# Realtime Single Image Depth Estimation

Yiheng Zhang
yihengz
yihengz@stanford.edu

## Abstract

*Differently from stereo depth estimation, single image depth estimation is always an ill posed problem due to ambiguities. In recent years, convolutional neural networks (CNN) have illustrated dominating capability in solving some ill posed problem, including single image depth estimation. In this project, I propose a U-Net based network structure to solve the single image depth estimation which runs at a real time inference speed. Both quantitative results and qualitative results show that my approach not only surpasses the baseline method in accuracy but also largely improved the computational efficiency.*

## 1. Introduction

Apart from the RGB color information, depth is one of the most important component for scene understanding and geometric relationship interpretation. It provides additional inter-object position information as well as distance information relative to the camera, which has been proved to be useful in a lot of areas, such as, 3d reconstruction [21], recognition and robotics [15]. Also, with the rapid development of self-driving vehicles, depth estimation via RGB cameras has great potential in providing assistance to Lidars, or even serves as a possible replacement with significantly lower cost.

While estimating depth based on stereo images is a far better studied topic traditionally, single image depth estimation meets the actual circumstances in real life as images are mostly taken not with a pair of cameras but rather casually one camera. In stereo cases, depth estimation can be reduced to image point correspondence searching problem, and the depth can be recovered deterministically [7]. In contrast, singe image depth estimation has intrinsic ambiguity thus requires leveraging global information and even semantic features. Fortunately, with the development of deep neural networks, people now have more tools to tackle vision tasks with features learnt and embedded by the neural networks. Line angles, perspective, object sizes, parallelism are all critical to approximate an accurate depth map. Semantic features such as object class are also proven to be useful [12].

Figure 1. An example of RGB image $I_c$ and depth image $I_d$ pair

This project aims to develop a deep neural network to perform single image depth estimation in an offline scenario at the speed of realtime inference, which is a fundamental step for realtime video depth estimation. Potential benefited fields including self-driving technologies, robotics and medical surgery. In this report, I will first formally define the single image depth estimation problem, then discuss a baseline method used as well as my refined approach. A combined loss function is adopted to focusing the training of the network in different aspects. Following the technical approach, both the quantitative results and qualitative results will be provided and analyzed to demonstrate my approach's improvements over the baseline in terms of both accuracy and inference speed. Lastly, a brief conclusion with current limitations and future work will close this report.

## 2. Related Work

A large amount of learning based approaches of single image depth estimation has been proposed recently. In this section, I will discuss three key relevant learning based areas in depth estimation, comparing their difference in applicable scenarios, technical focuses and approaches.

### 2.1. Supervised Stereo Depth Estimation

Typically, stereo depth estimation methods involve a pair of rectified images and a predefined method to calculate per pixel similarity and correspondence. Then the depth is in-

ferred from the disparity between correspondent pixels as the depth is a scaled inverse of the disparity. The predefined method is usually a search based method in traditional 3d vision. In contrast, learning based methods use CNNs to predict the matching function, treating the matching problem as a supervised learning problem. The training process requires a ground truth label of either the correspondence or the depth image. Numerous methods [24, 11] have proven that learning based matching methods' performance is superior to traditional hand crafted features.

Another approach in supervised stereo depth estimation is to directly computes the correspondence map or the disparity map between two images [5]. Similarly, this can be formulated as a regression problem with the help of existed correspondence label or depth label. This type of methods works in an end-to-end style reducing the steps to generating the final depth map.

Large amounts of accurately labelled ground truth disparity data and stereo image pairs are necessary for both of these methods. However, this type of data can be too difficult or too expensive to obtain in some real world scenarios. A work this problem is to use synthetic data to train the network. As computer graphics technology develops, synthetic data is becoming realistic and has been proven to work well in training neural networks [3].

