ESTIMATION OF DEPTH FROM DEFOCUS BLUR IN VIRTUAL ENVIRONMENTS COMPARING GRAPH CUTS AND CONVOLUTIONAL NEURAL NETWORK

A Thesis

Submitted to the Faculty

of

Purdue University

by

Prodipto Chowdhury

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

December 2018

Purdue University

Indianapolis, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

To my parents, Rumna Sengupta and Pranab Kumar Chowdhury.

ACKNOWLEDGMENTS

I would like to express my earnest gratitude to my advisor Dr. Lauren Christopher for giving me the opportunity to work in this project and mentoring throughout to make it successful. I would like to thank Dr. Brian King and Dr. Zina Ben Miled for serving on the thesis committee.

I am extremely grateful to David Emerson and Ashley Dale for their collaboration and I greatly appreciate their contribution to this project. I am also thankful to Prof. Albert William and Wendy Krogg for their valuable suggestions regarding Autodesk Maya.

I would like to acknowledge the support of NSWC Crane for completing this project. Finally, I would like to thank my family and friends for always being there for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABBREVIATIONS

DfD	Depth from Defocus
CNN	Convolutional Neural Network

ABSTRACT

Chowdhury, Prodipto. M.S.E.C.E., Purdue University, December 2018. Estimation of Depth from Defocus Blur in Virtual Environments Comparing Graph Cuts and Convolutional Neural Network. Major Professor: Lauren Christopher.

Depth estimation is one of the most important problems in computer vision. It has attracted a lot of attention because it has applications in many areas, such as robotics, VR and AR, self-driving cars etc. Using the defocus blur of a camera lens is one of the methods of depth estimation. In this thesis, we have researched this technique in virtual environments. Virtual datasets have been created for this purpose.

In this research, we have applied graph cuts and convolutional neural network (DfD-net) to estimate depth from defocus blur using a natural (Middlebury) and a virtual (Maya) dataset. Graph Cuts showed similar performance for both natural and virtual datasets in terms of NMAE and NRMSE. However, with regard to SSIM, the performance of graph cuts is 4% better for Middlebury compared to Maya.

We have trained the DfD-net using the natural and the virtual dataset and then combining both datasets. The network trained by the virtual dataset performed best for both datasets.

The performance of graph-cuts and DfD-net have been compared. Graph-Cuts performance is 7% better than DfD-Net in terms of SSIM for Middlebury images. For Maya images, DfD-Net outperforms Graph-Cuts by 2%. With regard to NRMSE, Graph-Cuts and DfD-net shows similar performance for Maya images. For Middlebury images, Graph-cuts is 1.8% better. The algorithms show no difference in performance in terms of NMAE. The time DfD-net takes to generate depth maps compared to graph cuts is 500 times less for Maya images and 200 times less for Middlebury images.

1. INTRODUCTION

Depth estimation refers to the problem of estimating the distance of each point in a scene from a 2D image. It is one of the most important problems in computer vision. It has attracted a lot of attention because accurate estimation of depth would be highly useful to areas like robotics [1], [2], scene understanding [3], virtual and augmented reality [4], [5], [6], human-computer interaction [7] and self-driving cars [1], [2]. Other computer vision tasks that could be benefited by depth estimation are scene recognition [8], Simultaneous Localization and Mapping (SLAM) [9], semantic segmentation [10], human activity recognition [11] and human pose estimation [12].

1.1 Literature Review

Researchers have taken different approaches for depth estimation. For example, Delage et al. [13] and Hedau at al. [14] have used geometric cues to estimate the depth of different sections (ceiling, wall, floor, etc.) of an indoor environment. However, they have made strong assumptions about the geometry of indoor environments and hence not suitable to estimate the depth of the outdoor scenes.

Scharstein et al. [15] and Sinz et al. [16] have used stereo for depth estimation, where the 3D information of a scene is reconstructed by using a pair of images of the scene. Furukawa et al. [17] have used a number of images of a scene captured from different viewpoints. Ranftl et .al's [18] approach was to move a camera through a scene and use two consecutive frames captured by it to make the depth map. The limitation of these methods is they need more than one images for depth inference.

Some researchers have attempted to perform depth estimation from single images. But they have made strong assumptions about the scene, and therefore, work only in specific cases. For example, shape from shading depends on the gradual variation of shade in an image, and does not work well for the images where the colors and textures are not uniform [19]. Criminisi et al. have done the 3D affine measurement of a scene by assuming that the vanishing point of a plane can be determined from an image [20]. Liu et al. formulated the depth estimation problem as a discrete-continuous optimization problem, where the continuous variables encoded the depth of each super-pixel in the image; and the discrete ones encoded the relationship between neighboring super-pixels [21].

1.1.1 Depth from Defocus

Focus and defocus have been used as important cues for depth estimation. Depth from focus is based on the fact that a camera lens can focus only a finite set of objects in a scene for a particular focus distance; the rest of the objects are out of focus. Therefore, in this method, a flurry of images are taken; each corresponding to a different focus distance, and consequently, each giving depth of a different set of objects. The limitation of this technique is obvious, a good number of images is required to construct the depth map of a scene. The number of images needed to be captured increase with the complexity of the scene.

On the other hand, in depth from defocus, depth perception is achieved by estimating the level of defocus of each object in a scene. Therefore, it usually requires only a pair of images; unlike depth from focus.

Researchers have taken different approaches to find depth from defocus. Favaro et al. have modeled defocus as a diffusion process, and represented defocus blur as heat diffusion by using the heat equation [22]. In another work, they have demonstrated a different method for depth from defocus where they assumed pixels with similar color belong to the same surface [23]. Subbarao et al. have developed a spatial domain convolution/deconvolution transform method, which is not highly accurate and the resolution of the yielded depth map is not high as well [24], [25]. Watanabe et al. have used an illumination pattern that is projected via the same optical path to acquire images [26]. Therefore, this method requires special optical settings.

Some researchers have attempted to solve this problem by designing special hardware. Ng et al. developed a light-field camera that could record the full light field and estimate the depth map [27]. Levin et al. modified the aperture of a conventional camera instead of making a new one [28]. They made the aperture more sensitive to defocus blur. However, since these methods require specially designed hardware, they are not accessible to everyone.

1.1.2 Application of Convolutional Neural Networks for Depth Estimation

Since the work of Krizhevsky et al. [29], deep convolutional neural networks (CNN) have been widely used to perform computer vision tasks. Some researchers have successfully applied CNNs to the depth estimation problem. Eigen et al. proposed a multi-scale architecture to predict depth [10]. Liu et al. [30], Mahmood et al. [31], Li et al [32], and Wang et al [33] proposed a joint deep CNN and conditional random field (CRF) framework for depth estimation. Grigorev et al. proposed a deep hybrid neural network by integrating convolutional layers with recurrent layers [34]. Roy et al. combined random forests and CNNs to present an architecture that predicted depth in continuous domain via regression [35].

1.1.3 Virtual Worlds as a Source of Data

Accurately annotated dataset is essential to train and test computer vision algorithms, in particular CNNs. However, properly labeled data at the right amount is often not available. For example, we needed a dataset with in focus image, out of focus image, and ground truth for this project. No dataset exists to the best of our knowledge that contains all these three elements.

Virtual worlds could be a solution to this problem. Recent progress in computer graphics and GPUs allows us to create virtual worlds that can provide photorealistic data. Researchers started testing the feasibility of this idea. Butler et al. [36] derived a synthetic dataset MPI-Sintel from the animated short film 'Sintel' [37] for optical flow estimation. Stark et al. [38] and Broggi et al. [39] developed 3D CAD models to use as synthetic data. Marin et al. trained pedestrian classifiers using virtual data and then tested those classifiers with real data [40]. The results of their experiments suggested classifiers trained in virtual scenes can successfully detect pedestrians in real images. Vazqez et al. showed in [41] there can be a dataset shift between virtual and real data, and proposed an unsupervised domain adaptation technique to overcome it in [42]. Hattori et. al also developed a similar approach, but their work was scene-specific and required prior knowledge about the geometry of the scene [43]. Taylor et al. developed a virtual simulation test bed for video surveillance [44]. Sun et al. [45] and Peng et al. rendered virtual data from 3D models and used that to train object detection algorithm [46]. In [47] Gaidon et al. showed deep neural networks trained on virtual data perform similarly in real and virtual environments. In addition to that, since real and virtual environments are almost the same except for the weather and imaging conditions, virtual worlds allow us to assess the impact of those conditions on the performance of the algorithm.

1.2 Motivation and Contribution

Chao Liu showed in his PhD thesis that graph cuts is the state of the art technique for estimating depth from defocus. He compared graph cuts with other algorithms and graph cuts produced best results. [48]. On the other hand, David Emerson applied a convolutional neural net (DfD net) for depth estimation and compared it with graph cuts in his PhD prelim. He showed the performance of DfD net is similar to graph cuts and it takes much less time to generate the depth maps [49]. However, both of them performed the expermients for natural (Middlebury) images.

In this research, we have studied the performance of graph cuts and DfD-net for virtual (Maya) images and compared it with the performance obtained for natural (Middlebury) images.

1.3 Organization

- Chapter 2 describes the datasets, the two types of blur, and their comparison.
- In Chapter 3, a theoretical overview of the algorithms (graph cuts and CNNs) and how they have been applied to this project have been discussed.
- Chapter 4 describes the performed experiments and discusses the results.

2. THE DATASETS

We have used two datasets in this project. One is the Middlebury dataset, created by Brad Hiebert-Treuer, Sarri Al Nashashibi, and Daniel Scharstein at Middlebury College [50], [51]. The dataset contains 486 in focus images with ground truth from 27 different scenes. However, the dataset does not contain blurred images. We created the blurred images synthetically and by Photoshop. It will be discussed later.

The second dataset has been created using Autodesk Maya. We have created 5 virtual scenes. Five hundred in focus images and their corresponding ground truth images have been constructed. Like Middlebury images, the blurred images have been formed synthetically and by Photoshop.

Maya provides all kinds of 2D and 3D shapes, which are the basic building blocks of a virtual environment. There are a lot of functionalities for modifying the shapes, which allows the designer to create any kind of shape. Then suitable colors and/or textures are applied to the shapes.

After a virtual environment is created, the next task is to capture images. For this purpose, we need to make a camera in Maya. The camera parameters can be adjusted according to our requirements. Then the camera position and angle are varied and we have a different image for each different camera position/angle.

An in focus image and its corresponding depth map is required for each captured image. This is achieved using the layers in Maya. There is a master layer, which renders the in focus image; and a zDepth layer that renders the depth map. Maya offers several choices for rendering. We have used Maya software as its render quality and rendering time is good enough for our work.

The third element of the dataset is the blurred image. We have used two blurring methods to blur the images.

- Synthetic blur
- Photoshop Lens blur

2.1 Synthetic Blur

Synthetically blurred images are created by convolving the in focus image with a blur kernel, which is assumed to be Gaussian. 256 values are assigned to the blur parameter (σ), from σ_{min} to σ_{max}. Each value represents a different blur level. Thus, 256 blurred versions of the in focus image are created. Each blur corresponds to a different level of brightness (brighter means closer to camera). The value of a particular pixel in the depth map works as the index of the blurred stack. From the blurred image, the same pixel is picked and used as the pixel for the defocus image. The process is demonstrated in the following figure.

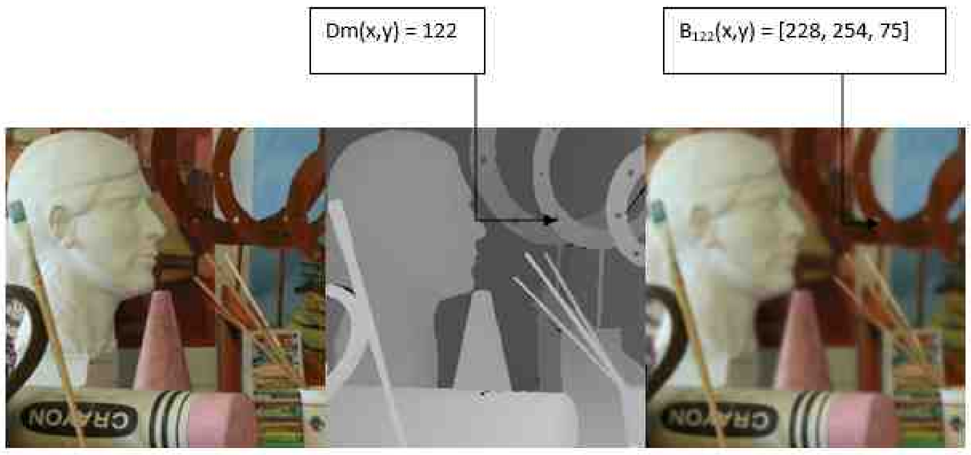


Fig. 2.1.: An image blurred synthetically

2.2 Photoshop Lens Blur

We have used another type of blur; namely 'lens blur' by Adobe Photoshop. This blur is supposed to mimic the defocus blur created by a real camera lens. Among few other blurs we tried (blur by Maya, for example), Photoshop lens blur seems to be the closest.

Like the synthetic blur, lens blur uses the depth map as the source to estimate the amount of blur of a pixel. The blur intensity can be controlled by changing the 'blur radius'. We have chosen the value of the blur radius as 10.

In Fig. 2.2, an image and its synthetic and photoshop lens blur are shown.

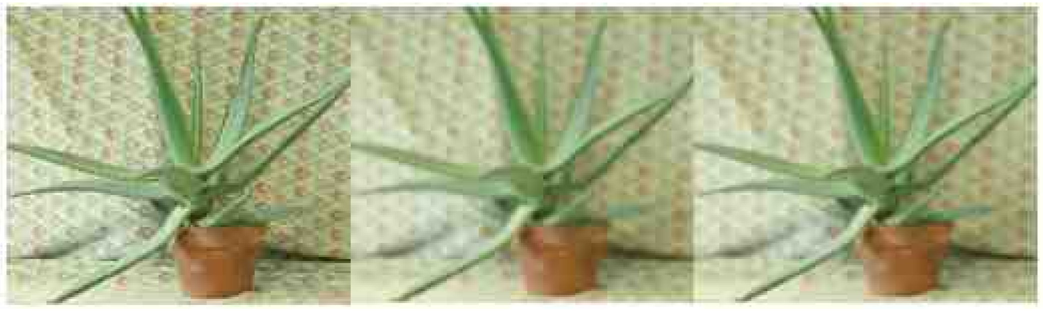


Fig. 2.2.: In-focus image, synthetic blur, and photoshop lens blur

2.3 Error Metrics

We have used three error metrics to compare the output of the algorithms with the ground truths.

- Normalized Mean Absolute Error (NMAE)): It is the average of the absolute differences between prediction and actual observation where all individual differences have equal weight. Since all errors have equal weight, outliers with high variance cannot affect overall error much [52].

- Normalized Root Mean Squared Error (NRMSE): It is the square root of the average of squared differences between prediction and actual observation. Since the errors are squared in this case, a few high variance outliers can lead to a higher error [52].

  For NMAE and NRMSE, the lower the value of the error, the smaller the difference is between the observation and the prediction.

- Structural Similarity Index Measurement (SSIM): It measures similarity between two images. It can be viewed as the quality measure of one of the images being compared, provided the other image is considered as of perfect quality [53]. SSIM value one means the images are exactly same and zero means there is no similarity at all.

In Fig. 2.3, the error maps between two blurs in Fig. 2.2 is presented.

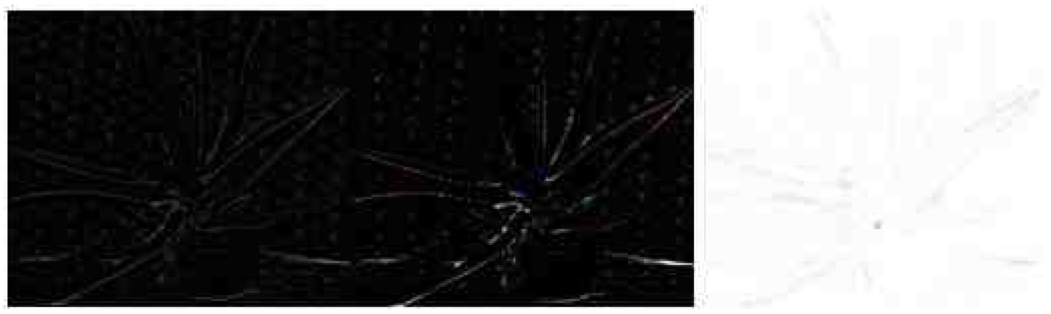


Fig. 2.3.: NMAE, NRMSE, and SSIM

The edges are different according to all three metrics. Experiments on other images also shown the higher the number of the sharp edges in an image, the more different two blurs are for that image. This makes the results for two blurs significantly different.

3. ALGORITHMS

We have used two algorithms to solve the depth from defocus problem-

- Graph cuts.
- Convolutional neural networks.

3.1 Graph Cuts

Graph cuts is a widely used algorithm in computer vision for solving optimization problems. Problems like image segmentation, stereo imaging, etc. have been modeled as finding the minimum cut of weighted graphs [54].

For applying graph cut to these problems, an image is considered as an undirected graph; where the pixels represent the vertices and the linkages between the pixels represent edges. The graph is $G =< V, E >$, where V and E are the sets of vertices and edges, respectively. Besides pixels, the graph contains another special type of node called terminals. There are two types of terminals: source, S, and the sink, T. An $s - t$ cut, $C = S, T$ is the partition of the vertices in V into two disjoint sets S and T such that $s \in S$ and $t \in T$.

