the contrary, [1] showed that a much simpler architecture in combination with transfer learning can achieve state of the art performance as well.

Self Supervised Self supervised neural networks have the advantage that they do not need ground truth data during training. They solely use the input data itself as a target which is a very well known method from autoencoders. This is especially interesting in the domain of depth estimation since ground truth depth data is not always available which can make a supervised approach infeasible. This was first explored by Garg et al. who proposed a self supervised neural network that exploits the geometric properties of a stereo image pair [6].

This approach requires a stereo pair during training (left - right) but can make predictions based on a single image. The idea is to predict a disparity map $\mathbf{d^l}$ for the left image $\mathbf{I^l}$ and use that to warp the right image $\mathbf{I^r}$ into the left image $\hat{\mathbf{I}}^\mathbf{l}$. A reconstruction loss can then be computed between the warped image $\hat{\mathbf{I}}^\mathbf{l}$ and the actual image $\mathbf{I^l}$ which serves as the learning signal. In order to be able to apply back propagation, Garg et al. linearized the warp image which is only a good approximation for small disparity values thereby ruling out images with large disparities.

Godard et al. expanded on this by introducing a left right consistency term to the loss and applying a fully differentiable bilinear sampler to warp the image [7]. This makes it possible to predict disparity maps with values of any magnitude and enforces consistency between disparities produced for the left and right image. Further additions were made to this approach, e.g. by not only predicting depth but also camera pose based on temporary adjacent images [8].

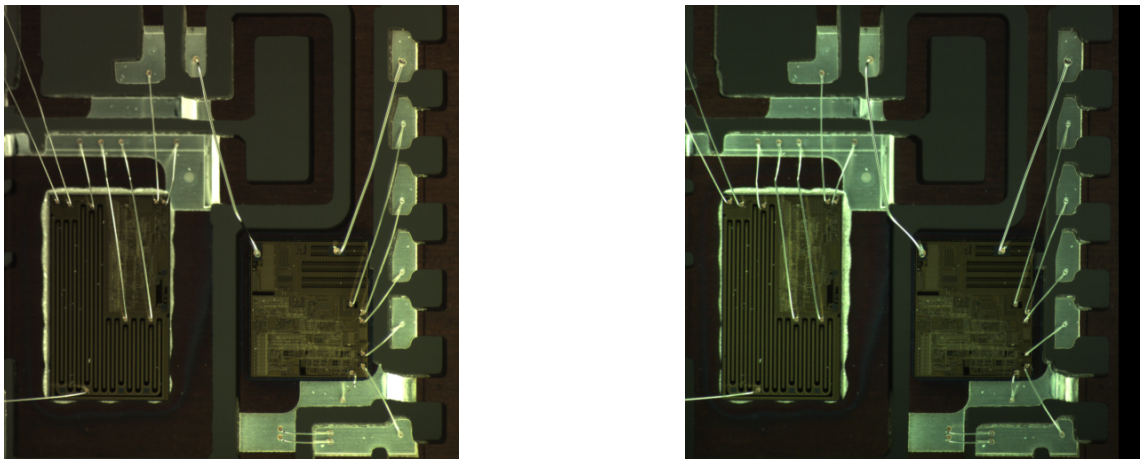
In [7] a single network predicts both the left *and* the right disparity map from a *single* image to compute the left right consistency loss. This can be problematic, e.g. when there is occlusion in the images [9]. Goldman et al. proposed a Siamese architecture that uses both images to predict the corresponding disparity maps which can then be used to compute the left right consistency loss [9].

Semi Supervised Combining the previous two approaches leads to a semi supervised network which can exploit the benefits of both methods. This is particularly interesting for datasets where only sparse ground truth data is available, e.g. KITTI, due to the low resolution of the LIDAR sensor[22]. Previously, supervised approaches used inpainting methods [17] to fill in the missing pixels [1]. A semi supervised approach can use the pixels for which the ground truth disparity is known as a supervisory signal and use the self supervised part of the network to fill in the gaps where no ground truth data is available [16, 2].

As a loss function, [16] used the berHu norm (supervised), a direct image alignment error (unsupervised) and a regularization term. [2] extended on this by adding more loss terms, e.g. left-right consistency as proposed by [7], as well as utilizing a Siamese network to share weights between the encoder and decoder.

Stereo Correspondence By design, stereo correspondence algorithm require (at least) a left and a right image and thus fall into the second category of models as defined at the start of this section. Before neural networks were available, most stereo correspondence algorithms were based on the four step outlined in [25], namely matching cost computation, cost aggregation, disparity computation/optimization and disparity refinement. Such algorithms can give a reasonable first estimate but they usually perform poorly when lighting or color conditions differ between the images or when there is occlusion. A recent development is to stay with these four steps but use (convolutional) neural networks for some or all of the previously mentioned steps [19, 14, 31].

This is in clear contrast to neural networks that directly go from a monocular image to a disparity map. It is important to realize this difference as pointed out in [26] because stereo matching algorithms have significantly more information available and therefore



(a) Left image (b) Right image

Figure 1: One example stereo image pair

also should perform better.

Zbontar et al. used a convolutional neural network to perform the first step, i.e. the matching cost computation [30]. Once the matching cost is obtained, steps from classical algorithms can be borrowed to arrive at the final disparity values. They showed that this produced lower error rate disparity maps than any other published method at the time.

Kendall et al. showed that it is also possible to include all four steps in a single neural network architecture called GC-Net [14]. They build a cost volume that contains the matching costs for all possible disparity values and use a differentiable soft-argmax function to obtain the final disparity map. This was extended upon by [31] who introduced a semi-global aggregation layer and a local guided aggregation layer to improve the second step by capturing local and whole-image cost dependencies respectively.

3 Data

The data set that will be used to train and evaluate models consists of three rectified stereo pairs of one 2024 by 2024 pixel and two 1024 by 1024 pixel RGB images. Each pair depicts a different semiconductor device. An external light source was used to illuminate the devices. The original data set included no ground truth data, i.e. disparity maps.

In general, obtaining accurate ground truth depth data is rather difficult. Often, depth maps are inaccurate too coarse and need to be postprocessed and enhanced, e.g. if they were collected by a LIDAR sensor. For semiconductor devices this problem is even more pronounced because the micro scale makes it almost impossible to collect depth information using common methods.

As such, instead of collecting depth data directly e.g. using an RGB-D or LIDAR sensor, the stereo RGB images will be used to generate a depth map which can then be used to train the network. In the following, the process of partially manually generating the ground truth data, in this case disparity, will be described.

3.1 Observations

In order to make it easier for me to motivate certain choices, I would like to highlight some observations that I made while inspecting the dataset.

1. Lighting conditions differ significantly between the left and the right image. As a result, there are color differences between images. See figure 1 where the right image

is much greener than the left image. Furthermore, next to the wires, there are other highly reflective regions in the image that look significantly different between both images.

2. There are many texture-less, homogeneous regions. Thus, any operation that compares areas of both images by e.g. computing the co-variance will be negatively influenced.

3. Similar to the previous point, there are large regions with repetitive patterns which makes it very difficult to correctly match two images.

4. It seems that most areas of the image are parallel to the camera and not slanted. Thus, we can expect the final disparity map to be mostly smooth (no image gradient) except the parts where two areas meet. There, a high gradient is expected as the two areas have a height difference.

5. Wires are always on top. Thus wires should always have a higher disparity than the background.

3.2 Disparity Estimation

Given two images taken by two identical and aligned cameras, disparity indicates how much pixels have shifted between these two images. In other words it indicates by how much these two images differ at any given pixel. Given the focal length and the position of both camera, this can be converted to depth according to the following equation:

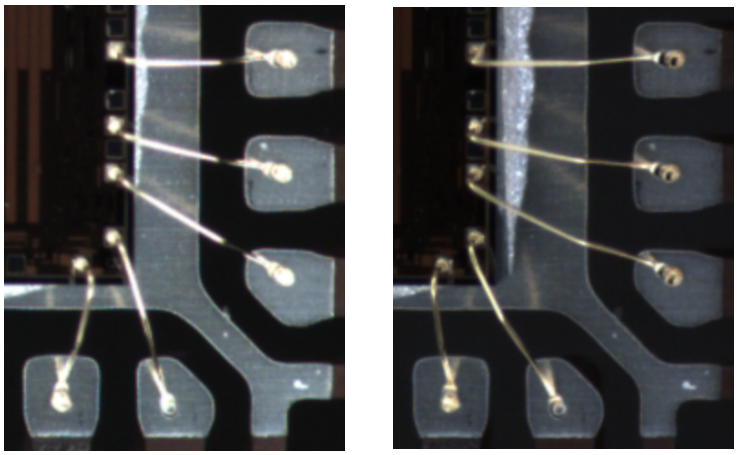
$$depth = \frac{Bf}{disparity}$$

where B refers to the distance between two cameras and f refers to the focal length.