### 2.2. Supervised Single Image Depth Estimation

Different from stereo depth estimation, single image depth estimation indicates that only a single image is available for depth prediction at test time. Local patch based method is a major approach to this problem. It generally involves dividing images into local patches and estimates the 3D location and orientation of planes in the patches [22]. However, this type of methods suffer from several disadvantages. One is sometimes they cannot reconstruct 3d information from some complex surfaces the other is the depth predictions are made locally thus lack global information and structural continuity.

Instead of using patch based method, Eigen et al. [4] showed that CNNs can predict dense pixel-level depth estimation directly. A two scale deep network was adopted and no hand crafted features were used. Several works also used techniques such as CRFs to reformulate the depth estimation problem from a regression problem to a classification in order to improve accuracy [2]. Some end-to-end networks were also proposed together with some combined loss functions to improve the prediction accuracy [13]. AdaBins further incorporates transformer blocks from natural language processing to adaptively estimate center depth value per image [1]. Works like [14] focuses on the inference speed of the network as real time depth estimation is critical to scenarios like self-driving and medical surgeries. In a word, supervised direct depth prediction methods generally outperforms patch based methods due to its capacity of leveraging global information as well as local information.

Again, supervised single image depth estimation requires large amount of labeled depth image to perform the training process, which still faces challenges in the data collection part.

### 2.3. Unsupervised Depth Estimation

Unsupervised Depth Estimation requires no ground truth during training. A common approach to this is leveraging view synthesis method. Given a pair of stereo image, using the left view to generate the corresponding right view and compare it against the real right view. With a chosen image reconstruction loss, the problem can be reduced to a regression problem. Based on the predicted right view, the disparity map can be calculated thus the depth can be inferred. Networks like The Deep3D network [23] has proved the feasibility of this approach. Godard et al. [6] explored more complex loss function and epipolar geometry constraints in training and achieved significantly better results.

However, unsupervised depth estimation can suffer from occlusion due to the lack of ground truth depth label. This requires additional work in occlusion reasoning, which increases the task difficulty and the network's inference time. Specular and transparent materials also pose corner cases in left and right consistency checking, which limits the generalization ability of the network. Last but not least, rectified and temporally aligned stereo pairs are necessary for unsupervised depth estimation.

I chose to tackle the single image depth estimation problem with a supervised approach, focusing on maintaining acceptable accuracy and visual quality of the prediction as well as increasing the inference speed to achieve real time computational efficiency.

## 3. Technical Approach

To practically solve this problem, I will first formulate the single image depth estimation as a regression task for CNN. Two critical elements are required for a CNN regression problem, the network structure and the loss function. They will both be discussed in detail following the problem statement part.

### 3.1. Problem Statement

The problem of depth estimation from a single RGB images is an ill-posed problem. From the class we know structure ambiguities will occur and even certain types of materials such as reflective materials will contribute to geometry ambiguities. Recently, the development of CNNs provides a regression-based method to tackle this problem.

The single image depth estimation problem can be formulated as an arbitrary RGB image $I_c$ passed into a function