All the vertices connect with terminals. Therefore, the edges can be classified into two types- [54]:

- N-link: the edges connect the pixels with their neighbors.
- T-link: the edges connect the pixels with terminals.

Every edge in this S-T graph has a non-negative weight or cost. Cutting an N-link edge will have a penalty cost for neighboring pixels. And similarly, cutting a T-link edge will lead a cost for assigning the corresponding label to the pixel. So after one cut, the cost of all edges has the minimum value, it is called minimum-cut.

Boykov and Kolmogorov [54] developed a multi-label graph cuts algorithm for multi-class problems. They used the energy function shown below to obtain the minimum cut of S-T graph.

$$E(L) = \sum_{s \epsilon S} D_s(L_s) + \sum_{(s,r) \epsilon N} V_{s,r}(L_s, L_r) \tag{3.1}$$

Where L is a set of labels for each pixel in the image, $D_s()$ is a data penalty function of pixel s. $V_{r,s}()$ indicates the similarity of the pixel with its neighbors. And N is the set of all pairs of neighboring pixels. By minimizing the energy function, the original image can be segmented into different parts.

Depth from defocus algorithm can be described as assigning a label to each pixel in such a way that an energy function (Eq. 3.1) is minimized. The energy function is a map from the set of all possible labels and is minimized when the segmentation best conforms to a cut model. By using graph-cut algorithm to minimize the energy function (Eq. 3.1), 256 blur classes are used as terminals and the pixels in initial depth map are used as vertices for the S-T graph.

The best NMAE, NRMSE, and SSIM values obtained by the graph cuts method are 0.024529, 0.062146, and 0.954497, respectively. The in focus and defocus image, and ground truth and obtained depth map are shown below-

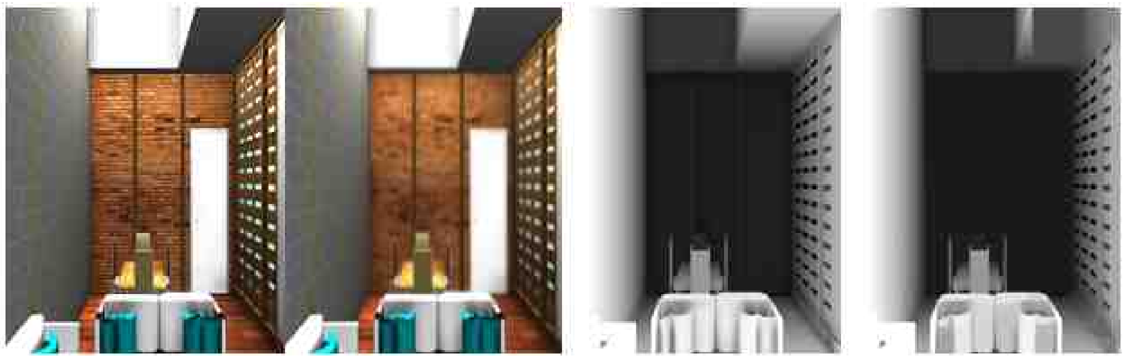


Fig. 3.1.: In-focus image, defocus image, ground truth, and constructed depth map

3.2 Convolutional Neural Network

Convolutional neural networks have emerged as an effective tool to solve computer vision problems. U-net, which is a special variant of the fully convolutional network, has been chosen as primary candidate architecture for this project. It has been developed by Ronneberger et al. to perform a binary segmentation of images of cellular structures [55].

The U-net consists of two main parts. One is the contracting path, which is similar to a classification neural network. It consists of two 3×3 convolutional layers, each followed by Relu and a 2×2 max pooling layer for downsampling. The other part is the expansive path. Each step in this path consists of a 2×2 up-convolutional layer. The output of this path is concatenated with the feature map obtained from the corresponding layer of the contracting path. Then there are two 3×3 convolutional layers, each followed by Relu. The expansive path is somewhat symmetric to the contracting path, resulting in the U-shaped architecture [55].

Several modifications have been done to the original U-net. Three convolutional blocks have been replaced with a residual block, as shown in the Fig. 3.3.

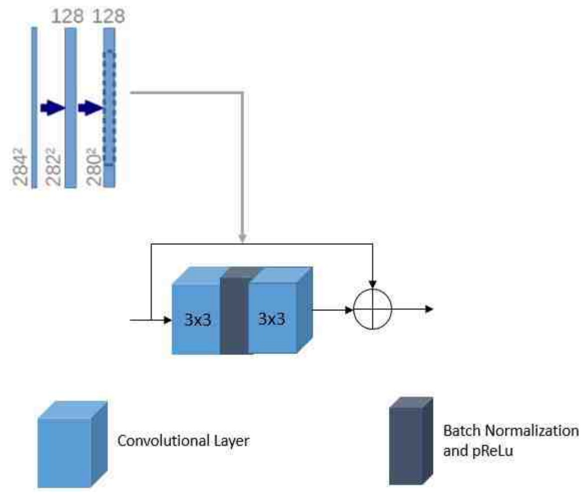


Fig. 3.2.: Convolution blocks replaced with residual block

The residual block consists of a 3×3 convolutional layer, batch normalization and pReLu, which is followed by another 3×3 convolutional layer. The input to the first convolutional layer is added with the output of the second convolutional layer. The inventors of residual network found resnet improves training accuracy for deeper networks [56].

In addition to adding the residual block, the number of levels in the network has been reduced from 5 to 4, and the number of classes has been increased from 2 to 256. The modified network has been shown in the following figure.

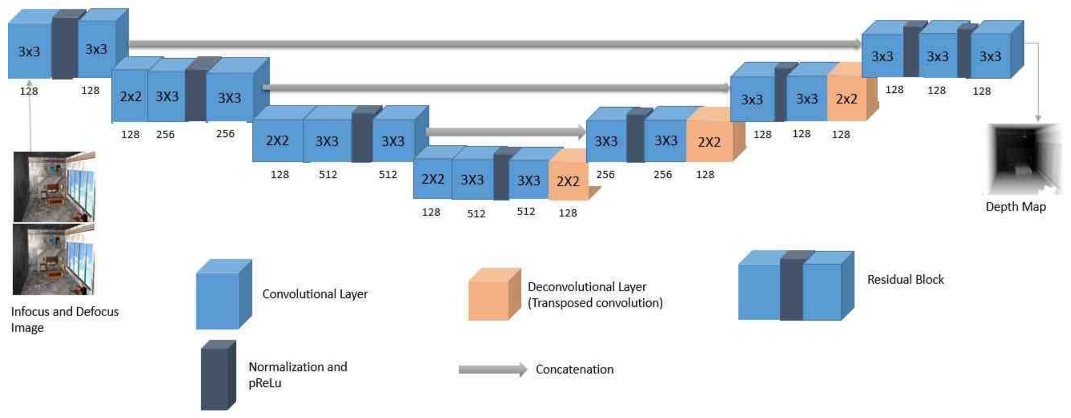


Fig. 3.3.: DfD Net

The DfD-Net architecture was built using the Dlib machine learning toolkit. The GPU we have used to train the network is NVIDIA Quadro M2000.

We have performed several experiments on graph cuts and neural nets, and found the overall performance of neural nets is better compared to graph cuts. It will be discussed in the next chapter.

4. EXPERIMENTS AND RESULTS

The experiments we have performed can be divided primarily into two sections.

- Experiments on graph cuts.
- Experiments on DfD-net.

4.1 Experiments on Graph Cuts

4.1.1 Graph Cuts on Synthetically Blurred Images

As discussed in Chapter 2, a σ_{min} and a σ_{max} are required to synthetically blur the images. In this project 0.32 and 2.88 have been chosen as σ_{min} and σ_{max}, respectively.

4.1.1.1 Middlebury

The performance of graph cuts on a synthetically blurred Middlebury image is shown here.

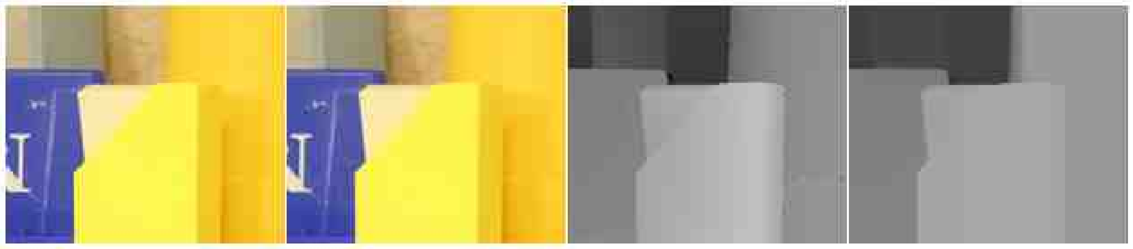


Fig. 4.1.: In-focus image, blurred image, ground truth and estimated depth map

Figure 4.1 shows an in focus image, its blurred version, the ground truth, and the depth map constructed by graph cuts. The NMAE, NRMSE, and SSIM between the ground truth and the generated depth map are 0.048789, 0.063127, and 0.976748, respectively.

Figure 4.2 demonstrates how the performance of graph cuts varies with maximum blur.

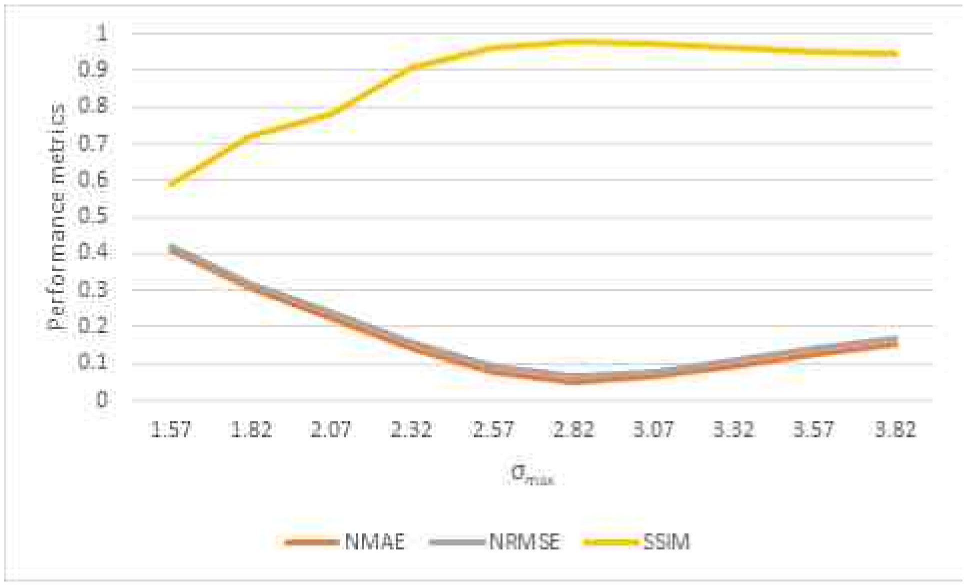


Fig. 4.2.: Graph cuts performance on a Middlebury image with respect to σ_{max}

The graph shows graph cuts gives the best performance when the value of σ_{max} is around 2.8. This is expected because blur generation in graph cuts has been achieved using the same method that was used to create the synthetically blurred images, and the σ_{max} has been chosen to be 2.88, as mentioned above. The average time for depth map generation is 1.008 minutes.

4.1.1.2 Maya

The performance of graph cuts on a synthetically blurred Maya image is shown in the Figure 4.3.

In Figure 4.3, we see an in focus image from Maya dataset, its blurred version, the ground truth, and the depth map generated by graph cuts. The NMAE, NRMSE, and

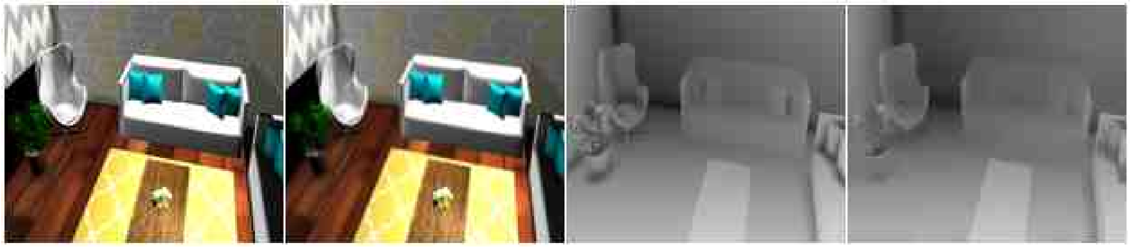


Fig. 4.3.: In-focus image, blurred image, ground truth and estimated depth map

SSIM between the ground truth and the estimated depth map are 0.019273, 0.03165, and 0.965767, respectively.

The variation of performance of graph cuts with respect to maximum sigma is shown in Figure 4.4.

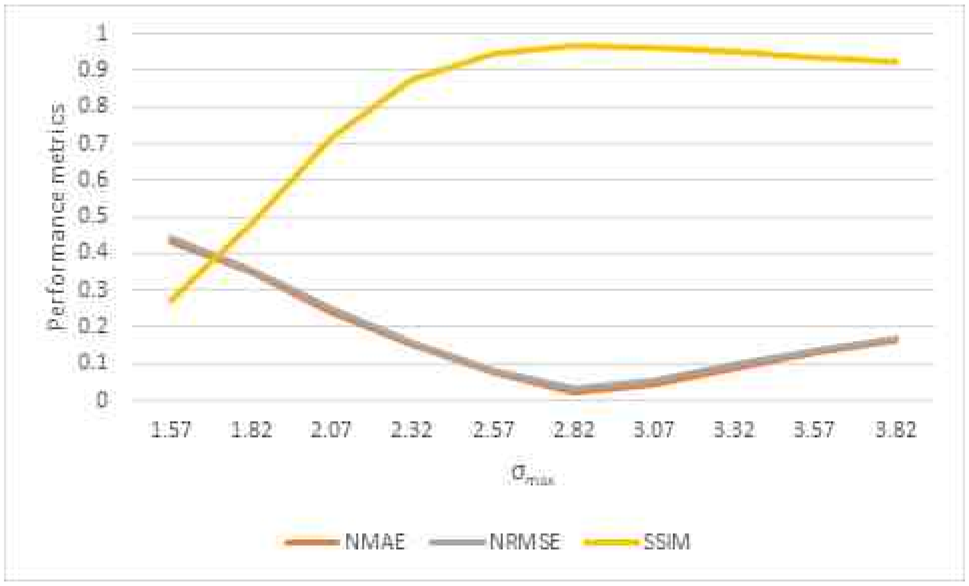


Fig. 4.4.: Graph cuts performance on a Maya image with respect to σ_{max}

Like the Middlebury image, the best performance is achieved near σ_{max} value 2.8; for the same reason described before. The average time for depth map generation is 3.134 minutes. The depth map generation time is about three times more compared to Middlebury. It is not surprising since the Maya images are also about three times bigger than Middlebury ones.

4.1.2 Graph Cuts on Photoshop Lens Blurred Images

In this section, the performance of graph cuts will be discussed with the same images shown in the last section. The only difference will be, in this case, the images will be blurred using lens blur from Photoshop instead of synthetic blur.

4.1.2.1 Middlebury

The performance of graph cuts on a lens blurred Middlebury image is shown here.

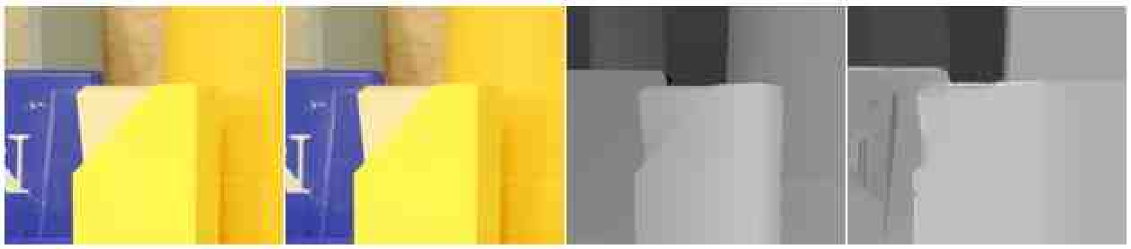


Fig. 4.5.: In-focus image, blurred image, ground truth and estimated depth map

Figure 4.5 shows an in focus image, its blurred version, the ground truth, and the depth map constructed by graph cuts. The NMAE, NRMSE, and SSIM between the ground truth and the generated depth map are 0.096849, 0.125599, and 0.896955, respectively.

Figure 4.6 demonstrates how the performance of graph cuts varies with maximum blur.

It can be seen by comparing Figure 4.1 with Figure 4.5 and Figure 4.2 and Figure 4.6 that the performance of graph cuts on synthetically blurred images is better than that on lens blurred images. This is because the blurring process of graph cuts is the same as the synthetic blur. On the other hand, the blurring process of lens blur is not known. And as discussed in Chapter 2, although it is close to synthetic blur, they have some differences.