As outlined in [25], most traditional algorithms follow a four step approach to match pixels and compute disparity. Arguably one of the most simple approaches is a block matching algorithm which uses a sliding window to match an image region of one image in the other image based on pixel intensity. Here, the difference in the x coordinate value between the two matching regions in the left and in the right image is the disparity.

This simple example already highlights one of the biggest limitation of these algorithms: it is difficult to obtain an accurate disparity in texture-less regions of the image. The sliding window will have the same correspondence value independent of how far it was shifted. Additionally, these algorithms make the assumption that the world is Lambertian, i.e. that all objects reflect light uniformly. If that is not the case, the same object might look different in the left and in the right image as it reflects the light differently, which again reduces the probability that it gets matched correctly.

In the domain of semiconductor devices these limitations are particularly pronounced. Due to the nature of these devices, there are a lot of areas on the chips with almost identical textures. Additionally, taking the images requires an external light source which gets reflected on the wires. These wires do not reflect light uniformly and thus look different on the two images making it hard for the algorithm to find a match (Figure 2). Lastly, the block matching algorithm searches along the horizontal axis of the image. If multiple wires happen to be close together in space, the algorithm might confuse one wires with another as they look almost the same and subsequently return an inaccurate disparity.



(a) Cut out of left image (b) Cut out of right image

Figure 2: Note the differences in wire appearance between the left and the right image. Especially the more horizontally oriented wires in the left image are very bright (almost white) on the left side and much darker on the right side. The wires in the right image barely change colors. Also note that the background is brighter in the left image.

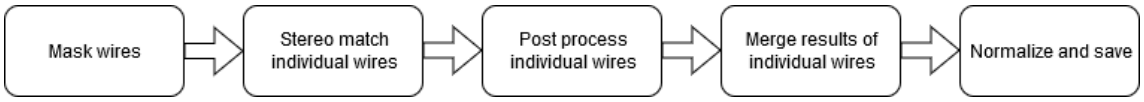


Figure 3: The complete preprocessing pipeline. First wires are masked from the background. Then wires are individually stereo matched based on their mask and the resulting disparity maps are post processed. Finally, the disparity maps per wire are merged and normalized so that all disparity maps have values ranging from zero to one.

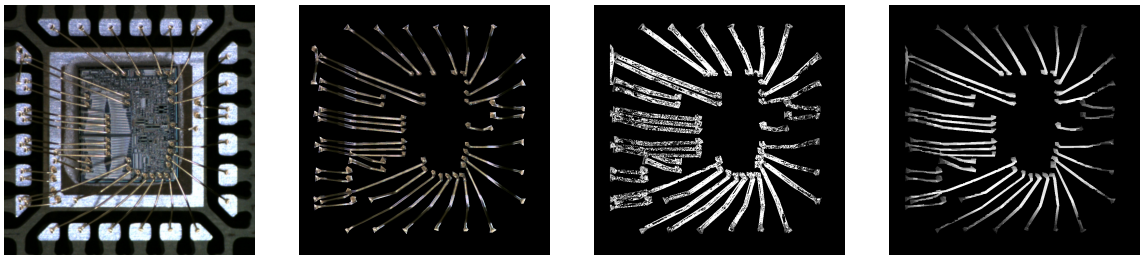
3.3 Method

Due to the above mentioned problems, traditional stereo matching algorithm will not provide an accurate disparity map which makes it unfeasible to train a supervised neural network. Therefore, the following relaxation is made to simplify the problem. Instead of estimating the disparity map of the whole image, only the disparity of the wires will be estimated thereby completely ignoring the background. This is a reasonable relaxation for this domain as most defects occur with the wires and thus depth information is mostly needed there. Wires will be matched individually to prevent the algorithm from mismatching two wires.

In total, three separate steps were taken to estimate the final disparity map. Firstly, the wires will be masked and separated from the background. Secondly, the disparity will be estimated for each wire individually. Lastly, the resulting disparity map will be post-processed.

Preprocessing Wires were masked from the background using LabelMe [27]. The images were converted to gray scale and every pixel that did not belong to a wire was set to black. Subsequently, wires were separated into separate images so that no two wires appear in the same image. The final result are two (left - right) 3D volumes of gray scale images where each image in the volume contains an individual wire.

Block Matching Next, each individual wire image pair was fed into a traditional block matching algorithm as implemented by OpenCV [3]. The relevant hyper parameters in this case are the minimum and the maximum disparity as well as the block size. The first



(a) Starting image (b) Masked out wires (c) Stereo matching (d) Post processing

Figure 4: Visualization of a left image after every step of the pipeline excluding the normalization and saving step. This shows all wires, i.e. after merging, while the actual pipeline runs on individual wires and only merges wires at the very end. In image (c) note how the stereo matching algorithm performs worse as the wires become more horizontally aligned.

two parameter indicate how much of the image the algorithm searches to find the best matching regions in the left and the right image. The block size determines the size of the sliding window that is used to find corresponding regions in the two images. Due to large differences between semiconductor devices, these parameters had to be hand-tuned on a case by case basis. The result is again a volume of disparity maps for each individual wire.

Note that these disparity maps are by default aligned to the *left* image. As will be shown later, the disparity map for the *right* image is also required. This map can be generated by mirroring each image in the volume and swapping the volumes before feeding them into the same algorithm. The resulting volume then again has to be mirrored [9]. This effectively transforms the left image into the right image and vice versa. Thus, the resulting disparity map is aligned to the right image. In total, the outputs are two disparity volumes; one is aligned to left and the other is aligned to the right.

Postprocessing Since wires tend to be textureless, the performance of the algorithm quickly degrades as the wires become more horizontally aligned because the matching cost over the wire is more or less constant. This results in very sparse disparity estimates for horizontal wires. Closing these gaps is essential for fine disparity maps that can be used to train a network. This was achieved by first dilating and then eroding the disparity maps using a rectangular kernel adjusted to each stereo pair. These operations were again done for each wire individually to prevent close wires from fusing together in the dilation step. The resulting disparity map volumes were combined individually to yield the left and right disparity maps by taking the maximum value along the depth axis. This ensures that if wires overlap, the disparity information of the highest wire will be kept. Additionally, the wire masks were applied again. Even though the dilation and erosion operation largely cancel each other out at the edges, they still marginally increase the width of the wire. Applying the masks again ensures that the disparity wires stay thin and match their counterparts in the intensity image.

The very last step is to normalize the disparity maps due to different disparity ranges in the images. In case of negative disparities, the right image and disparity map were shifted to the left to ensure that all disparity values are positive. Secondly, the disparity maps were stretched so that every disparity map has values ranging from 0 to 255 and saved as gray scale images. The original, absolute disparity range was also stored to allow the reconstruction of absolute disparity values out of the stretched values.

The final outcome are two disparity maps for the left and right image respectively with values between 0 and 255.