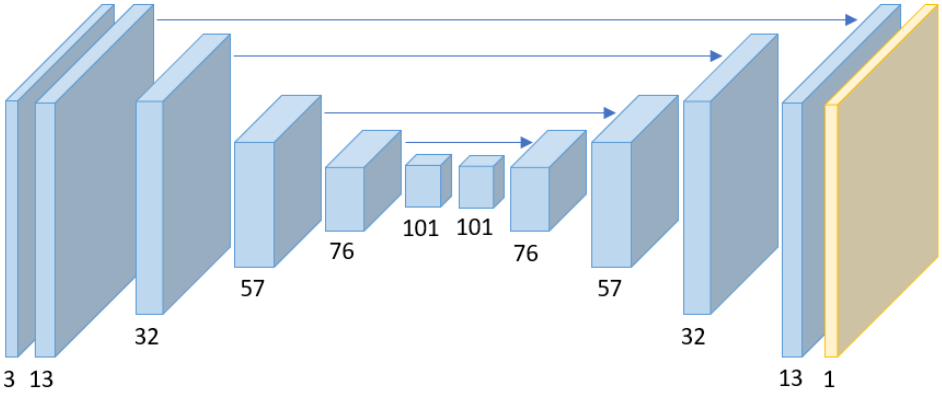


Figure 2. A feature map flow chart

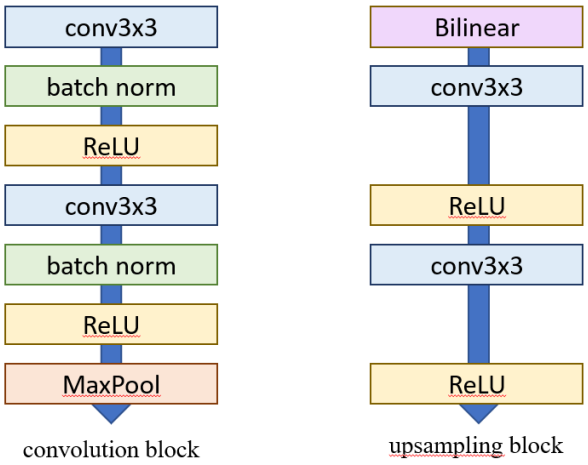


Figure 3. Network basic blocks

$f$ to calculate a corresponding depth image $I_d$. Since this project is based on deep learning techniques, a trained CNN $\mathcal{N}$ will be used as $f$ to produce the output depth image $I_d$

$$I_d = \mathcal{N}(I_c). \tag{1}$$

This simple equation formulates an end-to-end approach to perform the single image depth estimation with a CNN.

With a predefined loss function $L(I_x, I_y)$, we will be able to come up with a regression problem on the training set, where the target function is

$$\min L(\mathcal{N}(I_c), \hat{I_d}). \tag{2}$$

$\hat{I_d}$ denotes the ground truth label of $I_c$, which is a collected depth image.

### 3.2. Network Structure

My network structure is based on U-Net, a commonly adopted method in computer vision tasks, such as object detection, object selection and semantic segmentation. To be specific, my model consists of an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$, whose functions are defined as follows,

$$\phi(I) = \mathcal{E}(I) \tag{3}$$

$$I = \mathcal{D}(\phi(I)), \tag{4}$$

where $\phi(I)$ is the semantic feature of image $I$. Semantic feature of the original color image $\phi(I_c)$ will be extracted by the encoder $\mathcal{E}$ and the decoder $\mathcal{D}$ will reconstruct a depth image $I_d$ based on semantic feature $\phi(I_c)$. The overall network design is illustrated in Fig. 2.

For the encoder, the basic component is a convolutional block consists of two 3x3 convolutional layer each followed with a batch normalization [9] and a ReLU activation function for non-linearity [17]. At the end of each convolutional block, there is an additional maxpooling layer [16] to down sample the feature map as well as to extract regionally important semantic information, represented as the maximum value in each pooling window. An illustration of a convolutional block can be found at Fig. 3.

The encoder contains 4 convolutional blocks as well as two additional layers at the beginning to perform feature extraction. The input RGB image $I_c$ will be transformed into a feature map with 101 channels and will be $\frac{1}{16}$ the size in height $H$ and width $W$ of the original RGB image.

And for the decoder, I use bilinear upsampling instead of using transpose convolution. In my experiment, transpose convolution tends to introduce some checkerboard artifacts due to the overlap of transpose convolutional kernels, which severely downgrades the quality level of the output depth image since most of the surfaces of objects are continuous and I expect low frequency to be the majority component of the depth image, except for edges. The basic component of the decoder is a upsampling block composed with a single upsampling layer and two 3x3 convolutional layers and ReLUs. A upsampling block is illustrated in Fig. 3 as well. No batch normalization is involved in the decoder.

Similar to the encoder, the decoder consists of 4 upsampling blocks and one additional 3x3 convolutional layer at the end. Together they transform the feature map back to the size of $(1, H, W)$. The output should be a depth image so it only has a single channel.