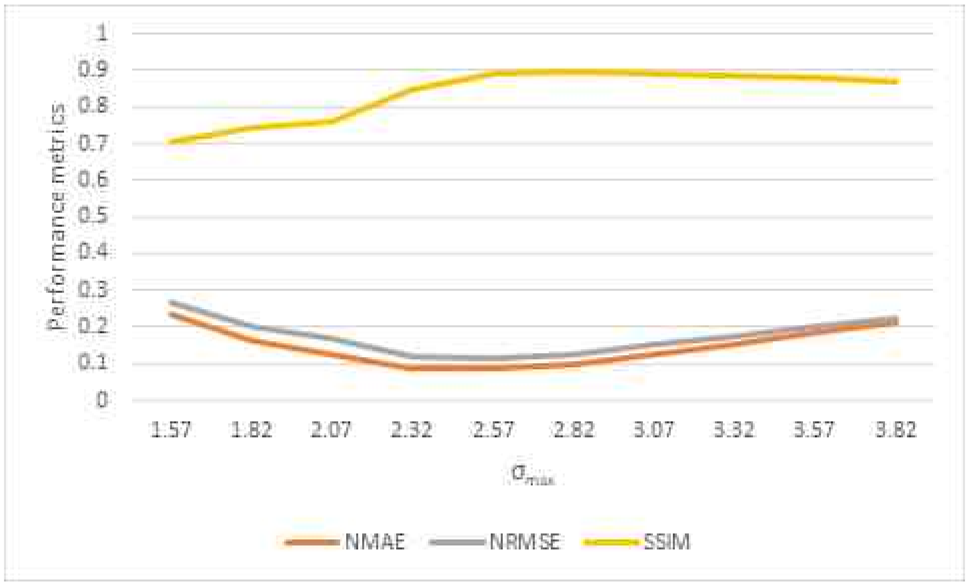


Fig. 4.6.: Graph cuts performance on a Middlebury image with respect to σ_{max}

Another noticeable information is, the best performance of graph cuts has been found around the σ_{max} value 2.8. But there is no σ to vary in Photoshop, therefore there is no direct way to match with the σ_{max} of the synthetic blur. However, there is a parameter called 'Blur radius' in Photoshop that controls the intensity of blur. We have varied the blur radius and found a value for which the performance of the algorithm matches with the one on synthetic blur.

4.1.2.2 Maya

The performance of graph cuts on a lens blurred Maya image is shown in the Figure 4.7.

In Figure 4.7, we see the same in focus image used in Section 4.1.1.2, its blurred version, the ground truth, and the depth map generated by graph cuts. The NMAE, NRMSE, and SSIM between the ground truth and the estimated depth map are 0.078003, 0.104065, and 0.827061, respectively.

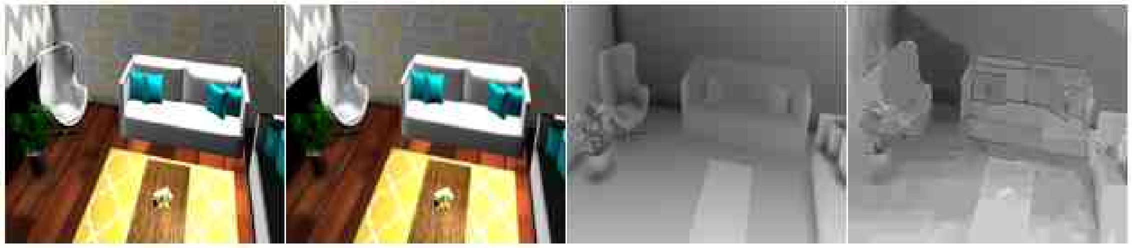


Fig. 4.7.: In-focus image, blurred image, ground truth and estimated depth map

The variation of performance of graph cuts with respect to maximum sigma is shown in Figure 4.8.

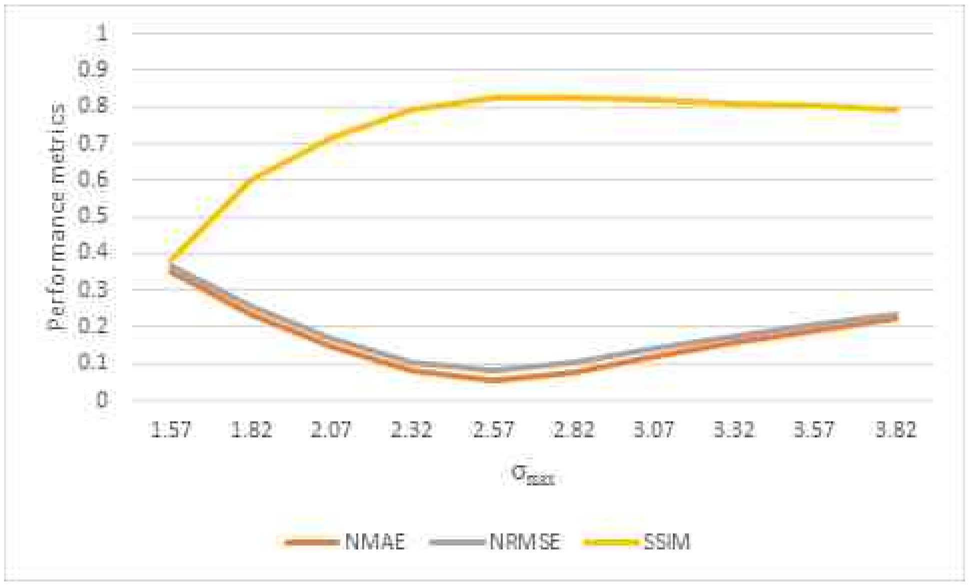


Fig. 4.8.: Graph cuts performance on a Maya image with respect to σ_{max}

As discussed in the last section for Middlebury images, here we can see as well by comparing the images in Figure 4.3 and Figure 4.7 and the graphs in Figure 4.4 and Figure 4.8 that the performance of graph cuts on lens blur is worse compared to synthetic blur; for the reasons mentioned above. However, like the Middlebury images, here we have been able to match the algorithm with synthetic blur in terms of σ_{max}.

We have applied graph-cuts to 54 Middlebury images and 35 Maya images. These images have been used to test the DfD-net as well. The results are summarized in Table 4.1.

Table 4.1.: Graph Cuts Performance: Middlebury vs Maya

Performance Metrics (Average)	Middlebury (54 images)	Maya (35 images)
NMAE	0.024832±0.000139	0.024357±0.00014
NRMSE	0.046687±0.000211	0.038209±0.0002
SSIM	0.9576±0.00046	0.919477±0.027331

4.2 Experiments on DfD-net

4.2.1 Division of Datasets

For performing the experiments on the DfD-net, both the datasets (Middlebury and Maya) have been divided into train and test sets.

There are 486 images in the Middlebury dataset. Among these images, 432 images have been chosen for the training dataset and the rest of the 54 are for the test dataset. While dividing the datasets, the important consideration was to make sure the depth map values of the train and test sets overlap so that the trained network can perform well with the test images. It has been demonstrated in the following figure-

It can be seen from Figure 4.9 that the depth map values of the test images significantly overlap with those of the train images.

In the Maya dataset, there are 505 images. Among them, 434 images have been selected as train images and rest are as test images. The depth map distribution for the Maya images is shown here-

There is substantial overlap of the depth map values between the train and test images in this case as well. However, one noticeable difference is, for Maya images,

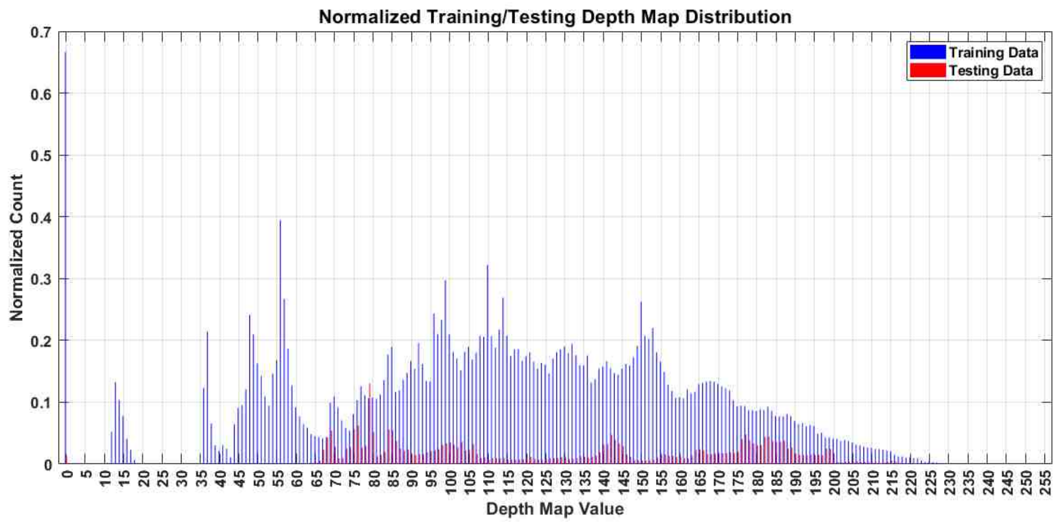


Fig. 4.9.: Depth map distribution for train and test images for Middlebury dataset

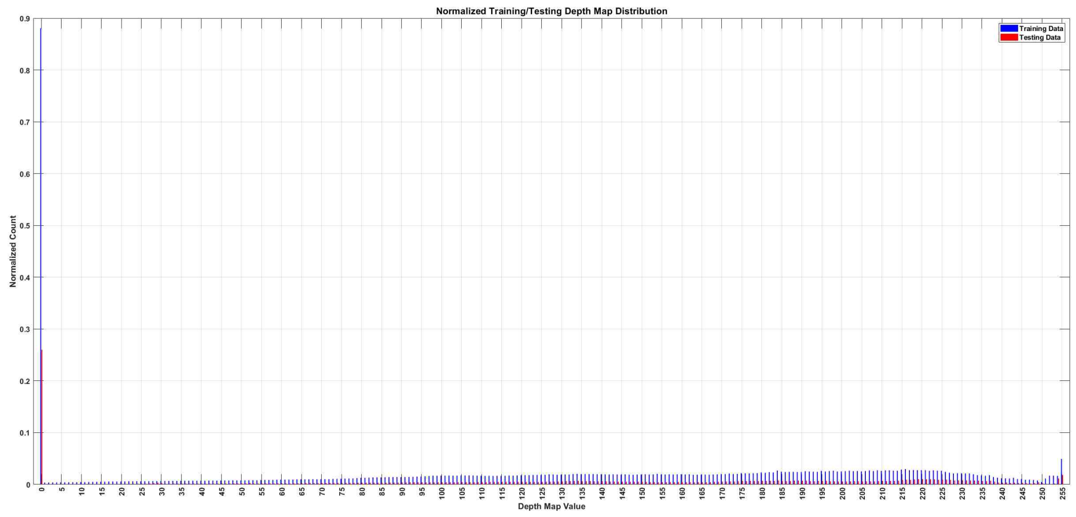


Fig. 4.10.: Depth map distribution for train and test images for Maya dataset

the depth map value ‘0’ is highly frequent; whereas the other values are more evenly distributed.

In addition to training and testing a network with the same dataset, we have experimented how a network trained with one dataset performs while testing with the other. For example, a network trained with Middlebury images have been tested with Maya dataset, and vice versa. Therefore, it is important to find the depth map distribution of training and test images from two different datasets.

In the following figure the depth map distribution of Maya train images and Middlebury test images is shown.

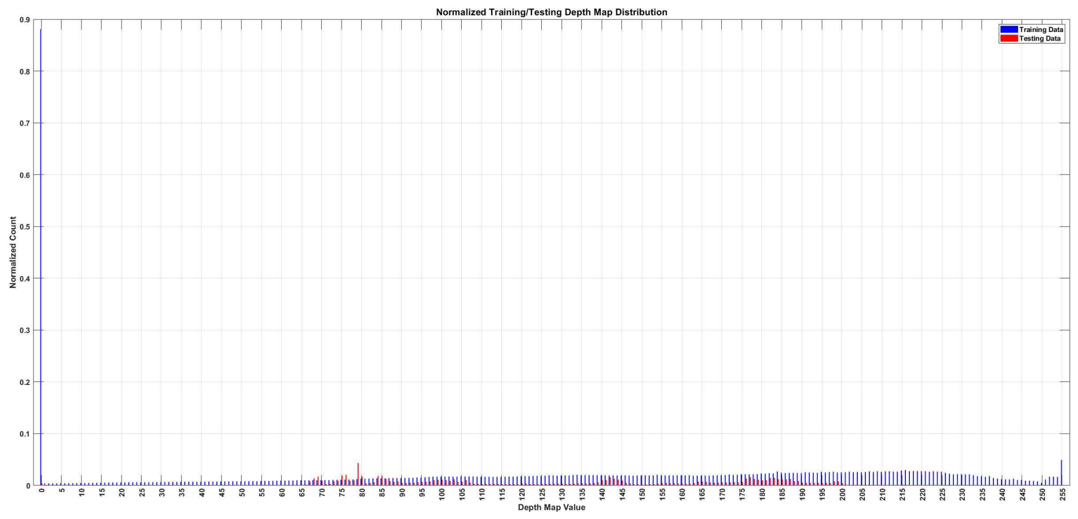


Fig. 4.11.: Depth map distribution for Maya train/Middlebury test dataset

As seen in Figure 4.11, the depth map values of the Middlebury images fit well inside the Maya images. It is not surprising, since we have seen in Figure 4.10 that the depth map distribution of the Maya training images is significantly wide; covering all the values from 0 to 255. On the other hand, according to Figure 4.9, the depth map distribution of the Middlebury test images are comparatively narrower, ranging between 70 and 200.

Like the Maya train/Middlebury test depth map distribution, we need to examine the depth map distribution of the other way around as well; i.e., the distribution of the Middlebury training images and Maya test images. Figure 4.12 illustrates this-

It can be seen from Figure 4.12 that a large number of test depth map values are outside the range of the training ones. Again, this is not much surprising given

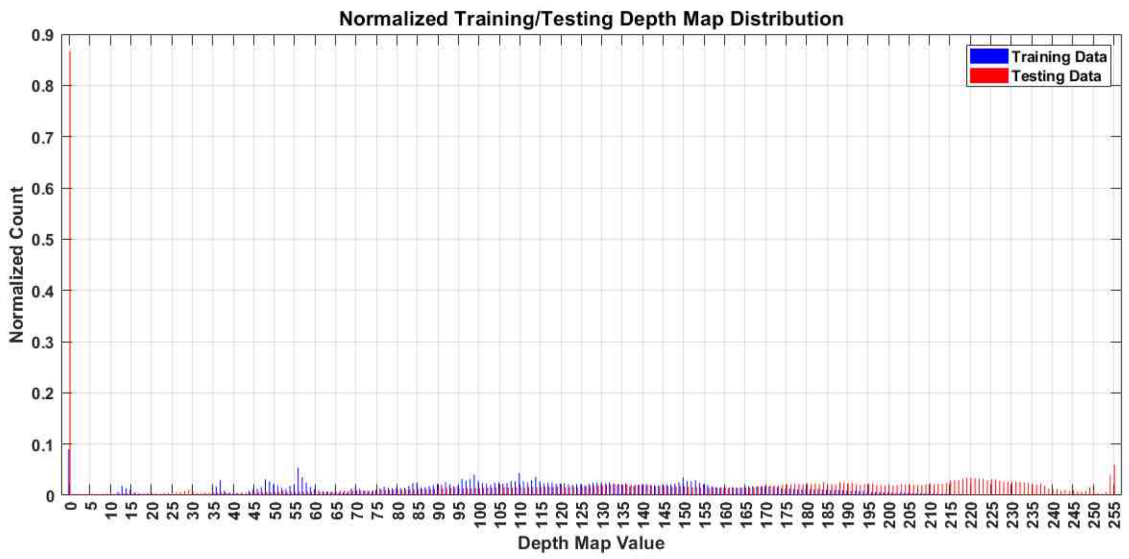


Fig. 4.12.: Depth map distribution for Middlebury train/Maya test dataset

the fact that the distribution of Maya depth values is wider than that of Middlebury. Anyway, this is not what we want; because the depth values of the training and test images should overlap to obtain the desired performance. Therefore, we experimented with the test images and found the ones whose depth values fit inside the depth values of the training images. The modified depth map distribution is shown in Figure 4.13.

4.3 Performance of DfD-net with Synthetically Blurred Images

In this section, we are going to discuss the performance of DfD-net using the synthetically blurred images as the dataset. At first, the performance of the network trained with the Maya images is presented.