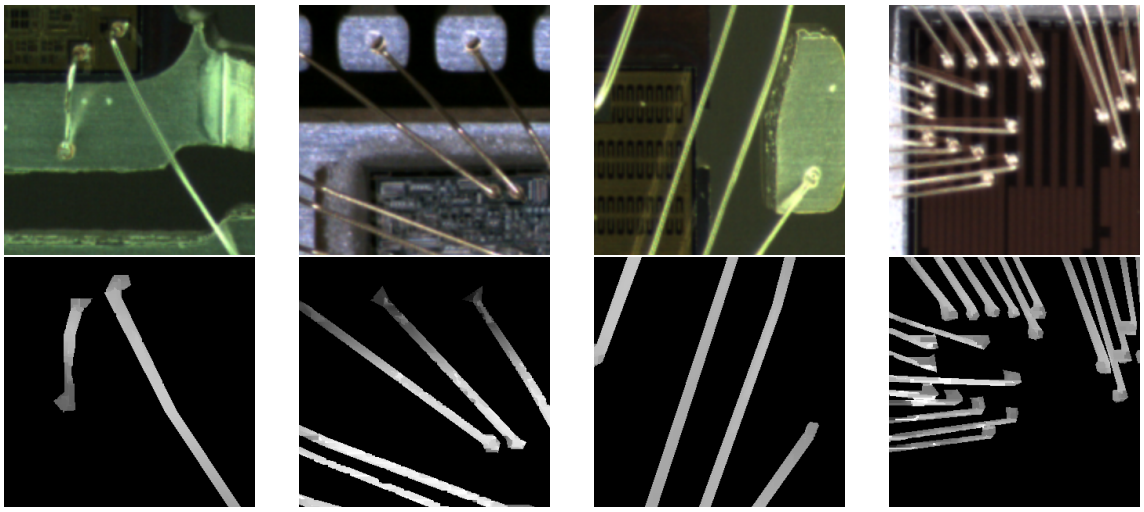


Figure 5: Samples of the training data. The bottom row shows the generated ground truth data.

3.4 Sampling

The left and right images and disparity maps were sampled into images of 320 by 320 pixels. Due to the low number of images (3), the sampling windows was moved with a stride of 1/8th oo the sample size thereby oversampling the training data. Every fifth image was held out as validation data. Note that due to over sampling, the training and validation samples do largely overlap. Only for testing, a single 320 by 320 pixel sample per image was held out completely without overlap. Ultimately, this results in ca. 1500 pairs of images and disparity maps as training data. There are ca. 400 pairs of validation data and three pairs of test data.

4 Methods

As discussed earlier, having sparse ground truth depth data is not uncommon. As an example, the KITTI dataset [22] also contains mostly sparse depth data because the lidar resolution is lower than the camera resolution. As a result, many pixels in the image do not have a counterpart in the depth map. This makes it difficult to train a supervised neural networks and often requires inpainting [17] or other methods to fill in the gaps.

In this scenario the situation is different. The disparity maps are also sparse but contain very dense depth information for a small proportion of the image (the wires) and no information at all for the background. Therefore, simply filling in the gaps between the missing regions of the image is not an option.

The result of this is a three-fold approach. First, a supervised neural network is trained using the sparse disparity data with the goal to predict accurate disparity information for the wires and ignore anything else in the image. This is described in the first section. Secondly, a self supervised network is trained similar to [7, 9]. This can be seen as an extension to the first, supervised version as it uses the same encoder and decoder to predict the disparity map. However, instead of computing a loss over the disparity map directly, the disparity map is used to warp one image into the other. The original image and the warped image is then used to compute a reconstruction loss which serves as the training signal. This is described in the second section. Thirdly, the previous two versions are combined yielding a semi supervised approach [2, 16].

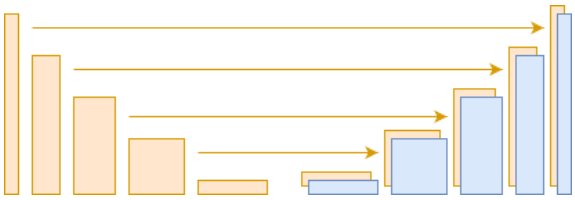


Figure 6: Overview of the supervised network architecture. In total there are four skip connections. The output disparities are half the size of the input images.

4.1 Supervised

The supervised network is largely inspired by the architecture of DenseDepth [1] due to its simplicity while still producing state of the art results on common datasets. This seemed adequate given that the domain is difficult but also limited and because the amount of training data is rather low. The original DenseDepth network takes a single RGB image as an input and uses DenseNet-169 pretrained on ImageNet [13] to encode it. The decoder is composed of four upsampling blocks which consist of a $2\times$ bilinear upsampling layer, followed by skip connections and finally two convolution layers. In the original architecture, the decoder does not contain Batch Normalization [12] and uses a leaky ReLu [20] as an activation function. The final layer is convolutional with a linear activation function. The output is a gray scale image half the size of the original input image.

In order to adept the network to the task at hand, I made the following adjustments. The encoder was changed from DenseNet-169 to DenseNet-121 [11] because the smaller encoder did not decrease performance while speeding up training and arguably reducing overfitting. The final activation function was changed to the logistic function because the target disparities only contain values between 0 and 1 unlike DenseDepth where target values lie between 10 and 1000. This however led to a vanishing gradient problem where the network would predict only black images. I found that adding a single Batch Normalization layer after the skip connections in each upsampling block solved that problem and performed the best.

4.2 Loss Function of Supervised Network

The loss function is defined as a measure that indicates how much the predicted disparity map $\hat{y}$ deviates from the true disparity map y. It is common for networks that predict depth or disparity to have loss functions consisting of multiple terms. The original DenseDepth [1] used a loss function consisting of three terms: the point-wise L1 loss, a loss based on the Structural Similarity (SSIM) [28] and the L1 loss over the gradients of the disparity map. It is common that the L1 and the SSIM loss get weighted against each other because the L1 loss is more accurate but also more susceptible to changes in color and gamma values whereas the SSIM is more forgiving in that regard but also less accurate. The goal of the gradient loss is to penalize high frequency details which tend to occur more often around objects boundaries. The ground truth data was normalized to only have values between zero and one before computing the loss.

Recall that the ground truth data is sparse and that the disparity is only defined for the wires. This is a significant relaxation of the problems but also bears the questions what components of the original loss are still required. In the end, keeping the same components as proposed by [1] resulted in the best performance. For ease of notation, the L1 and the SSIM loss are combined as the image loss. Thus, the total loss is:

$$L_{super} = L_{img} + L_{edge} \tag{1}$$

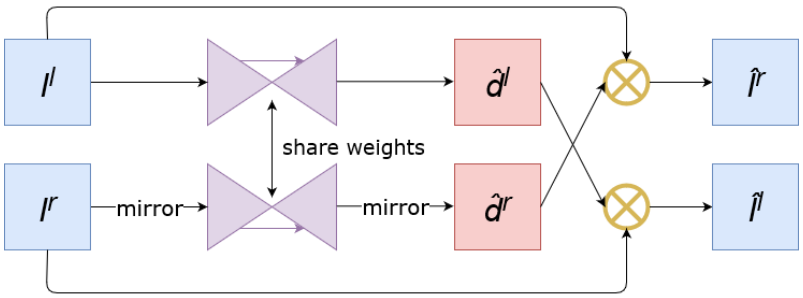


Figure 7: Overview of the self supervised network architecture. The yellow component represents the bilinear sampler. Note the mirroring of the right image.

Image Loss The image loss consists of a combination between the SSIM and the L1 norm with a higher weight on the SSIM as this has been shown to lead to better performance [28, 1, 9].

$$L_{img}(y, \hat{y}) = \alpha_{SSIM} L_{SSIM}(y, \hat{y}) + (1 - \alpha_{SSIM}) L_{L1}(y, \hat{y}) \tag{2}$$

Since the original range of the SSIM lies between -1 and 1 where smaller values indicate a lower SSIM, it is adjusted to represent an appropriate loss term:

$$L_{SSIM}(y, \hat{y}) = \frac{1 - SSIM(y, \hat{y})}{2} \tag{3}$$

The L1 loss is simply the absolute difference between two pixels:

$$L_{l1}(y, \hat{y}) = \frac{1}{N} \sum_{i,j} \mid y_{i,j} - \hat{y}_{i,j} \mid \tag{4}$$

Edge Loss The edge loss is the difference between the x and y gradient of the true and the target disparity map [1]. It can be seen as a regularization term and promotes that disparities are smooth and do not contain sudden jumps in height.

$$L_{edge}(y, \hat{y}) = \frac{1}{N} \sum_{i,j} \mid \partial_x y_{i,j} - \partial_x \hat{y}_{i,j} \mid + \mid \partial_y y_{i,j} - \partial_y \hat{y}_{i,j} \mid \tag{5}$$

4.3 Self Supervised

The self supervised network uses the same encoder - decoder part as the supervised network. However, instead of taking a single image as an input it operates as a Siamese network and takes two input images, namely the left and the right image. Due to the weight sharing property of the Siamese architecture, this does not increase the total number of parameters but allows the network to predict two disparity maps at a time. These will be needed later when computing the loss. The yellow component indicates the bilinear sampler as proposed and implemented by [7].