Furthermore, skip connection is used to connect convolutional blocks and upsampling blocks with the same feature map size. Skip connections are proved to be effective in both gradient back propagation [8] and high frequency [19] reconstruction. Here I use the skip connections to help stabilize the training as well as to preserve the edges in the image harmed during maxpooling.

### 3.3. Loss Design

Our target is to ensure the estimated depth image $I_d$ is close to the ground truth $\hat{I}_d$. And I used three types of loss functions combined as the total loss.

The first one is the $\mathcal{L}_1$ loss. $L_1$ loss encourages the overall similarity with the ground truth, and thus will result in better performance in low frequency regions. In practice $\mathcal{L}_1$ always works as the basic supervision approach for image reconstruction problem.

The second is the structural loss $\mathcal{L}_s$, which is

$$\mathcal{L}_s = 1 - SSIM(I_d, \hat{I}_d) \tag{5}$$

, where SSIM denotes structural similarity index measure. Instead of directly measuring the difference between pixel values from the network output and the ground truth, SSIM represents a perceptual similarity level. This works as pixels tend to have inter-dependencies especially when they are spatially close. The structural loss $\mathcal{L}_s$ provides additional supervision for image perceptual similarities.

The last one is a high frequency loss $\mathcal{L}_e$ aimed at improving the prediction quality on high frequency regions such as edges and discontinued regions.

$$\mathcal{L}_e = (1 - e^{-g_x})|p - \hat{p}| + (1 - e^{-g_y})|p - \hat{p}| \tag{6}$$

, where $g_x$ and $g_y$ denotes the gradients in horizontal and vertical directions of the RGB image and $p$ and $\hat{p}$ denote the estimated depth value and the ground truth depth value of a pixel.

Together, I have the final combined loss $L$ as

$$\mathcal{L} = \frac{\alpha}{\alpha+\beta+\gamma}\mathcal{L}_1 + \frac{\beta}{\alpha+\beta+\gamma}\mathcal{L}_s + \frac{\gamma}{\alpha+\beta+\gamma}\mathcal{L}_e \tag{7}$$

where $\alpha, \beta, \gamma$ are positive hyperparameters to control the weights of different loss functions.

## 4. Experiment

### 4.1. Experiment Settings

There exists a wide variety of training and testing dataset for depth estimation, among which the most popular is the NYU-Depth-v2 dataset [18]. It consists densely labeled pairs of aligned RGB and depth images from 464 different indoor scenes. The training set has 50688 image pairs and the test set has 654 pairs. Furthermore, I split 1500 RGB and depth image pairs from the training set to compose the validation set. Examples of the NYU-Depth-v2 dataset can be found at Fig. 1

I deployed my CNN model on a desktop with Intel 9700k CPU, 32 GB memory and a NVIDIA Titan Pascal graphics card. The framework I chose for implementation is PyTorch 1.4.0 with Python 3.6. Adam optimizer was applied [10], the learning rate was set to $10^{-4}$, and the batch size was set to 8. For the loss function in Eq.11, $\alpha$ was set to 2 and $\gamma$ was set to 5. The large value of $\gamma$ is to compensate the smaller magnitude of $\mathcal{L}_e$, which I found is about $\frac{1}{10}$ of $\mathcal{L}_1$ and $\mathcal{L}_s$. Random horizontal flip and Random rotation up to 15 degrees were performed as the data augmentation methods. Also, the ground truth depth map label is normalized to $[0, 1]$ in order to reduce the training difficulty. The total training process takes around 72 hours to finish 50 epochs until the network converges. The training loss pattern can be found in Fig. 4.

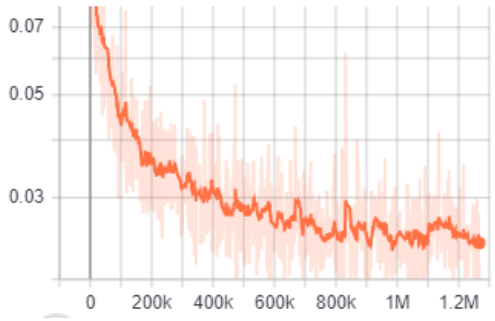

Figure 4. A training pattern of the combined loss function

### 4.2. Evaluation

For the evaluation, I picked the first U-Net [20] proposed in segmentation problem as the baseline method. The baseline model was trained from scratch with the same batch size, learning rate, data augmentation method for fair comparison.