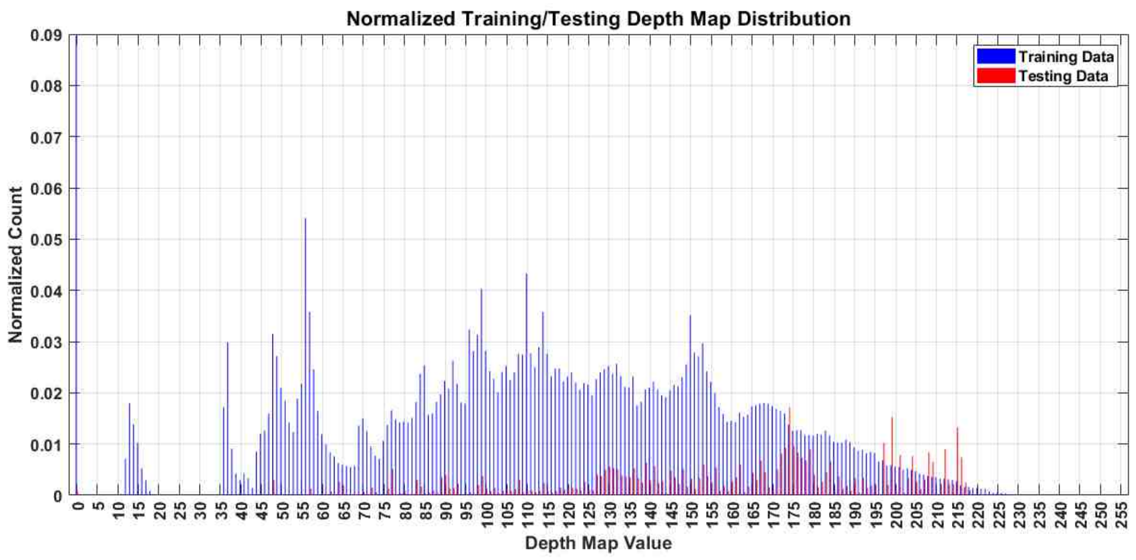
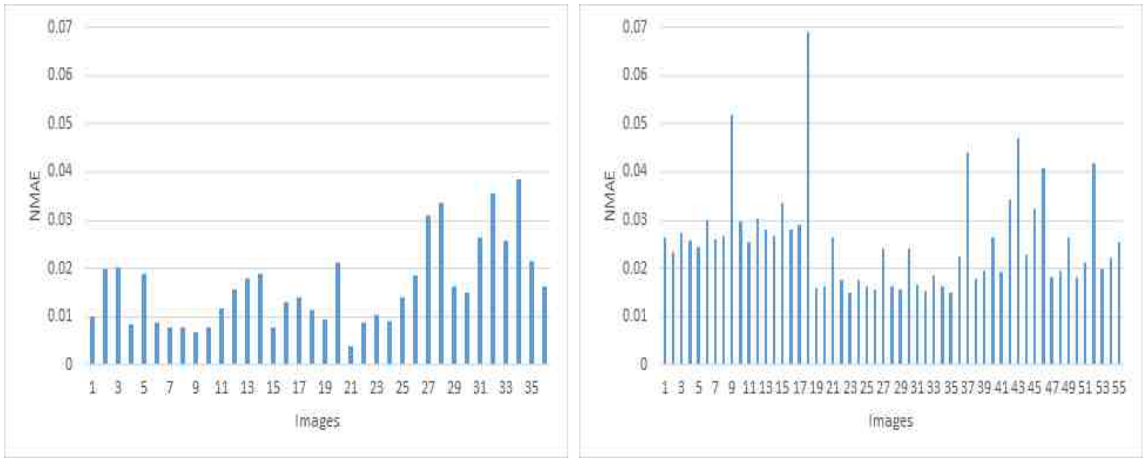


Fig. 4.13.: Modified depth map distribution for Middlebury train/Maya test dataset



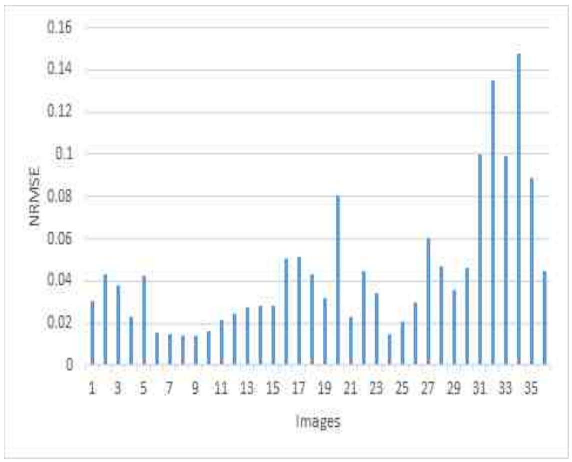
(a) NMAE performance of DfD-net trained and tested with Maya images

(b) NMAE performance of DfD-net trained with Maya and tested with Middlebury images

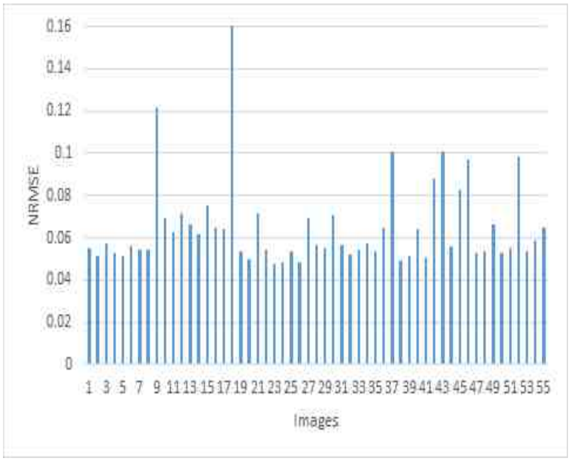
Fig. 4.14.: NMAE performance of DfD-net trained with Maya images

4.3.1 Performance of DfD-net trained with Maya images

Figure 4.14, 4.15, and 4.16 show the performance of the Maya trained network on three metrics and with two test datasets. It is evident from the figures that the network performs well for both Maya and Middlebury test data.

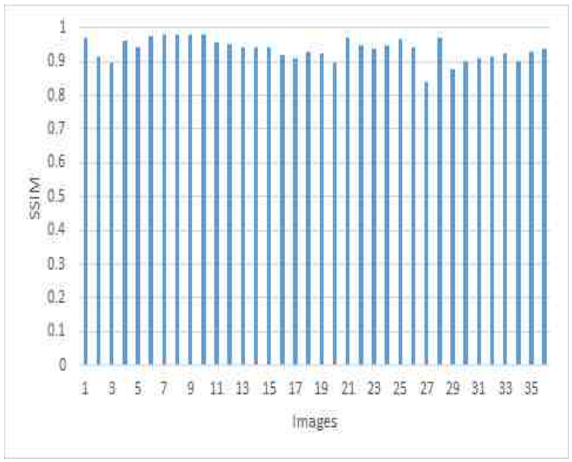


(a) NRMSE performance of DfD-net trained and tested with Maya images

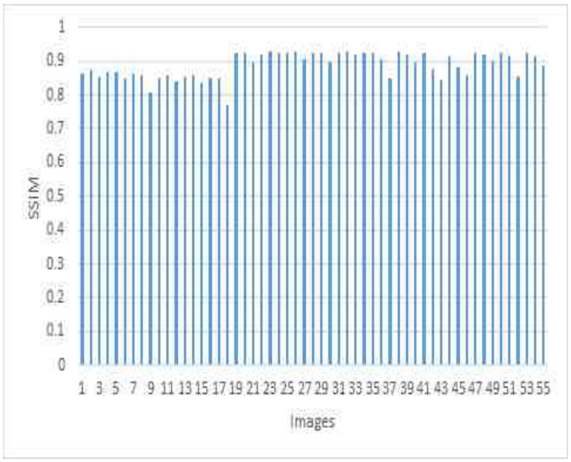


(b) NRMSE performance of DfD-net trained with Maya and tested with Middlebury images

Fig. 4.15.: NRMSE performance of DfD-net trained with Maya images



(a) SSIM performance of DfD-net trained and tested with Maya images



(b) SSIM performance of DfD-net trained with Maya and tested with Middlebury images

Fig. 4.16.: SSIM performance of DfD-net trained with Maya images

The average NMAE, NRMSE, and SSIM for the Maya test images are 0.016166, 0.044702, and 0.936073, respectively. The standard deviations of the same are 0.00866, 0.03295868, and 0.032393725.

For the Middlebury dataset, the average NMAE, NRMSE, and SSIM are 0.025532, 0.064937, and 0.888098, respectively. The standard deviations of those metrics are 0.010276573, 0.020526833, and 0.037547284.

It took 15.919270 hours to train the network. The average time to generate the depth map for Maya and Middlebury images are 0.551711571 seconds and 0.598093736 seconds, respectively.

Among the generated depth maps, the best two and worst two are presented in the Figures 4.17-4.24.

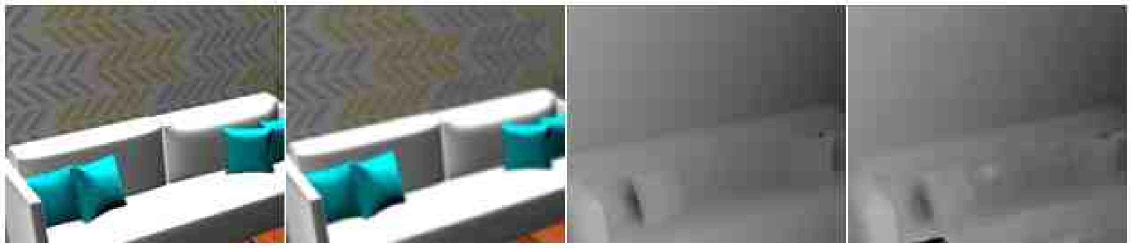


Fig. 4.17.: Best performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

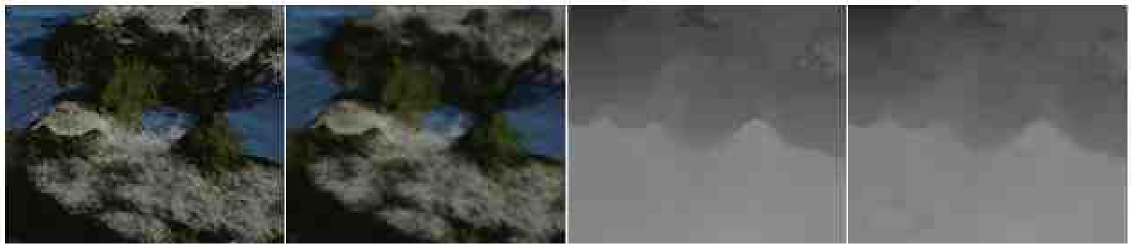


Fig. 4.18.: Second best performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

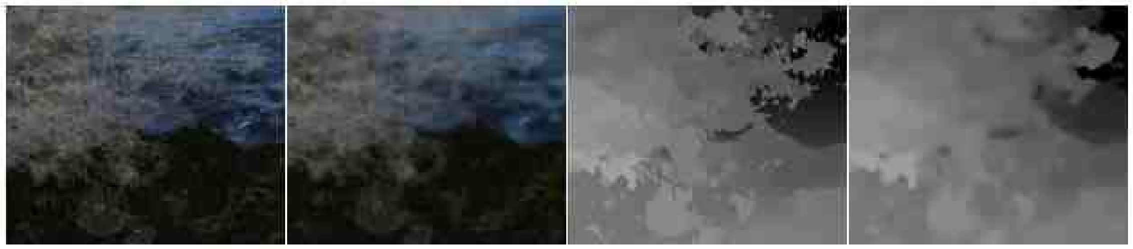


Fig. 4.19.: Worst performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

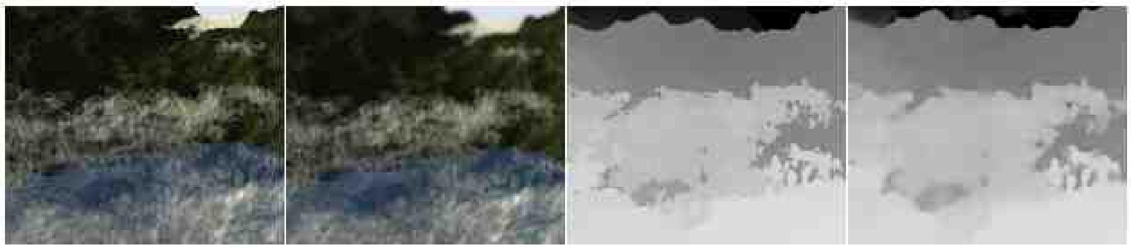


Fig. 4.20.: Second worst performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

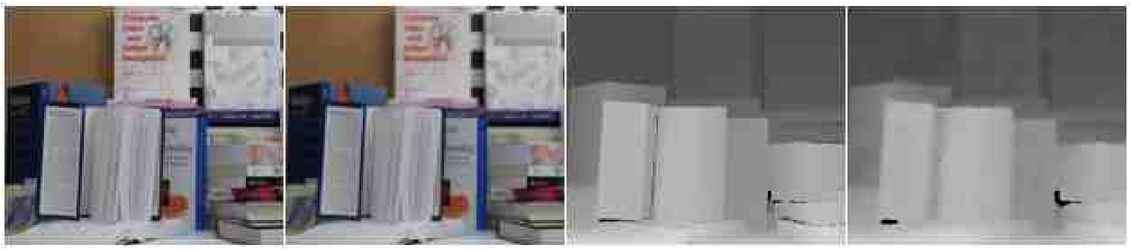


Fig. 4.21.: Best performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map

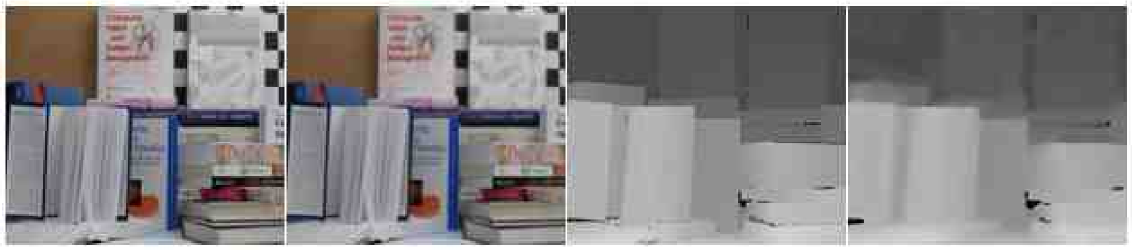


Fig. 4.22.: Second best performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map

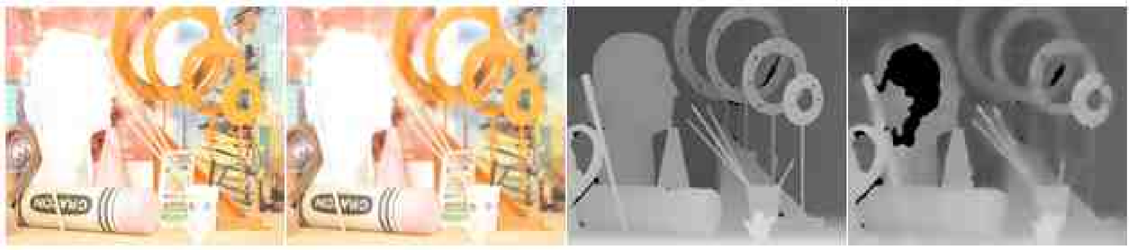


Fig. 4.23.: Worst performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map

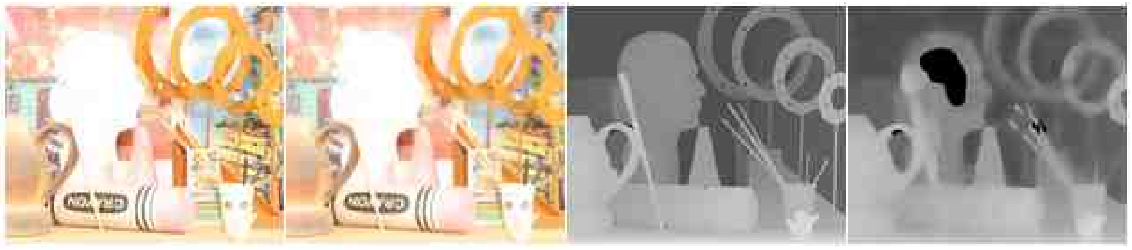
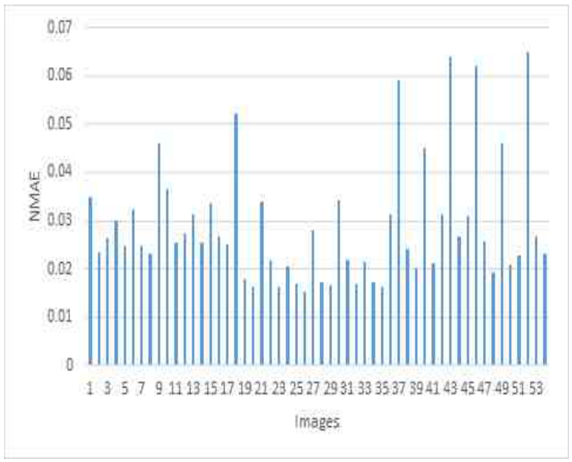


Fig. 4.24.: Second worst performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map

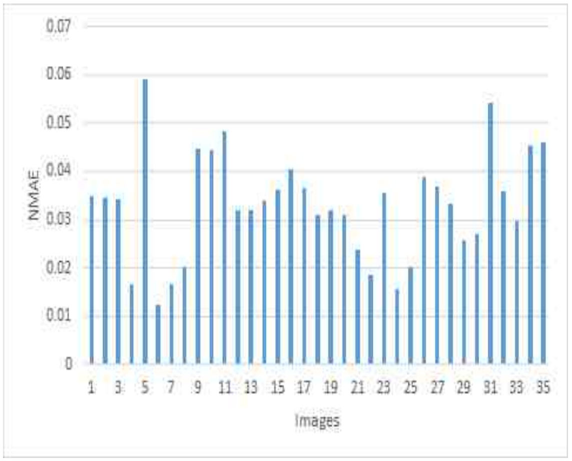
Middlebury is a stereo dataset. For each scene, there are two views. The network has generated best pair of images for the same scene. Only the view is different. The same goes for the worst pair of images as well.

4.3.2 Performance of DfD-net trained with Middlebury images

In this section, the performance of the DfD-net trained with Middlebury images is demonstrated.