As pointed out by [9] predicting the disparity map for a left and for a right image is *not* the same function. In other words, there are two mappings namely $f(\mathbf{I_l}) = \hat{\mathbf{d}}_\mathbf{l}$ and $f'(\mathbf{I_r}) = \hat{\mathbf{d}}_\mathbf{r}$ but the functions $f()$ and $f'()$ are not the same. The two images $\mathbf{I_l}$ and $\mathbf{I_r}$ are sampled from two different distribution; at least because the cameras from which the images were taken have different relative positions [9].

Therefore, using the naive approach of feeding in both images as is will result in decreased performance. It is of course possible to train two separate networks but that

would double the amount of parameters while also ignoring all geometric symmetries that do lie between the two images.

As a solution to this problem, [9] proposed to mirror the right image. Then, the two images $\mathbf{I_l}$ and $m(\mathbf{I_r})$ can be considered to be from the same distribution. This works because mirroring one image effectively reverses the direction of the disparity. As a result, both input images have the same disparity direction which in turn allows the function $f()$ to be applied to both images [9] thereby making Siamese networks feasible. As a result, the right image is mirrored before the encoder and mirrored again right after the decoder 7.

Note that both [7] and [9] use a variation of DispNet [21] which produces outputs on four different scales. This is not the case here as the decoder is the same as for the supervised network and thus only produces an output on a single scale.

4.4 Loss Function Self Supervised Network

The loss functions consists of four components in total. Three of them are similar to others [7, 9]. The forth, additional component is a regularization term. Contrary to [7], the loss is only computed over a single scale, i.e. the final output.

$$L_{self} = \alpha_{img}(L^l_{img} + L^r_{img}) + \alpha_{lr}(L^l_{lr} + L^r_{lr}) + \alpha_{tv}(L^l_{tv} + L^r_{tv}) + \alpha_{reg}(L^l_{reg} + L^r_{reg}) \quad (6)$$

Image Loss The image loss follows the same definition as the image loss for the supervised network but is computed over the reconstructed image and the original image. It provides a measure indicating the quality of the image reconstruction. However, as stated by [7] achieving a high image reconstruction quality does not necessarily entail that the underlying disparity map is also accurate which is why the other loss components are needed.

Left-right consistency loss Similar to others [7, 9] the left right consistency check exploits geometric properties and results in higher quality disparity maps. It is defined by:

$$L^l_{lr}(d^l, d^r) = \frac{1}{N}\sum_{i,j} \mid d^l_{i,j} - d^r_{i,j+d^l_{i,j}} \mid \quad (7)$$

Total variational loss The total variational loss acts as a regularization term by enforcing smoothness of the predicted disparity map. Similar to [7], the loss is being made edge aware by weighting the disparity gradients with the gradients of the input images. A function to weight the gradients was proposed by [10] and is defined as $\partial_x d e^{-|\partial_x I|}$ for the x direction where d is the disparity map and I the image. This basically allows large gradients in the disparity map when there also is a large gradient in the original image and is motivated by the fact that depth discontinuities often occur at strong image gradients [10] .

$$L^l_{tv}(d) = \frac{1}{N}\sum_{i,j} \mid \partial_x d^l_{i,j} \mid e^{-|\partial_x I^l_{i,j}|} + \mid \partial_y d_{i,j} \mid e^{-|\partial_y I^l_{i,j}|} \quad (8)$$

Regularization Loss Due to the fact that large parts of the image are textureless, the network tends to shift the entire image instead of only warping the regions that need to be shifted. This works because it will shift the wires to the correct location while not resulting in a higher loss for the background regions because they are textureless. Thus, I

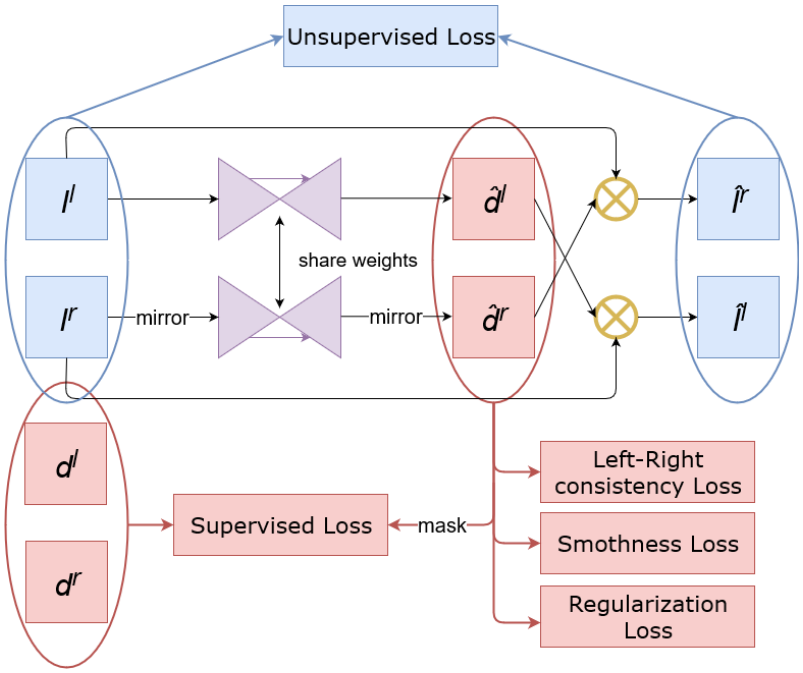


Figure 8: Overview of the different losses for the semi supervised network. The only difference to the self supervised network is the supervised loss. Note the masking operation between the disparity predictions and the computation of the loss.

added a regularization term over the entire image to limit the amount of high frequency component in images. In this case, this is just the L1 loss over the entire image:

$$L^l_{reg}(I) = \frac{1}{N} \sum_{i,j} \mid I^l_{i,j} \mid \tag{9}$$

4.5 Semi Supervised

The semi supervised network is a direct combination of the previous two networks. The architecture is the same as for the self supervised network but the supervised loss is added (Figure 8). Since the supervised loss has to be computed for both disparity maps, the total loss is as follows:

$$L_{semi} = \alpha_{super}(L^l_{super} + L^r_{super}) + \alpha_{self} L_{self} \tag{10}$$

The ground truth data only contains disparity values for the wires thus the supervised loss is exclusively calculated over those pixels that have a disparity value defined. This is in line with previous semi supervised network implementations such as [16, 2]. The architecture of [2] is very similar to this one and also uses a Siamese network but to my knowledge they do not mirror one of the two input images. Lastly, here the edge loss is not included in the supervised loss because it leads to artifacts due to the masking and because the self supervised loss already includes a smoothness term.

4.6 Augmentation

Employing a data augmentation pipeline is a standard procedure to increase performance and reduce over-fitting [15]. To ensure that the results of all three networks are comparable, the same augmentation pipeline was used for all networks. Since the self supervised network makes assumptions about the geometry of the two input images, not all augmentation policies are valid.

Parameter	Value
α_{SSIM}	0.9
α_{img}	1
α_{lr}	1
α_{tv}	0.1
α_{super}	1
α_{self}	1

Table 1: Hyper parameter settings grouped by model. The value of α_{reg}, which is missing here, is shown in section 6.

As already mentioned earlier, the self supervised Siamese network effectively learns to predict the disparity map for a *left* image. This is why the *right* image has to be mirrored before feeding it into the networks. Thus, it makes no sense to include mirroring as a preprocessing step since it would reverse the direction of the disparity. In a similar vein, rotating the image is also not a valid augmentation step as it changes the disparity direction. On the other hand, flipping the images vertically is possible because the images have no features that would indicate where the top or bottom is as opposed to outside pictures where the sky usually provides a strong indication. Thus, the chance of a vertical flip is 0.5. Lastly with a chance of 0.5 the three color channels of the image are randomly permutated [1].