There are three major aspects for evaluation, including quantitative metrics measuring the accuracy of the predicted depth map, inference time of a single RGB image input and qualitative visual results for direct comparison against the baseline and ground truth.

#### 4.2.1 Accuracy

As an image regression problem, the accuracy of the prediction compared with the ground truth labels are one of the most important evaluation steps.


Figure 5. Qualitative results of current experiments

Two types of metrics are adopted to evaluate the accuracy of our model. The first is mean square error (MSE). The MSE metric measures an average accuracy of the predicted depth map and tends to address more on the low frequency areas as they are the dominant parts of a depth map.

$$\text{MSE}(I_d, \hat{I}_d) = \frac{1}{n}\sum_{i}^{n}(p_i - \hat{p}_i)^2 \tag{8}$$

The MSE was an average MSE calculated from the test set.

The second one is the SSIM which measures the perceived quality and visual similarities of two images. The SSIM serves as an additional accuracy measurement to compensate MSE's disadvantages of poor correlation with human perception of visual system and incapability of distinguishing between different distortions in images. Similarly, the SSIM metric was a average SSIM calculated over the whole test set. All together, these two quantitative results are shown in Table. 1

Table 1. Quantitative accuracy comparison against the baseline

| Method | MSE | SSIM |
|---|---|---|
| Baseline | 0.04781 | 0.7549 |
| My method | 0.04588 | 0.7779 |

My method slightly outperforms the baseline in terms of both MSE (lower is better) and SSIM (higher is better). Even though my network is much shallower than the baseline network, it demonstrate its capacity in predicting the depth map is at least as strong as the baseline.

#### 4.2.2 Runtime Comparison

The runtime of the network is another key evaluation metric as one of my major goal is to make the network compact enough for realtime depth estimation.

First, I compare the parameter number of the baseline network and my network to evaluate the size of my network. In Table. 2 we can see that my network is $14\times$ smaller than the baseline U-Net.

Second, inference time is used as the speed measurement. Since the GPU cache miss has strong negative impact on inference time, GPU cache warm up is necessary for fair comparison. To test the inference time, I generate a random tensor in each iteration and feed it into the network for 100 times for the GPU cache to warm up, then use the average inference time of the next 10 iterations to measure the end to end time spent. Together I present the runtime results of the baseline and my network in Table. 2.

The quantitative results show that my method is able to run at approximately $6\times$ the speed of the baseline, which is 35.5 FPS. Either the realtime inference requirement is 24,

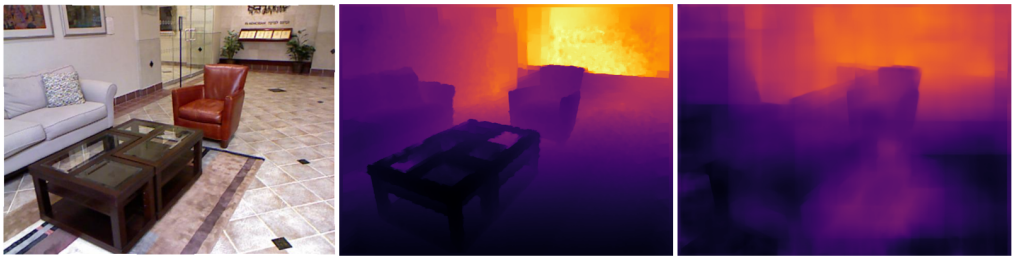

Figure 6. An example of failure cases in handling transparent materials

Table 2. Runtime comparison against the baseline

| Method | Inference Time (ms) | FPS | #Parameters |
|---|---|---|---|
| Baseline | 163.45 | 6.1 | 17.26M |
| My method | 28.11 | 35.5 | 1.22M |

25, or 30 FPS, my network's inference time now is decent and meets the real time computing requirement.