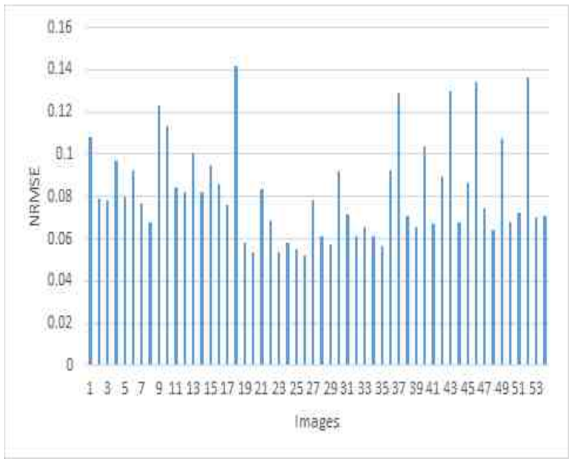


(a) NMAE performance of DfD-net trained and tested with Middlebury images

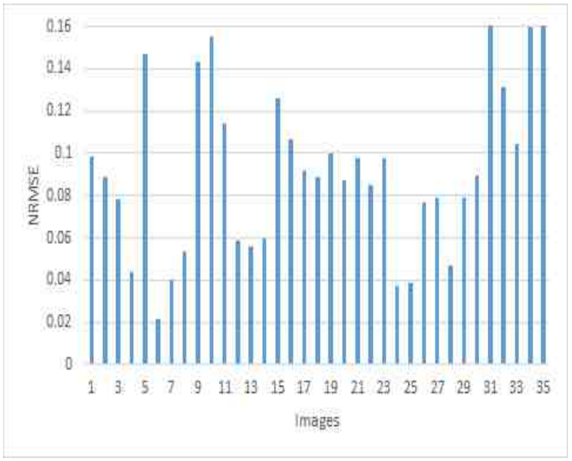


(b) NMAE performance of DfD-net trained with Middlebury and tested with Maya images

Fig. 4.25.: NMAE performance of DfD-net trained with Middlebury images

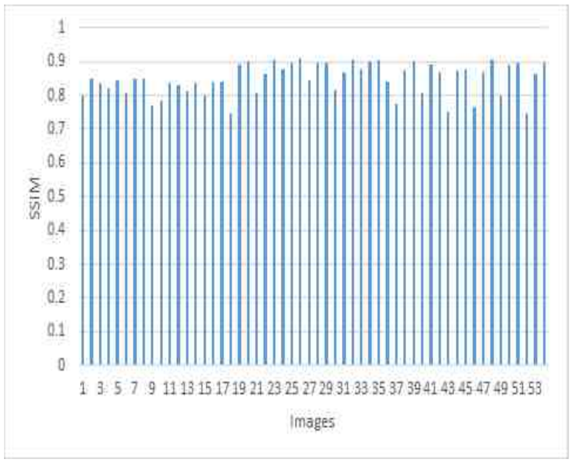


(a) NRMSE performance of DfD-net trained and tested with Middlebury images

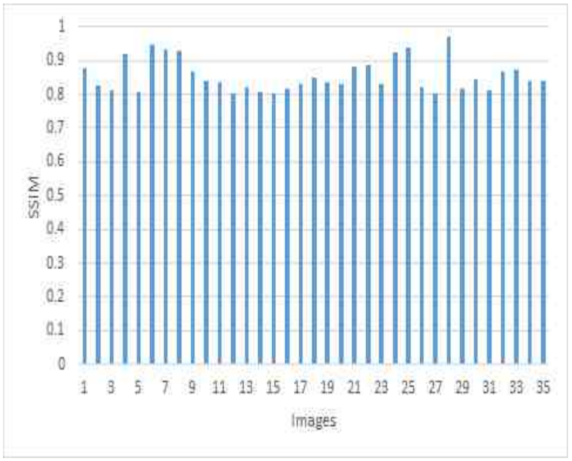


(b) NRMSE performance of DfD-net trained with Middlebury and tested with Maya images

Fig. 4.26.: NRMSE performance of DfD-net trained with Middlebury images



(a) SSIM performance of DfD-net trained and tested with Middlebury images



(b) SSIM performance of DfD-net trained with Middlebury and tested with Maya images

Fig. 4.27.: SSIM performance of DfD-net trained with Middlebury images

The average NMAE, NRMSE, and SSIM for the Middlebury test images are 0.028985019, 0.082491167, and 0.857652463, respectively. The standard deviation of the same are 0.012600189, 0.023377454, and 0.04656054.

For the Maya dataset, the average NMAE, NRMSE, and SSIM are 0.033088514, 0.091880914, and 0.8552404, respectively. The standard deviation of those metrics are 0.010991395, 0.039296738, and 0.047890585.

The time required to train the network is 25.371153 hours. The average time to generate the depth map for Middlebury and Maya images are 0.602062093 seconds and 0.5495642 seconds, respectively.

Among the generated depth maps, the best two and worst two are presented in the 4.28-4.35.

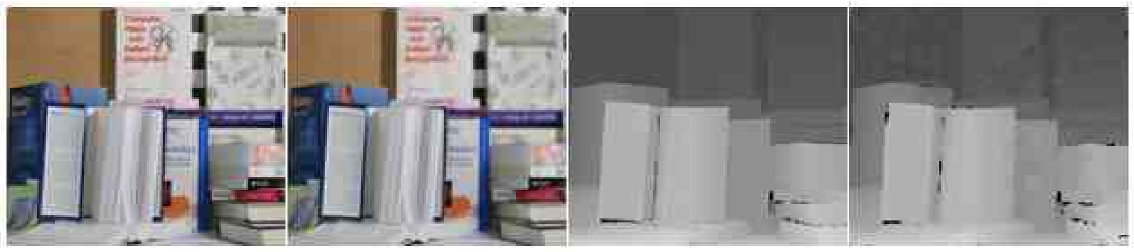


Fig. 4.28.: Best performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map

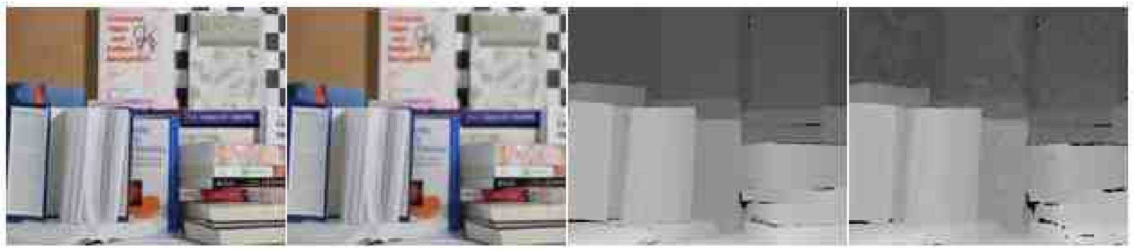


Fig. 4.29.: Second best performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map

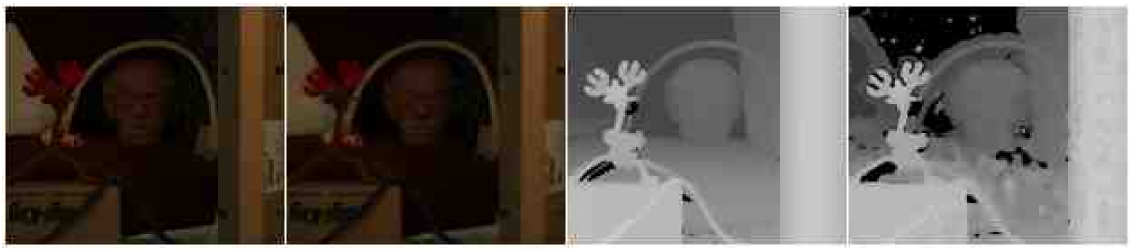


Fig. 4.30.: Worst performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map

From the results of this section, it is clear that the network trained with Middlebury images perform well for both Middlebury and Maya dataset. However, the performance of the network trained with Maya images is slightly better. Another notable observation is, in some cases, Maya and Middlebury trained network have given best and worst performances for same images.

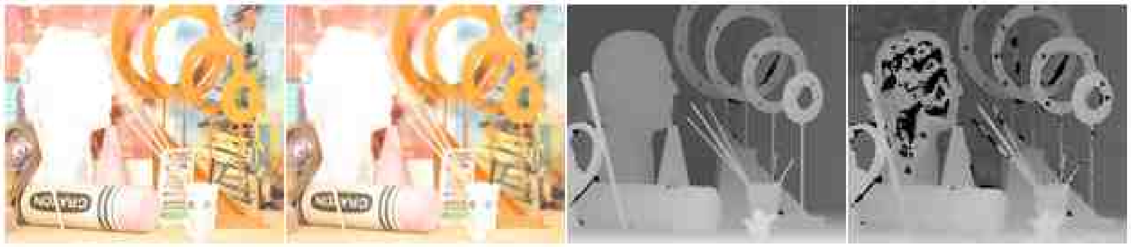


Fig. 4.31.: Second worst performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map

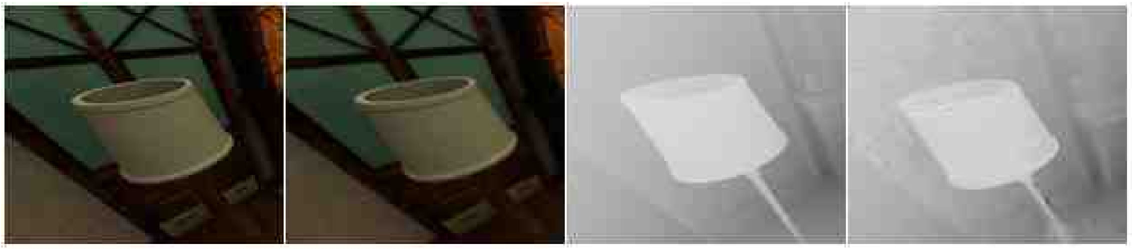


Fig. 4.32.: Best performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

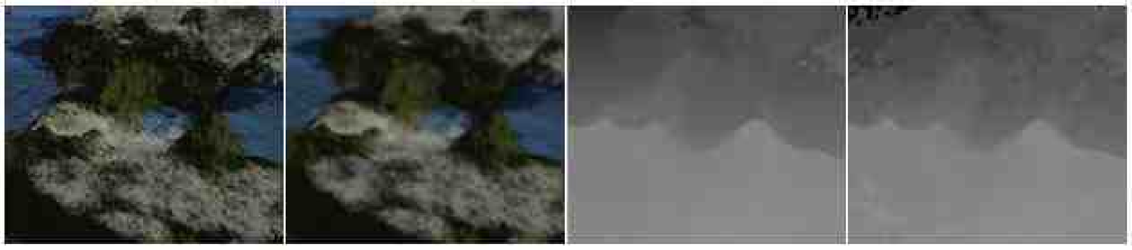


Fig. 4.33.: Second best performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

4.3.3 Performance of DfD-net trained with Maya and Middlebury images together

In this experiment, we have made a training dataset by combining Maya and Middlebury training images and trained the network with it. Its performance is presented in Figures 4.36-4.38.

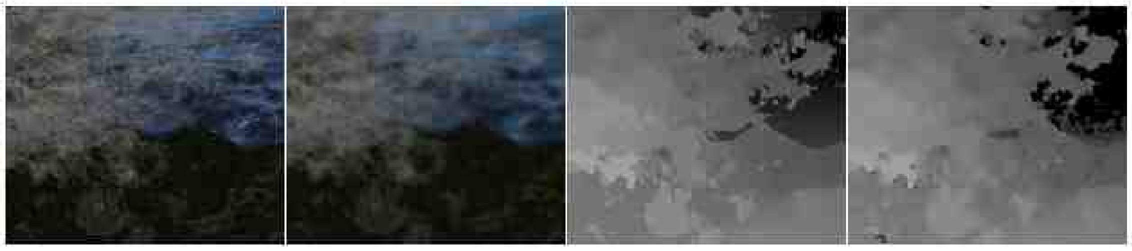


Fig. 4.34.: Worst performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

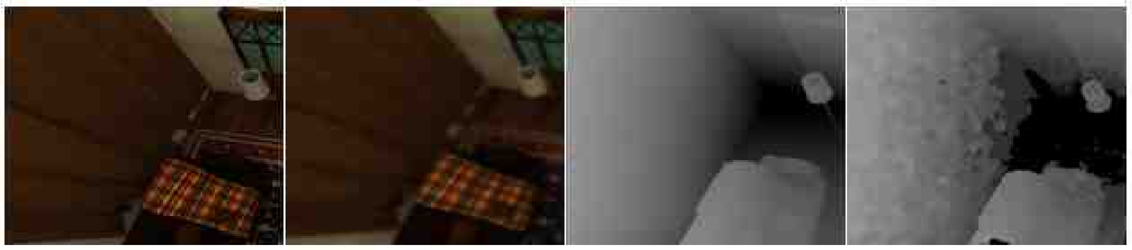


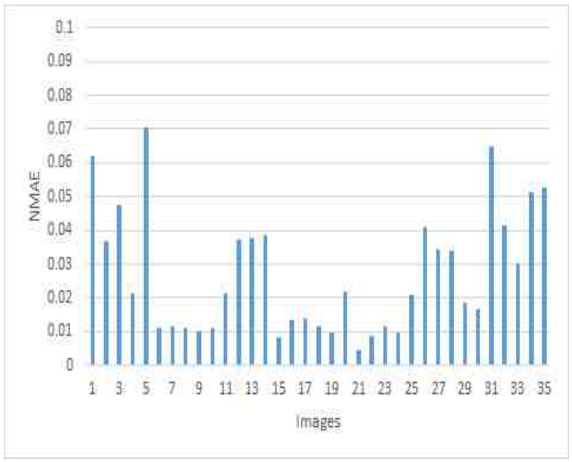
Fig. 4.35.: Second worst performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

The average NMAE, NRMSE, and SSIM for the Maya test images are 0.027040171, 0.074541029, and 0.884404857, respectively. The standard deviations of the same are 0.018258927, 0.051161158, and 0.066497327.

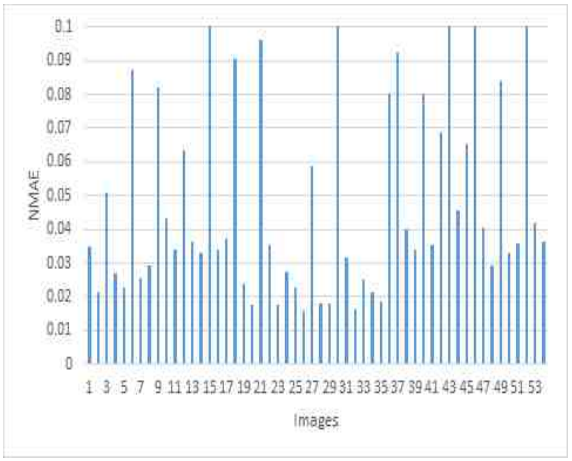
For the Middlebury dataset, the average NMAE, NRMSE, and SSIM are 0.047832, 0.112379556, and 0.791881389, respectively. The standard deviations of those metrics are 0.02881391, 0.05676711, and 0.097291282.

The time required to train the network is 57 hours. The average time to generate the depth map for Maya and Middlebury images are 0.553194543 seconds and 0.543485556 seconds, respectively.

Among the generated depth maps, the best two and worst two are presented in the 4.39-4.46.

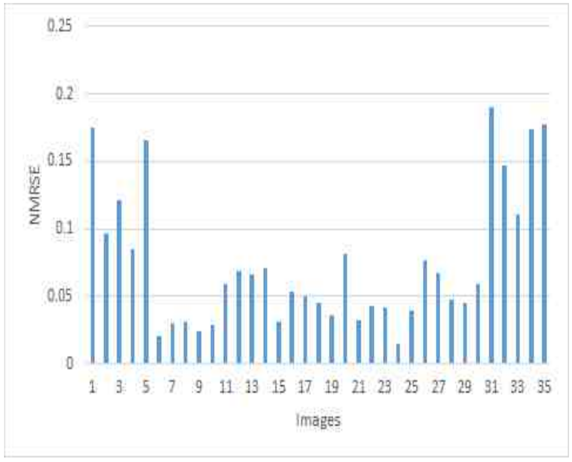


(a) NMAE performance of DfD-net trained with both dataset and tested with Maya

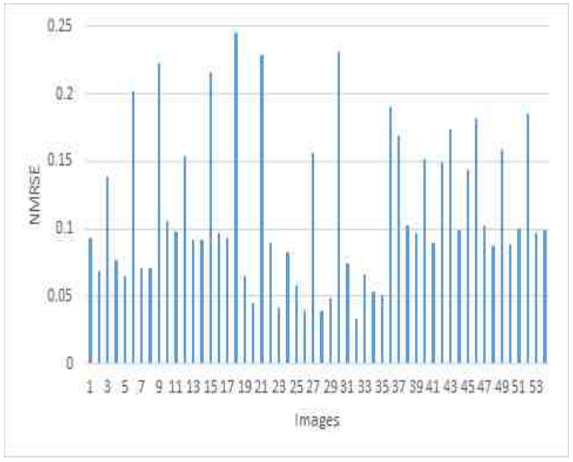


(b) NMAE performance of DfD-net trained with both dataset and tested with Middlebury

Fig. 4.36.: NMAE performance of DfD-net trained with Middlebury and Maya images together

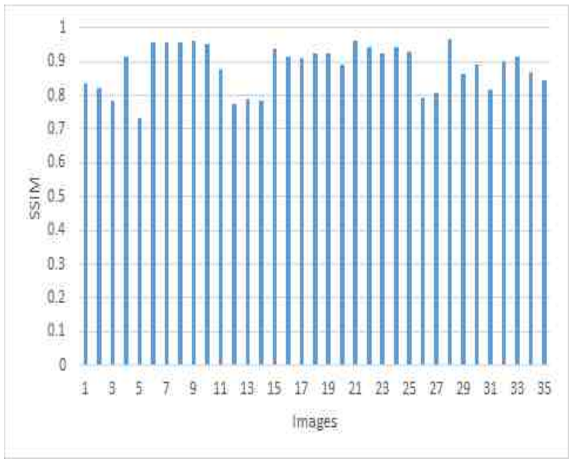


(a) NRMSE performance of DfD-net trained with both dataset and tested with Maya

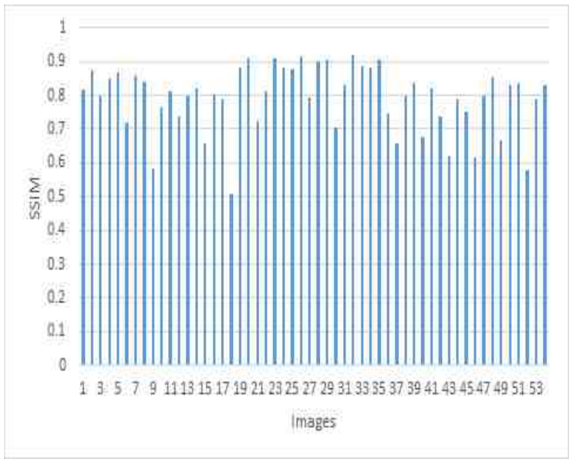


(b) NRMSE performance of DfD-net trained with both dataset and tested with Middlebury

Fig. 4.37.: NRMSE performance of DfD-net trained with Middlebury and Maya images together



(a) SSIM performance of DfD-net trained with both dataset and tested with Maya



(b) SSIM performance of DfD-net trained with with both dataset and tested with Maya

Fig. 4.38.: SSIM performance of DfD-net trained with Middlebury and Maya images together

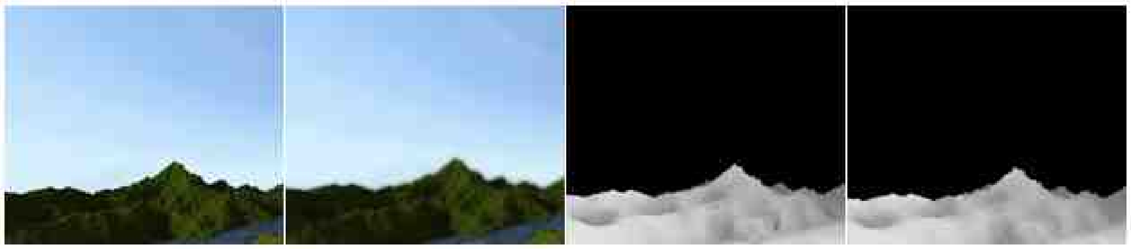


Fig. 4.39.: Best performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

The results from section 4.3.1, 4.3.2, and 4.3.3 are summarized in Tables 4.2, 4.3, and 4.4.