4.7 Inference and Postprocessing

Contrary to the supervised network which only takes one image as an input, the self and semi supervised network train on two input images, which is not monocular due to the Siamese structure. During inference however, only a sub graph of the network is used, i.e. the "left" part of the Siamese network. This results in an architecture that is identical to the supervised network and is thus monocular.

Furthermore, at test time it is common [7, 9] to not only predict the disparity of the given image but also to predict the disparity of the mirrored version of that same image and blend the two predictions together. This helps to reduce occlusion artifacts and generally improves the disparity maps [7, 9]. Results obtained using this post processing method are indicated with **pp** (Table 3).

5 Implementation Details

All three networks are implemented in TensorFlow and the self and semi supervised network use the exact same encoder - decoder as the supervised network. All three networks were run on a single NVIDIA TITAN V for 20 epochs. The weights were initialized randomly and ADAM was chosen as an optimizer with a learning rate of 0.0001 and parameter values $\beta_1 = 0.9$, $\beta_2 = 0.999$. There was no additional learning rate scheduling. All three networks had the same number of trainable parameters due to the weight sharing of the Siamese network, namely $10,712,097$.

5.1 Image Warping and Sampling

The self supervised network learns by warping one image into the other, i.e. by reconstruction of one image out of the other. However, such a reconstruction cannot always be perfect even if the underlying disparity map is perfect. Imagine a stereo pair where the left image contains an object at the very left. This object might not be visible in the

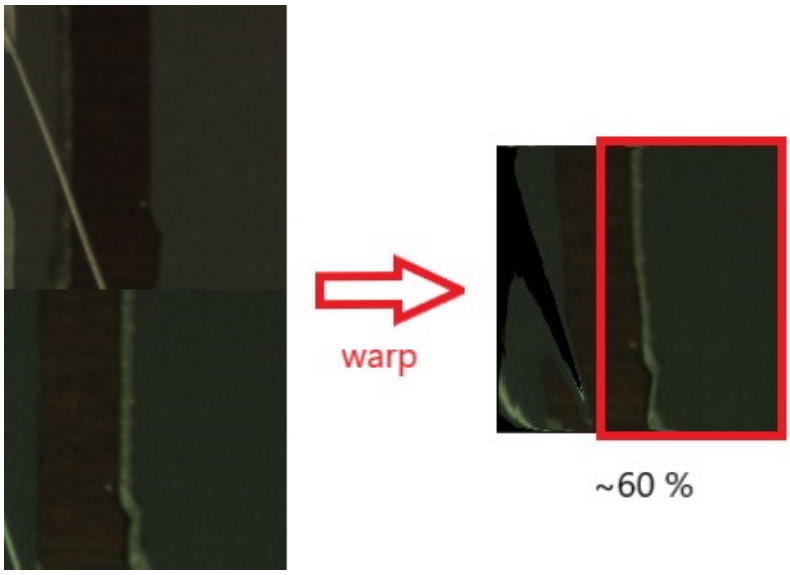


Figure 9: The network tries to warp the bottom (right) image into the top (left) image but fails because the wire can not be reconstructed. This can be seen in the black region where the wire approximately should have been. The proposed solution is to only compute the loss over the image section that can be faithfully reconstructed as indicated by the red rectangle.

right image due to the different camera positions. As a result, the right image cannot be perfectly warped into the left image because the pixels for the object in the left image cannot be reconstructed.

This is usually not a problem if the ratio between maximum disparity and width of the image is low. A ration of 0.1 would mean that at max 10% of the image cannot be faithfully reconstructed by the warping process. In reality that number is even lower because it is unlikely that there are a lot of close (thus high disparity) objects at the edge of the image.

However, as that ratio grows reconstructing one image out of the other becomes less and less feasible. This is a problem as sampling many small images out of one large image reduces the width of the image but not the disparity range. This is one reason why it is desirable to use images as a whole but the limited amount of training data makes sampling necessary. Given the sample image size of 320 by 320 pixels and a maximum disparity range of 144 pixels, this can result in images where the disparity to width ration is close to 40%. This problem can be dealt with in two ways:

Wrap Mode The bilinear sampler of [7] has two wrap modes. The wrap mode specifies how border pixels are handled. In border wrap mode, if the predicted disparity refers to a pixel outside of the image, the sampler will return a black pixel thus resulting in large reconstruction error for that pixel. In edge wrap mode, the sampler returns the pixel value of the edge pixel. This usually results in a much lower reconstruction error especially when that region has a homogeneous structure and can simply be stretched.

The downside of the latter mode is that the network will simply learn to warp (shift) the entire image instead of only the areas that actually need to be warped since the penalty of warping at an edge region is so low. Secondly, the semiconductor devices largely consist of homogeneous, textureless regions making warping the entire image a viable option.

Partial Loss Instead of computing the loss over the entire image thereby including regions that cannot be reconstructed, the reconstruction loss only gets calculated over the parts of the image that can be reliably reconstructed. For instance, with a known ratio of 40% between maximum disparity and image width, the reconstruction loss would only get calculated over the right 60% of the reconstructed left image and the left 60% of the right image.

This also means that the network does not get a training signal for 40% of the image. However, since subsequent image samples are oversampled and have an overlap of 87.5%, the network will still see all parts of the training data. Unfortunately, this approach faces the same issues as the previous one, namely that the network will simply learn to warp the entire image.

In practice, this is an ill-posed problem that is largely caused by the fact that the images are sampled into smaller patches. Of course, this problem will always remain as it is never possible to completely reconstruct the image but as the ratio between maximum disparity and width of the image decreases, so does the significance of this problem. In this case, the latter approach has been chosen in combination with the aforementioned regularization loss that penalizes disparity maps with a lot of high values.

5.2 Relative Disparities

As explained in section 2, the generated disparity maps contain relative disparity values and not absolute ones. This is necessary because two almost identical looking images can have different disparity ranges (and thus disparity maps) due to the way the cameras were positioned when taking the stereo images. As a result, it would be impossible for a neural network to associate the right disparity range to a given image. This is in contrast to other data sets, e.g. KITTI, where the disparity range does not change because the cameras are mounted on the car and do not change position. Predicting relative disparities values instead of absolutes ones allows the network to generalize over the dataset because assumptions over the global scale can be made, e.g. that wires should always have a rather large values associated to them.

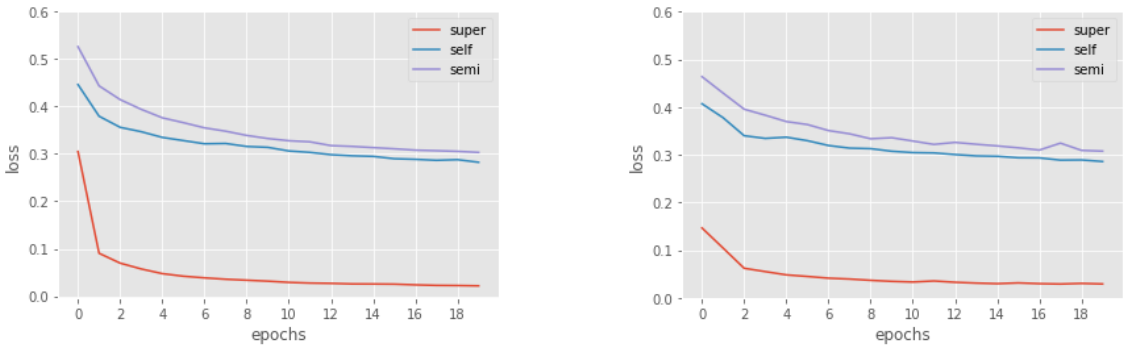
However, the self supervised network needs an absolute disparity value to warp the image, i.e. it needs to know by how many pixels to displace the pixels in the to be warped image. This is achieved by passing the disparity range of each input image as an additional input to the network. Thus, the self supervised network *can* predict relative disparity values $(0 - 1)$ and then use the disparity range to transform the relative scale in an absolute one to warp the image. This also allows the supervised and self supervised network to be combined. If one network were to predict relative disparities and the other one absolute disparities such a combination would not be possible.