#### 4.2.3 Visual Results

In this subsection, I will present the qualitative results and analysis some improvements of my method's predictions over the baseline's. Some typical failure cases will also be discussed.

Example quantitative results can be found at Fig.5. Clearly, my method outperforms the baseline method again in terms of visual similarities with the ground truth depth maps. My method produces good continuity in low frequency areas of the predicted depth map, as well as clearer and more accurate boundaries of the objects, e.g the table in the first image, the bed in the second, the sofa in the third and the chairs in the last one.

However, in regions with larger amount of objects, my network tends to smooth things out. For example, in the third example of Fig. 5, there are stacked small objects on the book shelf, which are extremely challenging. The baseline generates some random patterns while my methods fails to distinguish between different objects.

Another failure case can be found at Fig. 6. The transparent glass creates reflections which are hard for networks to distinguish from general textures, thus my network produces incorrect predictions in those regions.

## 5. Conclusion

In conclusion, I successfully achieved my goal of real-time single image depth estimation and demonstrated that my proposed network significantly outperforms the baseline method. With $\frac{1}{14}$ parameters, my network's predictions have lower MSE, higher SSIM, runs $6\times$ faster than the baseline. Also, visual results from my network clearly have better perceived visual quality.

However, there are still some improvements can be done, such as the prediction accuracy for smaller objects and for objects with transparent or reflective materials. Also, it would make more sense to integrate my network into a video depth estimation pipeline to test its inference speed in practice.

I would like to thank CS231A teaching staff for all the help throughout the quarter and this project definitely helped me gain much knowledge in depth estimation.

## References

[1] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. *CoRR*, abs/2011.14141, 2020.

[2] Bo Li, Chunhua Shen, Yuchao Dai, A. van den Hengel, and Mingyi He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1119–1127, 2015.

[3] Zhe Cao, Hang Gao, Karttikeya Mangalam, Qizhi Cai, Minh Vo, and Jitendra Malik. Long-term human motion prediction with scene context. 2020.

[4] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *CoRR*, abs/1411.4734, 2014.

[5] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *CoRR*, abs/1504.06852, 2015.

[6] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. *CoRR*, abs/1609.03677, 2016.

[7] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456, 2015.

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] Lubor Ladicky, Christian Häne, and Marc Pollefeys. Learning the matching function. *CoRR*, abs/1502.00652, 2015.

[12] B. Liu, S. Gould, and D. Koller. Single image depth estimation from predicted semantic labels. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1253–1260, 2010.

[13] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *CoRR*, abs/1502.07411, 2015.

[14] Jun Liu, Qing Li, Rui Cao, Wenming Tang, and Guoping Qiu. Mininet: An extremely lightweight convolutional neural network for real-time unsupervised monocular depth estimation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 166:255–267, 2020.

[15] Jeff Michels, Ashutosh Saxena, and Andrew Y. Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 593–600, New York, NY, USA, 2005. Association for Computing Machinery.

[16] J. Nagi, F. Ducatelle, G. A. Di Caro, D. Cireşan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, and L. M. Gambardella. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 342–347, 2011.

[17] Vinod Nair and Geoffrey Hinton. Rectified linear units improve restricted boltzmann machines vinod nair. volume 27, pages 807–814, 06 2010.

[18] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. 2012.

[19] Yali Peng, Lu Zhang, Shigang Liu, Xiaojun Wu, Yu Zhang, and Xili Wang. Dilated residual networks with symmetric skip connection for image denoising. *Neurocomputing*, 345:67–76, 2019. Deep Learning for Intelligent Sensing, Decision-Making and Control.

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[21] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):824–840, 2009.

[22] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):824–840, 2009.

[23] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. *CoRR*, abs/1604.03650, 2016.

[24] Jure Zbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *CoRR*, abs/1510.05970, 2015.

## Supplementary

My code is accessible at github repo