A couple of things are noticeable from the results. Firstly, the network trained with Maya images performs the best for both datasets. It performs better than the Middlebury trained network even for the Middlebury test images. While the joint trained network produces better results for Maya images, for the Middlebury data, Middlebury trained network is superior.

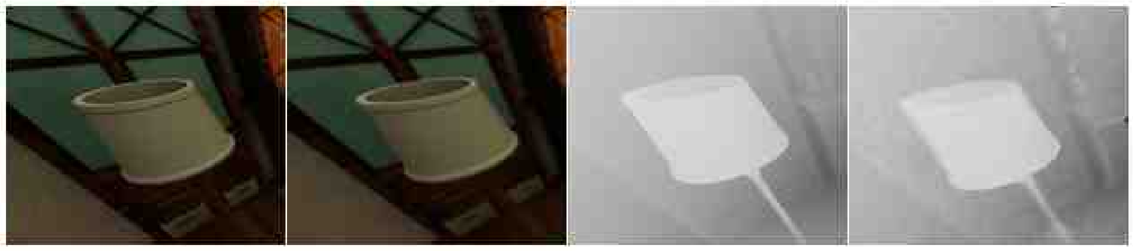


Fig. 4.40.: Second best performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

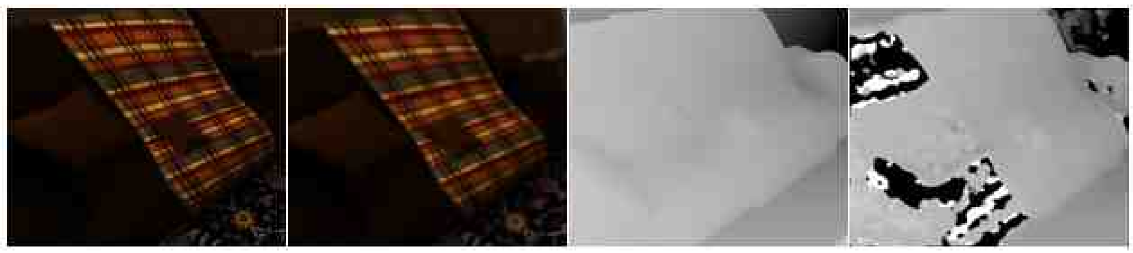


Fig. 4.41.: Worst performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

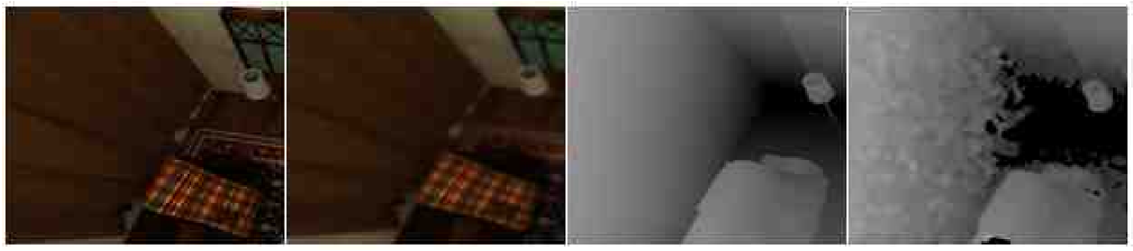


Fig. 4.42.: Second worst performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

Secondly, the Maya-trained and joint-trained network perform better for Maya images than the Middlebury ones. Even the Middlebury-trained network's performance for Maya dataset is as good as for the Middlebury.

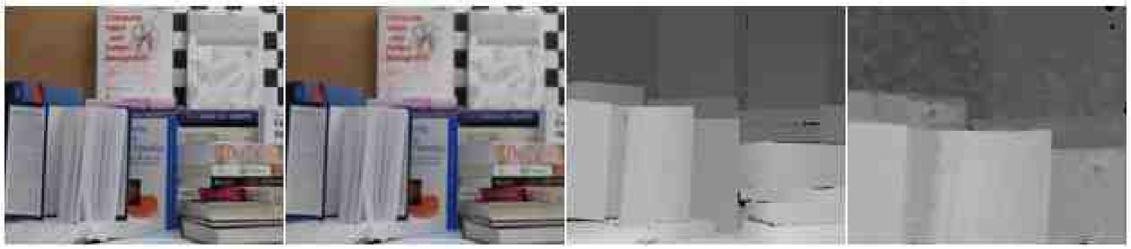


Fig. 4.43.: Best performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map

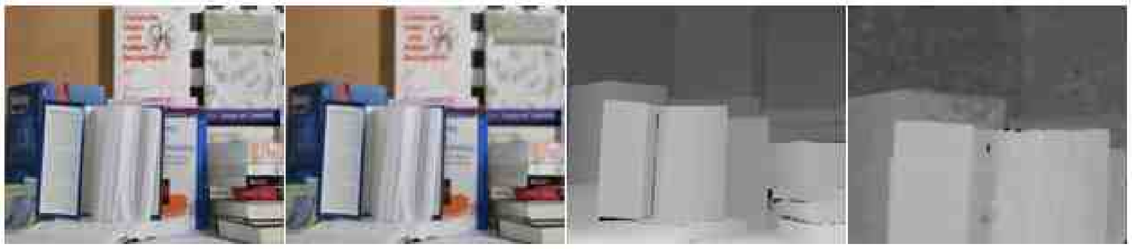


Fig. 4.44.: Second best performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map

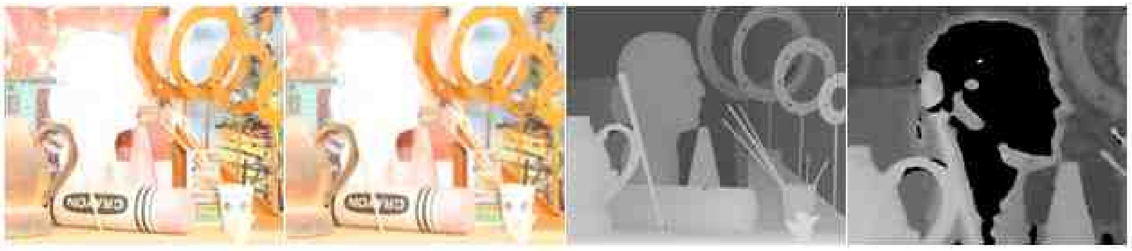


Fig. 4.45.: Worst performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map

4.4 Performance of DfD-net with Photoshop Lens Blurred Images

In this section, we are going to examine the performance of DfD-net using the Photoshop lens blurred images as the dataset. At first, the performance of the network trained with the Maya images is shown.

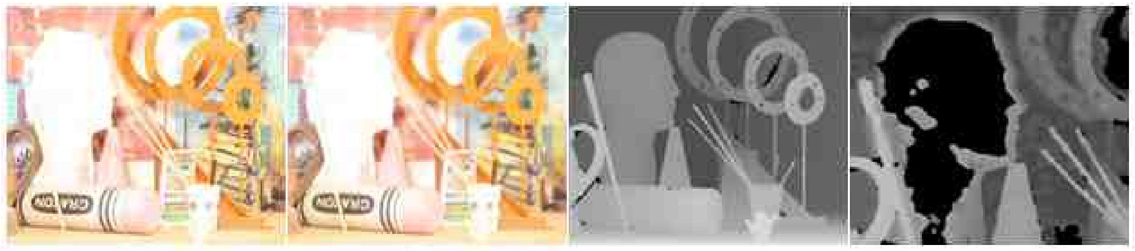


Fig. 4.46.: Second worst performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map

Table 4.2.: Maya trained network performance

Performance Metrics (Average)	Maya	Middlebury
NMAE	0.016166±0.00007	0.025532±0.0001
NRMSE	0.044702±0.001	0.064937±0.0004
SSIM	0.936073±0.001	0.888098±0.0014

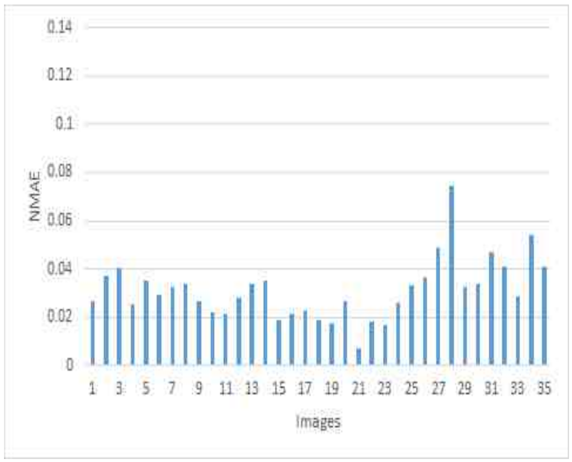
Table 4.3.: Middlebury trained network performance

Performance Metrics (Average)	Maya	Middlebury
NMAE	0.03308851±0.000117	0.02898502±0.000156
NRMSE	0.09188091±0.0015	0.08249117±0.000536
SSIM	0.85524±0.00222	0.85765246±0.002128

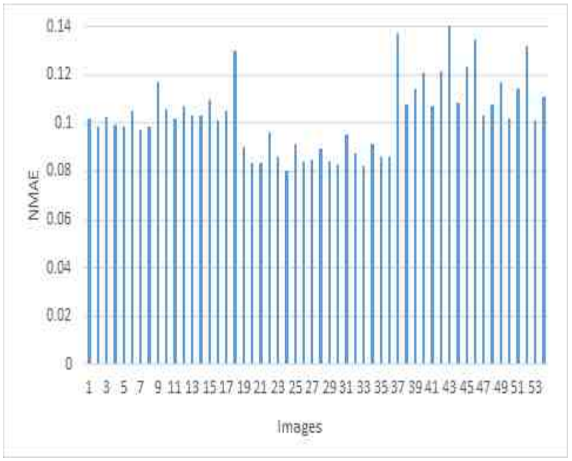
Table 4.4.: Joint trained network performance

Performance Metrics (Average)	Maya	Middlebury
NMAE	0.02704017±0.000467	0.04783232±0.001566
NRMSE	0.07454103±0.003343	0.11237956±0.004184
SSIM	0.88440486±0.004296	0.79188139±0.00929

4.4.1 Performance of DfD-net trained with Maya images

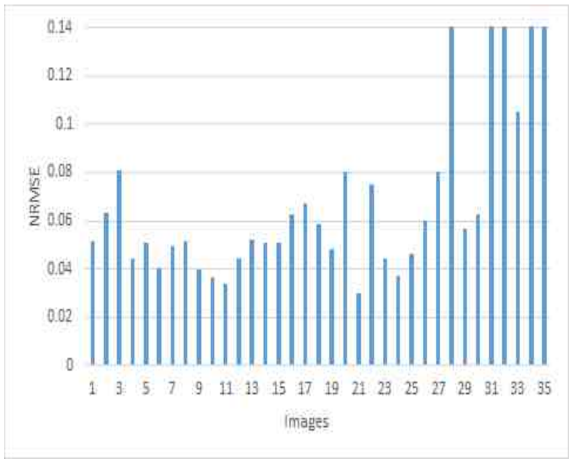


(a) NMAE performance of DfD-net trained and tested with Maya images

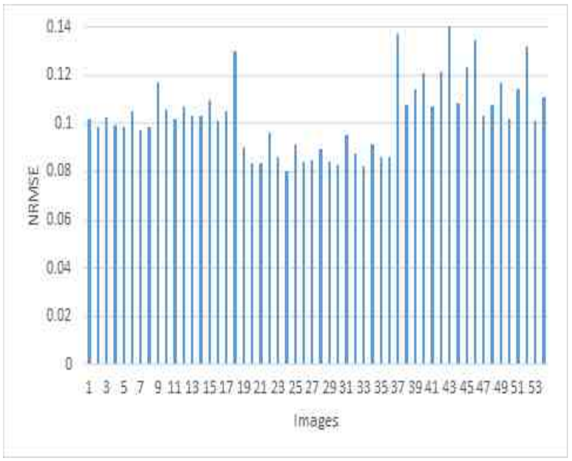


(b) NMAE performance of DfD-net trained with Maya and tested with Middlebury images

Fig. 4.47.: NMAE performance of DfD-net trained with Maya images

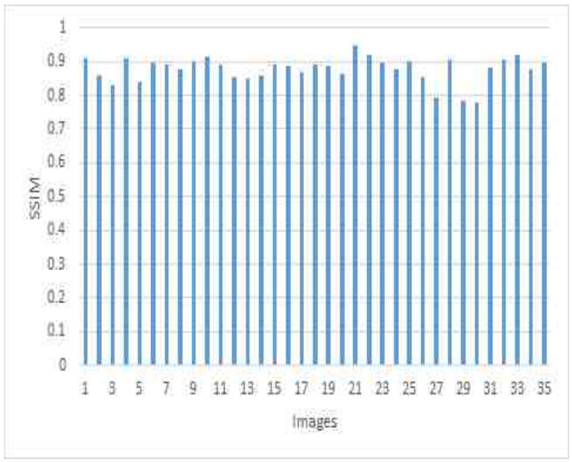


(a) NRMSE performance of DfD-net trained and tested with Maya images

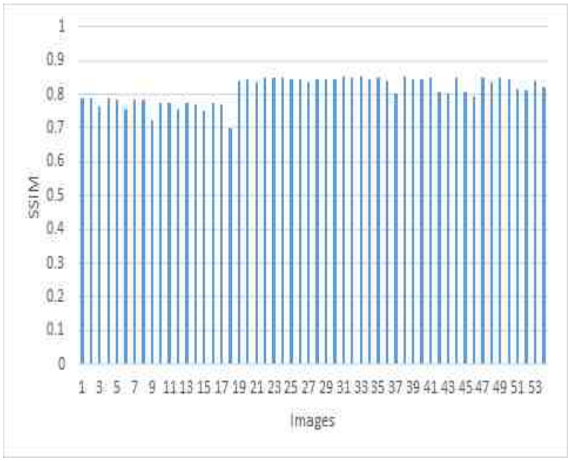


(b) NRMSE performance of DfD-net trained with Maya and tested with Middlebury images

Fig. 4.48.: NRMSE performance of DfD-net trained with Maya images



(a) SSIM performance of DfD-net trained and tested with Maya images



(b) SSIM performance of DfD-net trained with Maya and tested with Middlebury images

Fig. 4.49.: SSIM performance of DfD-net trained with Maya images

The average NMAE, NRMSE, and SSIM for the Maya test images are 0.029961088, 0.067610118, and 0.877186771, respectively. The standard deviations of the same are 0.01002877, 0.038178461, and 0.032393725.

For the Middlebury dataset, the average NMAE, NRMSE, and SSIM are 0.100774, 0.12112274, and 0.815594547, respectively. The standard deviations of those metrics are 0.013467487, 0.014264182, and 0.035549268.

It took 15.919270 hours to train the network. The average time to generate the depth map for Maya and Middlebury images are 0.551711571 seconds and 0.598093736 seconds, respectively.

Among the generated depth maps, the best two and worst two are presented in the Figures 4.50-4.57.