The bilinear sampler of [7] was adjusted to reflect that change.

6 Experiments

I present test results for the test split of the training dataset. The test images were completely held out and were selected from a random area of the image. Recall that the predictions of the self and semi supervised neural network have disparities defined for the entire image but the ground truth data has disparities defined only for the wires. Thus, for these models the metrics are only calculated over the wires. The supervised model is evaluated on both the wires only and the full images because it should learn to predict no disparity for the background. To make a fair comparison, only the results for the wires are used to compare the supervised model to the other two.

Additionally, multiple loss variations of the the self and semi supervised model were



(a) Training loss

(b) Validation loss

Figure 10: Overview of the losses of the three default models

Model	Loss	α_{reg}
Self	partial	0.05
Self*	full	0
Self**	partial	0
Semi	partial	0.1
Semi*	full	0

Table 2: Overview of loss variations per model

tested to evaluate whether they actually lead to an increase in performance. The default models were trained with the partial loss as described in section 5.1 and with a regularization term. The variations were trained with a full loss and no regularization loss *or* a partial loss and regularization loss.

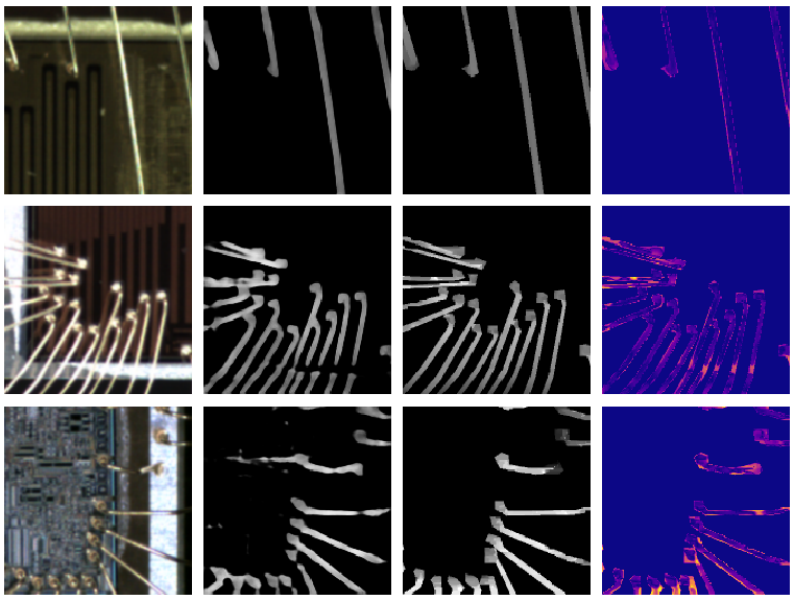
6.1 Evaluation Metrics

The models are evaluated on the standard metrics previously used by other researchers [4, 7]. Before computing the metrics, the relative disparity maps which the network outputs were scaled using the known disparity range of each image. This ensures that absolute disparity values are evaluated and compared and not relative ones.

- absolute relative error: $\frac{1}{N}\sum_{p}^{N}\frac{|y_p-\hat{y}_p|}{y_p}$
- squared relative error: $\frac{1}{N}\sum_{p}^{N}\frac{(y_p-\hat{y}_p)^2}{y_p}$
- Root-mean squared error (RMSE): $\sqrt{\frac{1}{N}\sum_{p}^{N}(y_p-\hat{y}_p)^2}$
- $\log_{10}$ error: $\frac{1}{N}\sum_{p}^{N}\mid \log_{10}(y_p)-\log_{10}(\hat{y}_p)\mid$
- threshold accuracy δ_i: % of y_p such that $\max(\frac{y_p}{\hat{y}_p},\frac{\hat{y}_p}{y_p})=\delta<thr$ for $thr=1.25,1.25^2,1.25^3$

Method	Abs Rel ↓	Sq Rel ↓	RMSE ↓	Abs log ↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
Super (wires)	**0.457**	10.425	17.167	0.281	**0.516**	0.682	0.754
Super (full)	0.739	22.796	8.196	0.085	0.885	0.912	0.925
Self	0.601	16.430	27.846	0.411	0.093	0.204	0.368
Self*	0.709	22.458	33.020	0.586	0.043	0.099	0.184
Self**	0.460	9.786	16.986	0.187	0.450	0.643	0.788
Semi	0.459	**9.058**	**14.060**	**0.146**	0.481	**0.731**	**0.874**
Semi*	0.507	10.792	19.229	0.234	0.236	0.590	0.766
Super_{pp} (wires)	**0.438**	9.475	16.168	0.247	**0.515**	0.696	0.775
Self_{pp}	0.599	16.300	27.794	0.405	0.095	0.193	0.356
Semi_{pp}	0.449	**8.341**	**13.615**	**0.146**	0.478	**0.729**	**0.883**

Table 3: The results of the different networks. Self* refers to the self supervised network where the full loss was computed, i.e. over the entire image, and without the extra regularization loss. Self** refers to the self supervised network where only the partial loss was computed but without imposing a regularization term. Semi* indicates the semi supervised network with a full loss and no regularization term (Table 2).



(a) Test data

(b) Validation Data

Figure 11: Results of the supervised network (Super) on three test images without **pp**: input RGB images, my predicted disparity maps, my generated ground truth disparity maps, difference between predictions and ground truth for the wires.

6.2 Results

Supervised Based on the qualitative results, the supervised network largely manages to capture the wires while ignoring the background (Figure 11). Unsurprisingly, the network tends to performs worse when wires have a similar color to the background as can be seen

in the bottom two rows. Furthermore, the largest errors occur around the edges of the wires (right most column). This could be due to inaccurate labeling of wires, meaning that the wires as seen in the true disparity map do not perfectly align with the wires in the image. If this was the case, then such errors are actually desired showing that the network does not overfit. In addition to that, Figure 11b shows an example where the ground truth data (third column) is not complete. Nevertheless, the network predicts a disparity for the whole wire. Lastly, note that due to smoothing effects of the network, close wires tend to fuse together as shown in the middle row (Figure 11a).

With respect to the metrics, the supervised network outperforms the semi supervised models on two out of the seven metrics. However, it is worth pointing out that the metrics could certainly be better. Especially compared to the results this model achieved on the KITTI dataset in [1]. The low number of test samples or again the incorrect masking could offer an explanation for this. Evaluated on the full disparities, the model's results are much better indicating that it successfully learns to ignore the background and predict no disparity for it.

Regarding the training process, the loss quickly drops which is an indicator that the network learns to completely ignore the background as every disparity outside of the wires will result in a large loss. A similar pattern can be seen in the validation loss. In later epochs the loss barely decreases and the network presumably only fine tunes the wire disparities.

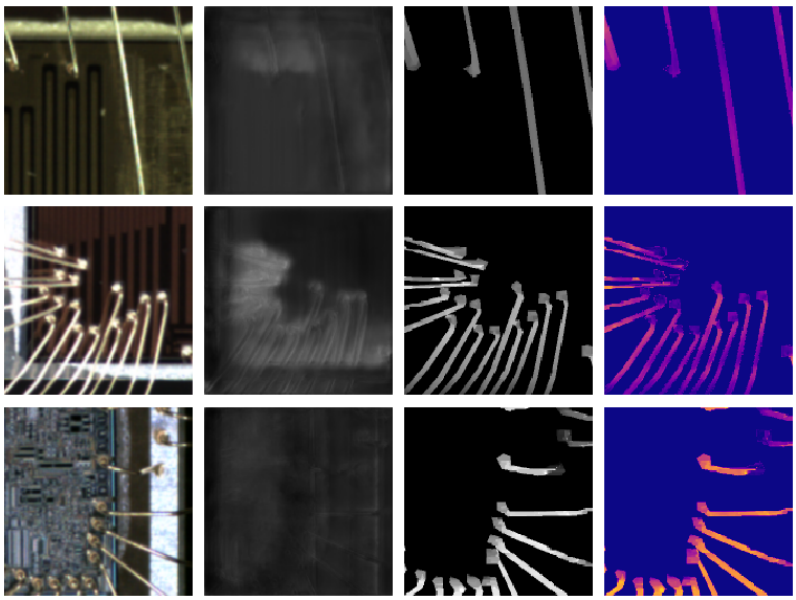


Figure 12: Results of the self supervised network (Self) on three test images without **pp**: input RGB images, my predicted disparity maps, my generated ground truth disparity maps, difference between predictions and ground truth for the wires.

Self Supervised The self supervised network partially captures some background structures but fails to detect most wires (Figure 12). The overall performance is rather poor which is also supported by the metrics. Only the self supervised model with a full loss (Self*) performs even worse (Table 3). Interestingly, the metrics seem to indicate that the variation of the self supervised network that did not include the regularization term (Self**) performed the best. The reason for this can be seen in Figure 13. The network learns to predict smooth disparity maps with high average values thereby also predicting high disparity values for the wires. This leads to a good performance (Table 3) because the error metrics are only computed over the wires. However, the underlying disparity

Figure 13: Left and right image with disparity map of the left image as predicted by the Self** model. When comparing the left and the right image, it is clear that the disparity should be very low with respect to the scale. Yet, the unregularized model predicts a high disparity.

maps are not very accurate and do not seem to match the underlying image (Figure 13). Thus, the seemingly good performance is caused by the incomplete evaluation.

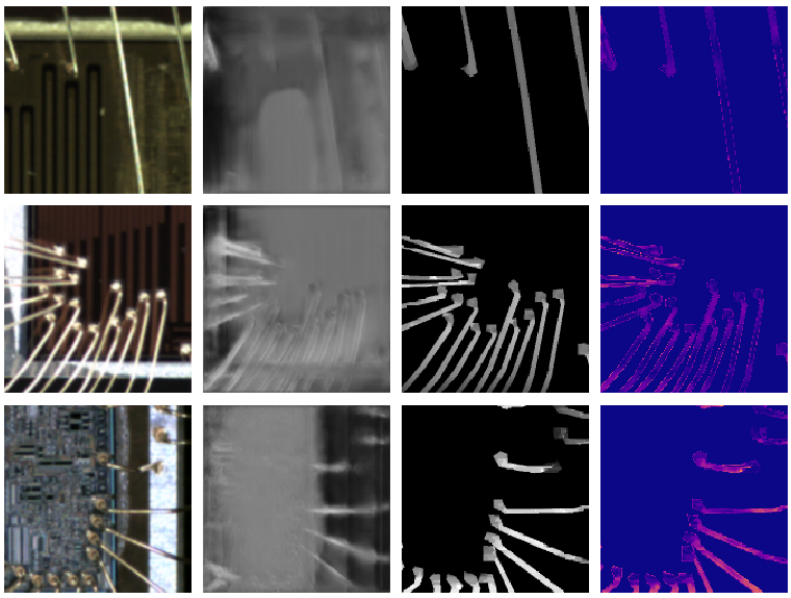


Figure 14: Results of the semi supervised network (Semi) on three test images without **pp**: input RGB images, my predicted disparity maps, my generated ground truth disparity maps, difference between predictions and ground truth for the wires.

Semi Supervised The semi supervised network appears to combine the properties of the previous two models quite well. While the wires are not as clear as in the supervised model, it is clear that the supervised loss did put attention on the wires, especially when compared to the self supervised model. Furthermore, the predicted background disparities seem to align to the images. In the first row of Figure 14, it can be seen that repetitive patterns are still a problem and that the predicted disparity does not really align with the image.

The semi supervised network outperforms the supervised network in five out of the seven metrics. Looking at the test images this seems largely due to the fact that the semi supervised network predicts the pixels around the edges of the wires better. However, this does not mean the the model actually detects wires better. It seems more that the model predicts a high disparity for regions with a lot of wires. Thus, there is a positive disparity defined for the edges as opposed to the supervised network where no disparity is defined.

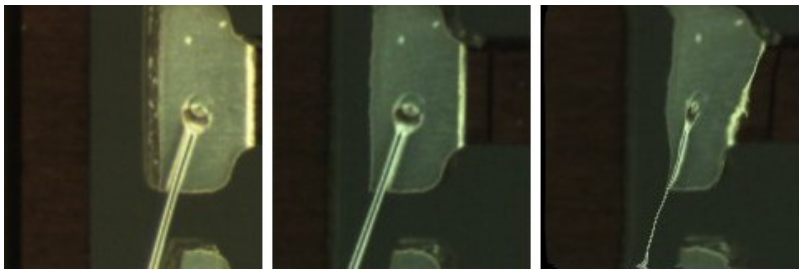
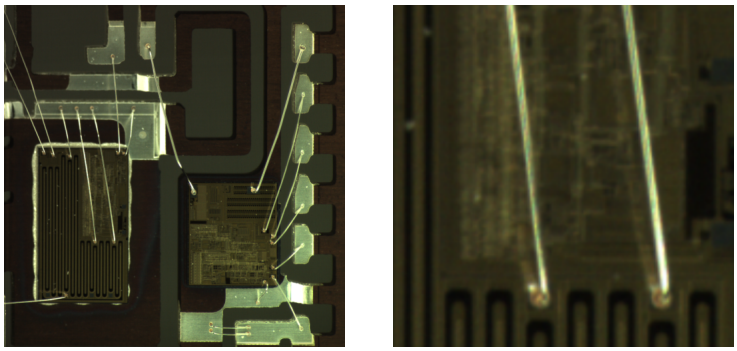


Figure 15: Left, right and and warped right to left image. Note how thin the warped left wire is.

6.3 Discussion

Model Comparison Taking the previous qualitative and quantitative results of the three models into account, the supervised network performs the best if the focus lies on the wires. It clearly detects the wires, ignores everything and predicts disparity values close the ground truth. The biggest weakness of the self supervised network is that it fails to reliably detect wires. This is largely due to the fact that these wires tend to be very thin which makes it difficult to warp. Sometimes, this even results in wires (partially) disappearing (Figure 15). As this is a result of the warping process itself and thus crucial to learning, it poses a clear limitation of the self supervised approach in this domain. The semi supervised model on the other hand seems like a promising solution because it largely solves the wire detection problem. However, due to the incomplete ground truth data, it is difficult to say how accurate the predicted background disparity is. On a qualitative basis, it seems to capture most structures well but faces the same problem as other stereo correspondence algorithms, e.g. repetitive patterns.



(a) Full left image (b) 320 by 320 pixel patch

Figure 16

Disadvantages of Sampling The original images have a resolution of 2024 by 2024 pixels and are thus too large to directly feed them into a neural network. The standard procedure to decrease the resolution of the images is usually to down scale them by a given factor. This has the advantage that the global scale of the image gets preserved. Unfortunately, this approach was not feasible here due to the very limited amount of training data. As a result, small image patches were sampled out of large images. However, this approach comes with its own set of problems.

It makes it impossible for the network to learn how different parts of the image relate to each other. Thus the network has to fully rely on prior knowledge to solve these situations. This problem becomes very clear in figure 16. The patch belongs to a part of the image

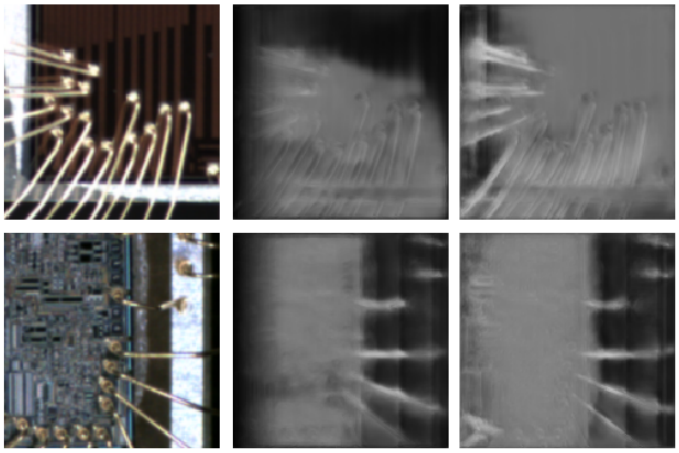


Figure 17: Overview of how the partial loss influences the prediction quality of the semi supervised neural network. In the second column the network was trained with a complete loss. In the third column the network was trained with a partial loss.

that is elevated with respect to other part of the chip but this is impossible to deduce from the patch only.