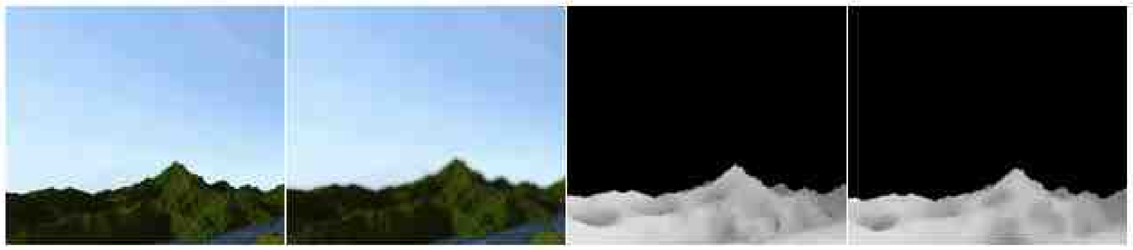


Fig. 4.50.: Best performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

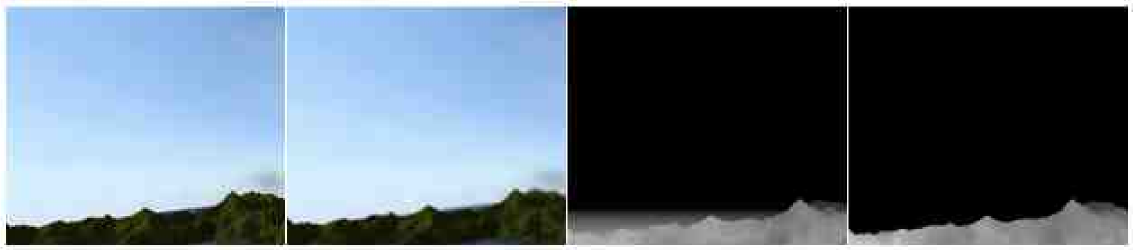


Fig. 4.51.: Second best performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

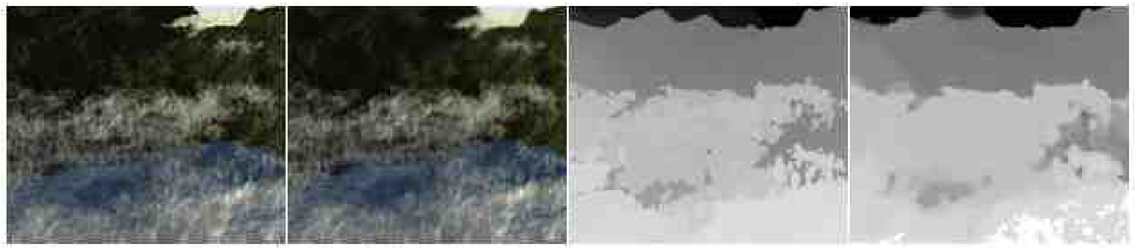


Fig. 4.52.: Worst performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

4.4.2 Performance of DfD-net trained with Middlebury images

In Figures 4.58-4.60, the performance of the DfD-net trained with Middlebury images is illustrated.

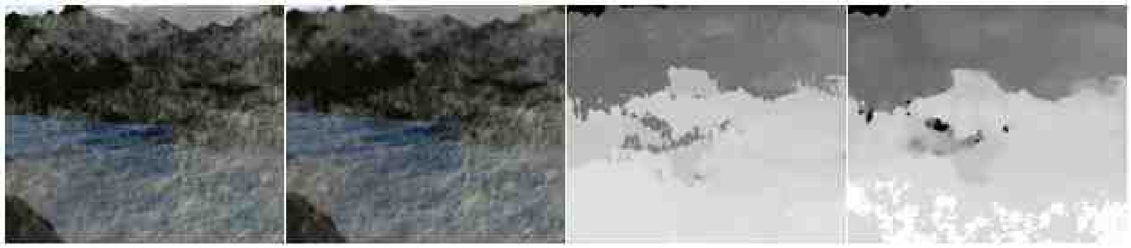


Fig. 4.53.: Second worst performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

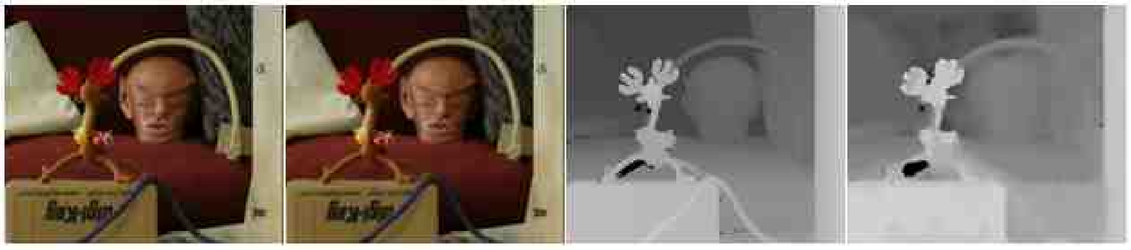


Fig. 4.54.: Best performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map

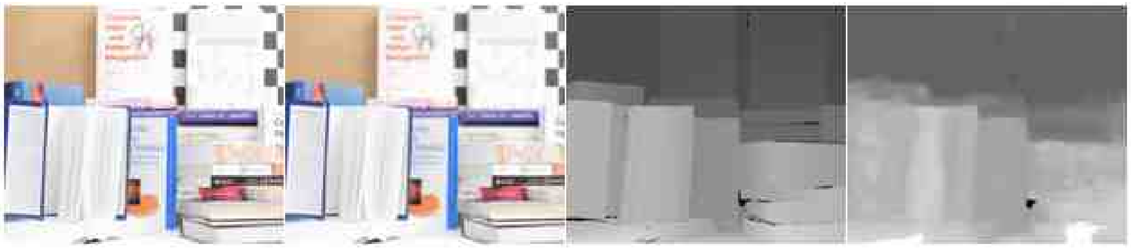


Fig. 4.55.: Second best performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map

The average NMAE, NRMSE, and SSIM for the Maya test images are 0.145088529, 0.212262194, and 0.687927485, respectively. The standard deviations of the same are 0.099089337, 0.064511737, and 0.11156528.

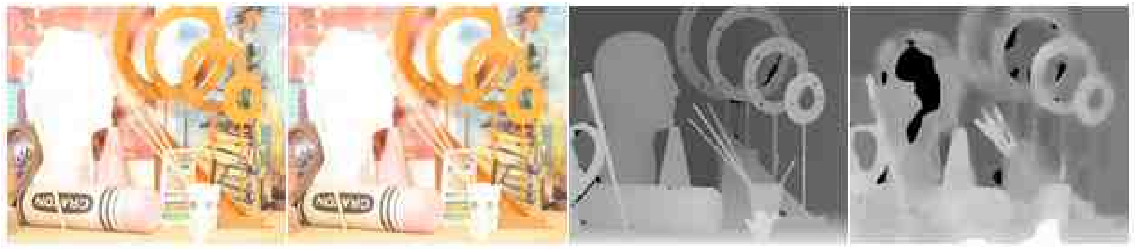


Fig. 4.56.: Worst performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map

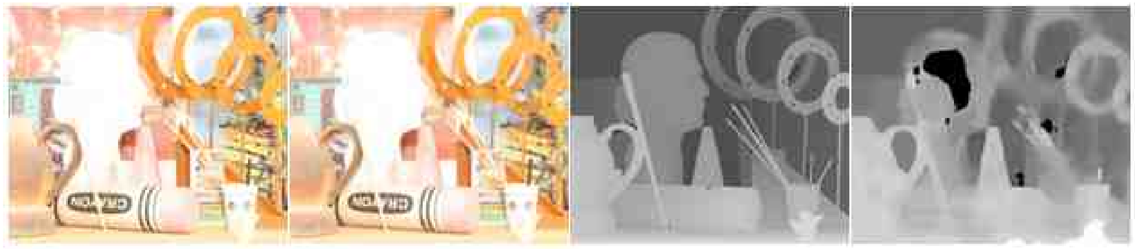
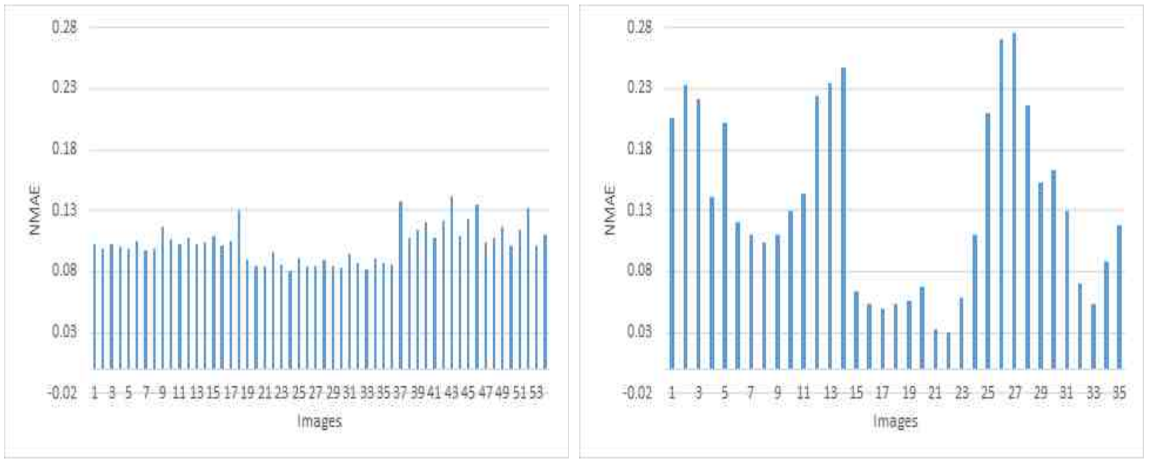


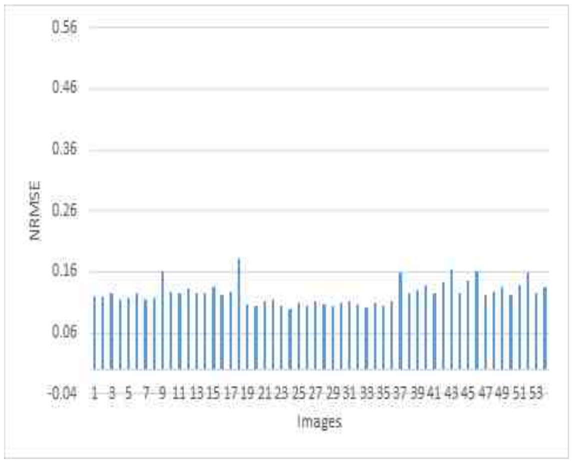
Fig. 4.57.: Second worst performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map



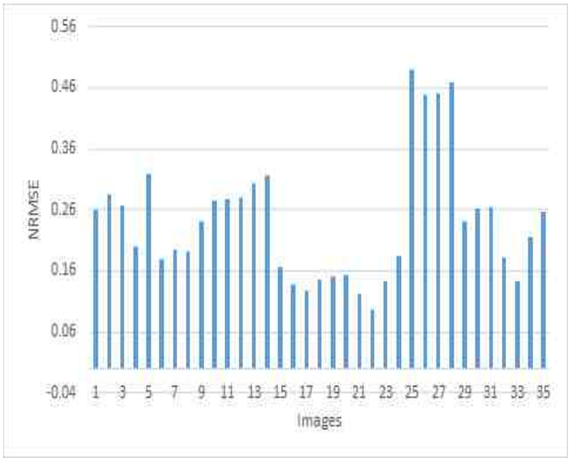
(a) NMAE performance of DfD-net trained and tested with Middlebury images

(b) NMAE performance of DfD-net trained with Middlebury and tested with Maya images

Fig. 4.58.: NMAE performance of DfD-net trained with Middlebury images

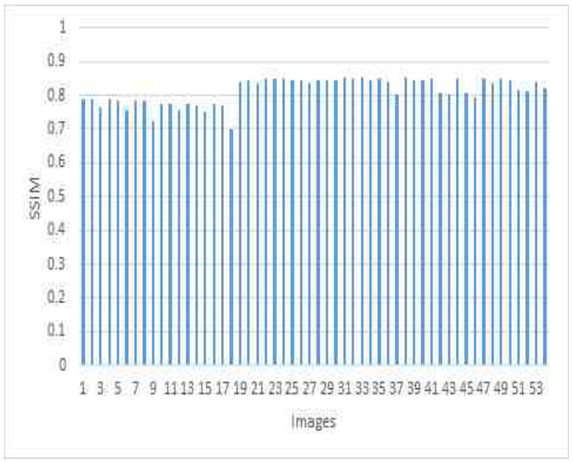


(a) NRMSE performance of DfD-net trained and tested with Middlebury images

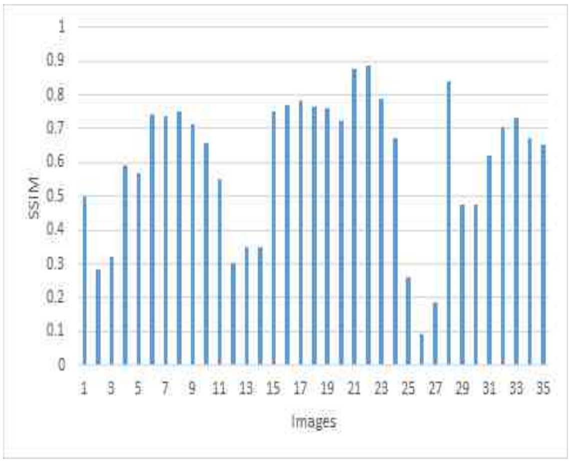


(b) NRMSE performance of DfD-net trained with Middlebury and tested with Maya images

Fig. 4.59.: NRMSE performance of DfD-net trained with Middlebury images



(a) SSIM performance of DfD-net trained and tested with Middlebury images



(b) SSIM performance of DfD-net trained with Middlebury and tested with Maya images

Fig. 4.60.: SSIM performance of DfD-net trained with Middlebury images

For the Middlebury dataset, the average NMAE, NRMSE, and SSIM are 0.102209, 0.122149882, and 0.815594547, respectively. The standard deviations of those metrics are 0.014531822, 0.016064662, and 0.035549268.

Among the generated depth maps, the best and worst ones are presented in Figures 4.61-4.68.

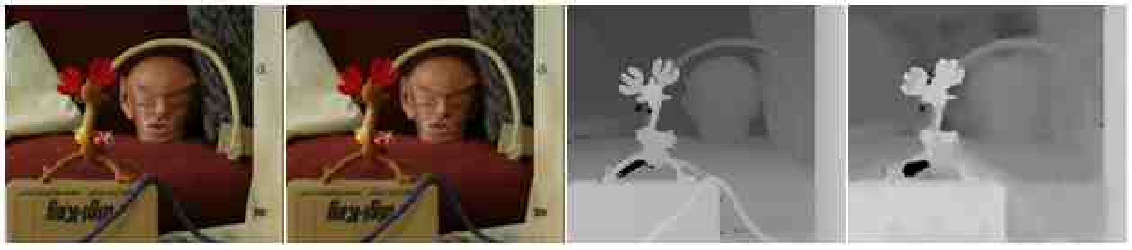


Fig. 4.61.: Best performance in the Middlebury dataset. In focus image, out of focus image, ground truth, and generated depth map

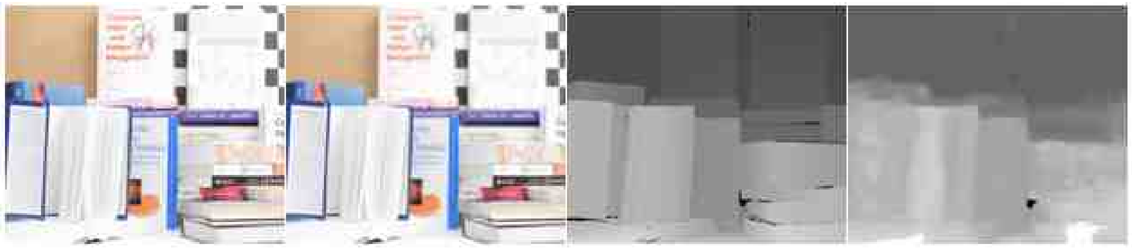


Fig. 4.62.: Second best performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map

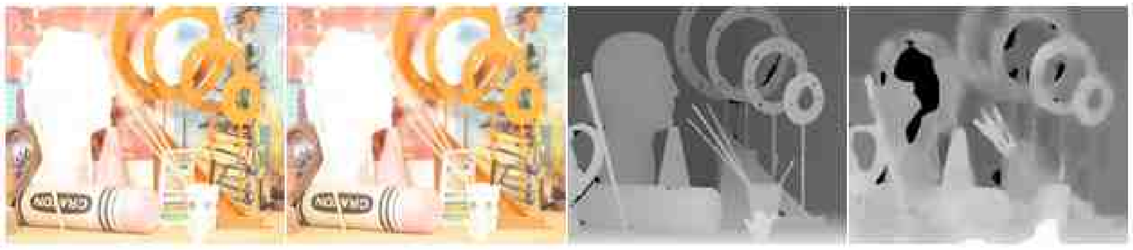


Fig. 4.63.: Worst performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map

The results from Section 4.4.1 and Section 4.4.2 are summarized in the Table 4.5 and Table 4.6.