Partial Loss and Regularization As discussed in previous sections, only computing a partial loss helps the network to cope with the fact that the training data contains relatively large disparity values compared to the width of the image. This results in both quantitative improvements (Table 3) and well as qualitative improvements (Figure 17). Note that especially the regions close to the left and right border of the image are much more clearly defined.

6.4 Limitations and Future Work

Overall, the dataset used for training and testing is very small. This requires practices such as sampling many small patches out of one large and using overlapping patches which arguably leads to less accurate disparity maps as discussed in previous sections. Additionally, the network is relatively large compared to the amount of data which begs the question whether the network actually generalized to the domain or if it was just overfitting to the three semiconductor devices. Furthermore, while the test dataset has no overlap with the training or validation set, it is still from the same semiconductor device. To fully assess whether the network generalized to the domain if would be necessary to test the network on semiconductor devices of types that it was not trained on.

Furthermore, there was no complete ground truth data available. This made it impossible to fully evaluate the self and semi supervised network. Having access to complete disparity maps would not only allow a complete evaluation but it would also enable the supervised network to learn to predict disparities for the whole image. Then, the supervised and self supervised networks could be compared more thoroughly.

Lastly, it could be of interest to train the self supervised network on datasets with known benchmarks such as KITTI. Since the encoder - decoder is different and arguably simpler than e.g. [7, 9], the performance differences could be an interesting aspect for future research.

7 Conclusion

In this work, I have shown one way of approaching the problem of depth estimation on a micro scale from generating ground truth data to training and evaluating a set of models. I presented a pipeline to extract disparity data from stereo images and to convert them into training data by masking out the wires. Furthermore, I proposed a series of three deep learning models that are built on top of each other with each being more complex and utilizing more features of the input data than the previous one.

My goal was to access how well current state of the art disparity estimation technology can be applied to the domain of semiconductor devices. I have shown that it is possible but that there are clear limitations. Furthermore, I have arrived at a set of models that can at least partially estimate depth from a monocular image. I pointed out many problems that are caused by the small data set and by the domain itself and proposed solutions to at least somewhat mitigate these problems. I believe that with more training data this approach is worth revisiting and could result in greatly enhanced performance.

References

[1] Ibraheem Alhashim and Peter Wonka. "High Quality Monocular Depth Estimation via Transfer Learning". In: *arXiv e-prints* abs/1812.11941, arXiv:1812.11941 (2018). eprint: 1812.11941. URL: https://arxiv.org/abs/1812.11941.

[2] Ali Jahani Amiri, Shing Yan Loo, and Hong Zhang. *Semi-Supervised Monocular Depth Estimation with Left-Right Consistency Using Deep Neural Network*. 2019. arXiv: 1905.07542 [cs.CV].

[3] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[4] David Eigen, Christian Puhrsch, and Rob Fergus. "Depth map prediction from a single image using a multi-scale deep network". In: *Advances in Neural Information Processing Systems*. 2014.

[5] Huan Fu et al. "Deep Ordinal Regression Network for Monocular Depth Estimation". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2018. ISBN: 9781538664209. DOI: 10.1109/CVPR.2018.00214. arXiv: 1806.02446.

[6] Ravi Garg et al. "Unsupervised CNN for single view depth estimation: Geometry to the rescue". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2016. ISBN: 9783319464831. DOI: 10.1007/978-3-319-46484-8_45.

[7] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. "Unsupervised monocular depth estimation with left-right consistency". In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. 2017. ISBN: 9781538604571. DOI: 10.1109/CVPR.2017.699.

[8] Clément Godard et al. *Digging Into Self-Supervised Monocular Depth Estimation*. 2018. arXiv: 1806.01260 [cs.CV].

[9] Matan Goldman, Tal Hassner, and Shai Avidan. "Learn Stereo, Infer Mono: Siamese Networks for Self-Supervised, Monocular, Depth Estimation". In: *Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019.

[10] Philipp Heise et al. "PM-Huber: PatchMatch with huber regularization for stereo matching". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013. ISBN: 9781479928392. DOI: 10.1109/ICCV.2013.293.

[11] Gao Huang et al. "Densely connected convolutional networks". In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. 2017. ISBN: 9781538604571. DOI: 10.1109/CVPR.2017.243. arXiv: 1608.06993.

[12] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *32nd International Conference on Machine Learning, ICML 2015*. 2015. ISBN: 9781510810587. arXiv: 1502.03167.

[13] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: 2009. DOI: 10.1109/cvprw.2009.5206848.

[14] Alex Kendall et al. "End-to-End Learning of Geometry and Context for Deep Stereo Regression". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017. ISBN: 9781538610329. DOI: 10.1109/ICCV.2017.17. arXiv: 1703.04309.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks". In: *Communications of the ACM* (2017). ISSN: 15577317. DOI: 10.1145/3065386.

[16] Yevhen Kuznietsov, Jörg Stückler, and Bastian Leibe. "Semi-supervised deep learning for monocular depth map prediction". In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. 2017. ISBN: 9781538604571. DOI: 10.1109/CVPR.2017.238. arXiv: 1702.02706.

[17] Anat Levin, Dani Lischinski, and Yair Weiss. "Colorization using optimization". In: *ACM Transactions on Graphics*. 2004. DOI: 10.1145/1015706.1015780.

[18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2015. ISBN: 9781467369640. DOI: 10.1109/CVPR.2015.7298965.

[19] Wenjie Luo, Alexander G. Schwing, and Raquel Urtasun. "Efficient deep learning for stereo matching". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2016. ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.614.

[20] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. "Rectifier nonlinearities improve neural network acoustic models". In: *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*. 2013.

[21] Nikolaus Mayer et al. "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2016. ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.438. arXiv: 1512.02134.

[22] Moritz Menze and Andreas Geiger. "Object Scene Flow for Autonomous Vehicles". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[23] Michael Oren and Shree K. Nayar. "Generalization of the Lambertian model and implications for machine vision". In: *International Journal of Computer Vision* (1995). ISSN: 09205691. DOI: 10.1007/BF01679684.

[24] Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng. "Learning depth from single monocular images". In: *Advances in Neural Information Processing Systems*. 2005. ISBN: 9780262232531.

[25] D. Scharstein, R. Szeliski, and R. Zabih. "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms". In: *Proceedings - IEEE Workshop on Stereo and Multi-Baseline Vision, SMBV 2001*. 2001. ISBN: 0769513271. DOI: 10.1109/SMBV.2001.988771.

[26] Nikolai Smolyanskiy, Alexey Kamenev, and Stan Birchfield. "On the importance of stereo for accurate depth estimation: An efficient semi-supervised deep neural network approach". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. 2018. ISBN: 9781538661000. DOI: 10.1109/CVPRW.2018.00147. arXiv: 1803.09719.

[27] Kentaro Wada. *labelme: Image Polygonal Annotation with Python*. https://github.com/wkentaro/labelme. 2016.

[28] Zhou Wang et al. "Image quality assessment: From error visibility to structural similarity". In: *IEEE Transactions on Image Processing* (2004). ISSN: 10577149. DOI: 10.1109/TIP.2003.819861.

[29] Dan Xu et al. "Monocular Depth Estimation Using Multi-Scale Continuous CRFs as Sequential Deep Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019). ISSN: 19393539. DOI: 10.1109/TPAMI.2018.2839602. arXiv: 1803.00891.

[30] Jure Žbontar and Yann Lecun. "Stereo matching by training a convolutional neural network to compare image patches". In: *Journal of Machine Learning Research* (2016). ISSN: 15337928. arXiv: 1510.05970.

[31] Feihu Zhang et al. *GA-Net: Guided Aggregation Net for End-to-end Stereo Matching.* 2019. arXiv: 1904.06587 [cs.CV].