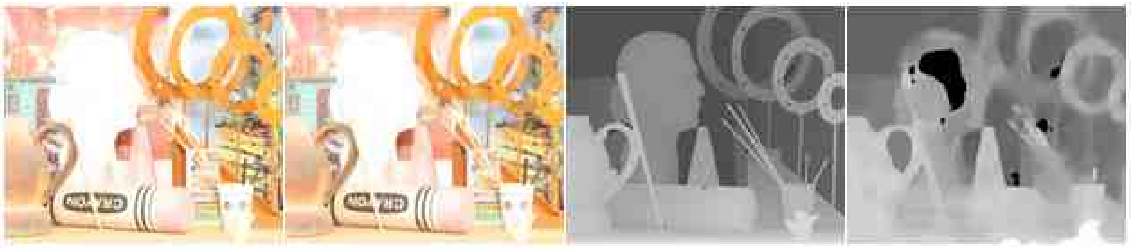


Fig. 4.64.: Second worst performance in the Middlebury dataset. In-focus image, out of focus image, ground truth, and generated depth map

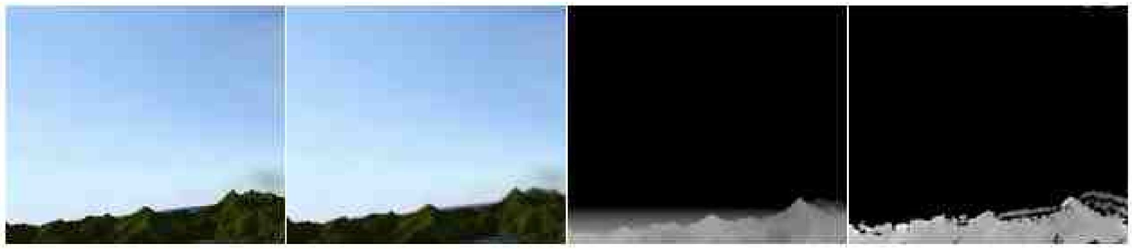


Fig. 4.65.: Best performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

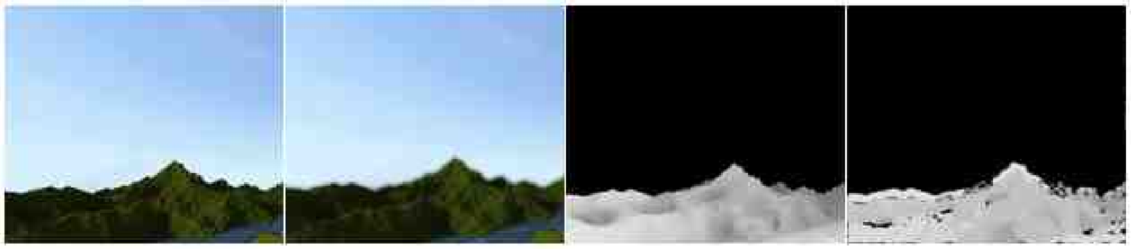


Fig. 4.66.: Second best performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

From the results of this section, it is evident that Maya trained network performs well for both datasets. On contrary, the Middlebury trained network does well only for Middlebury images. Besides, it is obvious by comparing the results of this section with the previous one that synthetic blur produces overall better results compared to Photoshop lens blur.

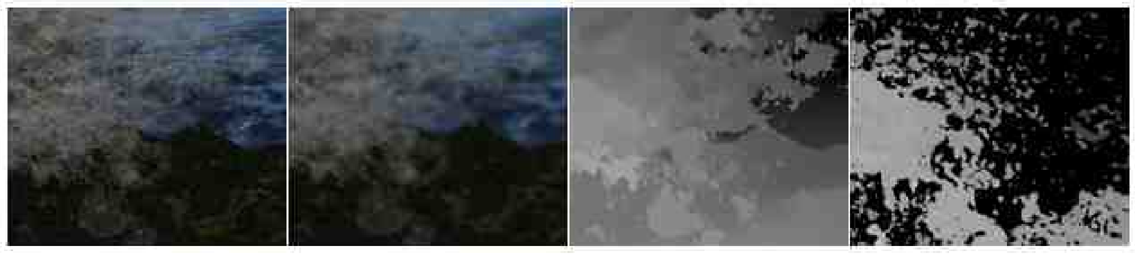


Fig. 4.67.: Worst performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

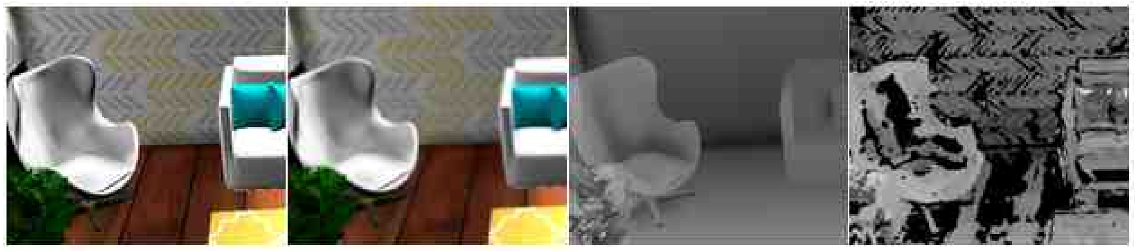


Fig. 4.68.: Second worst performance in the Maya dataset. In-focus image, out of focus image, ground truth, and generated depth map

Table 4.5.: Maya trained network performance

Performance Metrics (Average)	Maya	Middlebury
NMAE	0.02996109±0.0021893	0.1007744±0.0002259
NRMSE	0.06761012±0.0031968	0.1211227±0.0003216
SSIM	0.87718677±0.00135923	0.81559455±0.0014667

4.5 Performance Comparison between Graph Cuts and DfD-net

In Table 4.7, the comparison between graph-cuts and DfD-net is presented. The Maya trained network has been chosen as representative of DfD-net since it is the one that performed the best.

Table 4.6.: Middlebury trained network performance

Performance Metrics (Average)	Maya	Middlebury
NMAE	0.1450885±0.0183041	0.102209±0.00022586
NRMSE	0.2122622±0.01643671	0.1221499±0.00032155
SSIM	0.6879274±0.04303119	0.8134±0.0014667

Table 4.7.: Performance Comparison between Graph Cuts and DfD-net on Maya Trained, Maya and Middlebury Test Network

Performance Metrics (Average)	DfD Net		Graph-cuts	
	Maya	Middlebury	Maya	Middlebury
NMAE	**0.016±0.0007**	0.0255±0.0001	0.024±0.00014	**0.025±0.0001**
NRMSE	0.0447±0.001	0.0649±0.0004	**0.038±0.0002**	**0.047±0.0002**
SSIM	**0.9360±.001**	0.8880±0.0014	0.9195±0.0273	**0.958±0.0005**
Depth map generation time	0.5517±0.0003 second	0.5435±0.0003 second	5.14±3.7 minutes	2±0.36 minutes

Although graph cuts outperforms DfD-net to a small degree, it requires the matching the maximum blur level of synthetic blur to yield such high-quality performance. On the contrary, DfD-net does not have any such requirement. However, the bigger advantage of DfD-net over graph cuts is, it takes only half a minute on average to generate a depth map; whereas graph-cuts takes nearly two minutes on average for Middlebury images and more than five minutes for Maya images to generate a depth map.

5. CONCLUSION

5.1 Summary

In this research, we have compared graph cuts and convolutional neural network (DfD-net) to estimate depth from defocus blur using a natural (Middlebury) and a virtual (Maya) dataset. The performance of graph cuts is 4% better for Middlebury compared to Maya in terms of SSIM. However, with regard to NMAE and NRMSE, its performance is similar for both datasets.

The DfD-net has been trained using the natural and the virtual dataset and then combining both datasets. The network trained by the virtual dataset performed best for both datasets.

We have compared performance of graph-cuts and DfD-net. Graph-Cuts performance is 7% better than DfD-Net in terms of SSIM for Middlebury images. For Maya images, DfD-Net outperforms Graph-Cuts by 2%. With regard to NRMSE, Graph-Cuts and DfD-net shows similar performance for Maya images. For Middlebury images, Graph-cuts is 1.8% better. The algorithms show no difference in performance in terms of NMAE. The time DfD-net takes to generate depth maps compared to graph cuts is 500 times less for Maya images and 200 times less for Middlebury images.

5.2 Future Works

Even though we have achieved good results, there is room for improvements in our research. It is important to know how the network trained in virtual environments performs in the real world. Therefore, the network trained using virtual images will be tested with real data. We will be experimenting with different architectures of the

CNN to improve the result. We found that the blur is important to performance of DfD and our future work will be modelling the camera blur to improve the results.

The Middlebury images have been divided into different illumination and exposure levels. The DfD-Net has been proven to be robust to all levels of exposure and illumination. However, we have not experimented with exposure and lighting in the virtual environments. While we expect the DfD-Net to be indifferent to those parameters in virtual world as well, we need to explore the area in order to be sure.

REFERENCES

REFERENCES

[1] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller, and Y. LeCun, "Learning long-range vision for autonomous off-road driving," *Journal of Field Robotics*, vol. 26, no. 2, pp. 120–144, 2009.

[2] J. Michels, A. Saxena, and A. Y. Ng, "High speed obstacle avoidance using monocular vision and reinforcement learning," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 593–600.

[3] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *European Conference on Computer Vision*. Springer, 2012, pp. 746–760.

[4] C. Armbrüster, M. Wolter, T. Kuhlen, W. Spijkers, and B. Fimm, "Depth perception in virtual reality: distance estimations in peri-and extrapersonal space," *Cyberpsychology & Behavior*, vol. 11, no. 1, pp. 9–15, 2008.

[5] C. Yuan, M. Liao, X. Hu, and P. Mordohai, "18.2: Depth sensing and augmented reality technologies for mobile 3D platforms," in *SID Symposium Digest of Technical Papers*, vol. 43, no. 1. Wiley Online Library, 2012, pp. 233–236.

[6] W. Lee, N. Park, and W. Woo, "Depth-assisted real-time 3D object detection for augmented reality," *ICAT11*, vol. 2, pp. 126–132, 2011.

[7] Z. Ren, J. Meng, and J. Yuan, "Depth camera based hand gesture recognition and its applications in human-computer-interaction," in *2011 8th International Conference on Information, Communications and Signal Processing (ICICS)*. IEEE, 2011, pp. 1–5.

[8] X. Ren, L. Bo, and D. Fox, "Rgb-(d) scene labeling: Features and algorithms," in *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012, pp. 2759–2766.

[9] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-time dense monocular slam with learned depth prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2017.

[10] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2650–2658.

[11] R. Gupta, A. Y.-S. Chia, and D. Rajan, "Human activities recognition using depth images," in *Proceedings of the 21st ACM international conference on Multimedia*. ACM, 2013, pp. 283–292.

[12] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon, “The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, 2012, pp. 103–110.

[13] E. Delage, H. Lee, and A. Y. Ng, “A dynamic Bayesian network model for autonomous 3D reconstruction from a single indoor image,” in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2. IEEE, 2006, pp. 2418–2428.

[14] V. Hedau, D. Hoiem, and D. Forsyth, “Recovering the spatial layout of cluttered rooms,” in *2009 IEEE 12th international conference on Computer vision.* IEEE, 2009, pp. 1849–1856.

[15] D. Scharstein and R. Szeliski, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” *International journal of computer vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[16] F. H. Sinz, J. Q. Candela, G. H. Bakır, C. E. Rasmussen, and M. O. Franz, “Learning depth from stereo,” in *Joint Pattern Recognition Symposium.* Springer, 2004, pp. 245–252.

[17] Y. Furukawa, C. Hernández *et al.*, “Multi-view stereo: A tutorial,” *Foundations and Trends® in Computer Graphics and Vision*, vol. 9, no. 1-2, pp. 1–148, 2015.

[18] R. Ranftl, V. Vineet, Q. Chen, and V. Koltun, “Dense monocular depth estimation in complex dynamic scenes,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4058–4066.

[19] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah, “Shape-from-shading: a survey,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 21, no. 8, pp. 690–706, 1999.

[20] A. Criminisi, I. Reid, and A. Zisserman, “Single view metrology,” *International Journal of Computer Vision*, vol. 40, no. 2, pp. 123–148, 2000.

[21] M. Liu, M. Salzmann, and X. He, “Discrete-continuous depth estimation from a single image,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 716–723.

[22] P. Favaro, S. Soatto, M. Burger, and S. J. Osher, “Shape from defocus via diffusion,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 3, pp. 518–531, 2008.

[23] P. Favaro, “Recovering thin structures via nonlocal-means regularization with application to depth from defocus,” in *2010 IEEE Conference on Computer vision and pattern recognition (CVPR).* IEEE, 2010, pp. 1133–1140.

[24] M. Subbarao and G. Surya, “Depth from defocus: a spatial domain approach,” *International Journal of Computer Vision*, vol. 13, no. 3, pp. 271–294, 1994.

[25] M. Watanabe and S. K. Nayar, “Rational filters for passive depth from defocus,” *International Journal of Computer Vision*, vol. 27, no. 3, pp. 203–225, 1998.

[26] M. Watanabe, S. K. Nayar, and M. N. Noguchi, “Real-time computation of depth from defocus,” in *Three-Dimensional and Unconventional Imaging for Industrial Inspection and Metrology*, vol. 2599. International Society for Optics and Photonics, 1996, pp. 14–26.

[27] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, P. Hanrahan *et al.*, “Light field photography with a hand-held plenoptic camera,” *Computer Science Technical Report CSTR*, vol. 2, no. 11, pp. 1–11, 2005.

[28] A. Levin, R. Fergus, F. Durand, and W. T. Freeman, “Image and depth from a conventional camera with a coded aperture,” *ACM transactions on graphics (TOG)*, vol. 26, no. 3, p. 70, 2007.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[30] F. Liu, C. Shen, G. Lin, and I. D. Reid, “Learning depth from single monocular images using deep convolutional neural fields.” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 2024–2039, 2016.

[31] F. Mahmood and N. J. Durr, “Deep learning and conditional random fields-based depth estimation and topographical reconstruction from conventional endoscopy,” *Medical Image Analysis*, 2018.

[32] B. Li, C. Shen, Y. Dai, A. Van Den Hengel, and M. He, “Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1119–1127.

[33] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. L. Yuille, “Towards unified depth and semantic prediction from a single image,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2800–2809.

[34] A. Grigorev, F. Jiang, S. Rho, W. J. Sori, S. Liu, and S. Sai, “Depth estimation from single monocular images using deep hybrid network,” *Multimedia Tools and Applications*, vol. 76, no. 18, pp. 18 585–18 604, 2017.

[35] A. Roy and S. Todorovic, “Monocular depth estimation using neural regression forest,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5506–5514.

[36] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, “A naturalistic open source movie for optical flow evaluation,” in *European Conference on Computer Vision.* Springer, 2012, pp. 611–625.

[37] “Sintel, the durian open movie project,” accessed: 2018-11-5. [Online]. Available: https://durian.blender.org/

[38] M. Stark, M. Goesele, and B. Schiele, “Back to the future: Learning shape models from 3D CAD data.” in *Bmvc*, vol. 2, no. 4. Citeseer, 2010, p. 5.

[39] A. Broggi, A. Fascioli, P. Grisleri, T. Graf, and M. Meinecke, “Model-based validation approaches and matching techniques for automotive vision based pedestrian detection,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops, 2005.* IEEE, 2005, pp. 1–1.

[40] J. Marin, D. Vázquez, D. Gerónimo, and A. M. López, “Learning appearance in virtual scenarios for pedestrian detection,” in *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, 2010, pp. 137–144.

[41] D. Vázquez, A. M. López, D. Ponsa, and J. Marín, “Virtual worlds and active learning for human detection,” in *Proceedings of the 13th international conference on multimodal interfaces.* ACM, 2011, pp. 393–400.

[42] D. Vázquez, A. M. López, and D. Ponsa, “Unsupervised domain adaptation of virtual and real worlds for pedestrian detection,” in *2012 21st International Conference on Pattern Recognition (ICPR).* IEEE, 2012, pp. 3492–3495.

[43] H. Hattori, V. Naresh Boddeti, K. M. Kitani, and T. Kanade, “Learning scene-specific pedestrian detectors without real data,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3819–3827.

[44] G. R. Taylor, A. J. Chosak, and P. C. Brewer, “OVVV: Using virtual worlds to design and evaluate surveillance systems,” in *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on.* IEEE, 2007, pp. 1–8.

[45] B. Sun and K. Saenko, “From virtual to reality: Fast adaptation of virtual object detectors to real domains.” in *BMVC*, vol. 1, no. 2, 2014, p. 3.

[46] X. Peng, B. Sun, K. Ali, and K. Saenko, “Learning deep object detectors from 3D models,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1278–1286.

[47] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, “Virtual worlds as proxy for multi-object tracking analysis,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4340–4349.

[48] C. Liu, “Three dimentional moving pictured with a single imager and microfluidic lens,” Ph.D. dissertation, Purdue University, 2016.

[49] D. Emerson, “3-D scene reconstruction for passive ranging using deep learning,” Ph.D. prelim, Purdue University, 2018.

[50] D. Scharstein and C. Pal, “Learning conditional random fields for stereo,” in *2007 IEEE Conference on Computer Vision and Pattern Recognition CVPR’07.* IEEE, 2007, pp. 1–8.

[51] H. Hirschmuller and D. Scharstein, “Evaluation of cost functions for stereo matching,” in *2007 IEEE Conference on Computer Vision and Pattern Recognition CVPR’07.* IEEE, 2007, pp. 1–8.

[52] T. Chai and R. R. Draxler, “Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature,” *Geoscientific model development*, vol. 7, no. 3, pp. 1247–1250, 2014.

[53] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[54] Y. Boykov and V. Kolmogorov, “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[55] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[56] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.