Semi-Supervised Single Image Depth Estimation Using Deep Neural Network

by

Ali Jahani Amiri

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

There has been tremendous research progress in estimating the depth of a scene from a monocular camera image. Existing methods for single-image depth prediction are exclusively based on deep neural networks, and their training can be unsupervised using stereo image pairs, supervised using LiDAR point clouds, or semi-supervised using both stereo and LiDAR. In general, semi-supervised training is preferred as it does not suffer from the weaknesses of either supervised training, resulting from the difference in the camera's and the LiDAR's field of view, or unsupervised training, resulting from the poor depth accuracy that can be recovered from a stereo pair. In this thesis, we present our research in single-image depth prediction using semi-supervised training that outperforms the state-of-the-art. We achieve this through a loss function that explicitly exploits left-right consistency in a stereo reconstruction, which has not been adopted in previous semi-supervised training. Furthermore, we showed outputing inverse depth instead of disparity leads to better generalization and it is essential in the training. In addition, we describe the correct use of ground truth depth derived from LiDAR that can significantly reduce prediction error. The performance of our depth prediction model is evaluated on popular KITTI dataset, and the importance of each aspect of our semi-supervised training approach is demonstrated through experimental results. Our deep neural network model has been made publicly available.[1].

[1]Source code is available at https://github.com/a-jahani/semiDepth

Preface

This work is a collabrative work with Shing Yan Loo. I was mainly responsible for the model training and design of the loss while Shing Yan Loo was mainly responsible for LiDAR dataloader.

“Give me six hours to chop down a tree and I will spend the first four sharpening the axe.”

– Abraham Lincoln

Acknowledgements

I want to thank my supervisor, Prof. Hong Zhang. This work could not be done without his guidance. His advice was very constructive, and more importantly, he taught me how to approach a problem and do research. I would also like to thank Dr. Nilanjan Ray and Dr. Ron Kube for their great discussions and advice during my M.Sc.

Many thanks to my friends, colleagues, and those who were always open to help: Shing Yan, Moein, Martin, Kevin, Sean, Nazmus, Camilo, and Wei Nan.

Finally, I would like to thank my parents for their support. This accomplishment would not have been possible without their support.

Contents

List of Tables

List of Figures

Acronyms

SLAM Simultaneously Localization and Mapping

CNN Convolutional Neural Network

GAN Generative Adverserial Network

GPU Graphical Processing Unit

LiDAR Light Detection and Ranging

Sonar Sound Navigation and Ranging

Radar Radio Detection and Ranging

SfM Structure from Motion

DfD Depth from Defocus

SfS Shape from Shading

Abs Rel Absolute Relative Distance

RMSE Root Mean Square Error

Sq Rel Squared Relative Distance

S.o.t.A State of the Art

NLP Natural Language Processing

ReLU Rectified Linear Unit

IR Infrared Radiation

SSIM Structural Similarity

HVS Human Visual System

Chapter 1

Introduction

1.1 Motivation and Background

Perceiving the 3D world is an important task in the field of robotics and computer vision. To perceive the 3D world around us, knowing how far each pixel is from our robot, known as capturing depth, is essential. Acquiring depth is important in many situations. A solution to this task can be used in a broad range of applications. Some of these applications are 3D point tracking in localization of the robot poses [35], [63], 3D reconstruction in simultaneous localization and mapping [51], knowing the distances to each objects in collision avoidance [7], and figuring out the shape of the object and distance in grasping [44], finding out the scale [8] or figuring out occlusions [3] in artificial reality, and also taking a portrait mode in photo editing tools [56].

As a human, we have many cues to figure out depth of an object including having two eyes or moving our head to use parallax, or our depth of field when we try to focus on objects close and far away to us, or the prior our brain has learned about the size of objects and occlusions. In order for our robot to be able to perceive the 3D environment around it, knowing the depth is an essential task. In general, depending on the task, the accuracy needed varies. For example, for high definition maps in driverless car applications, the object locations have the accuracy of up to 10 cm. Depending on the environment and the tasks, many methods and sensors for capturing depth have been proposed and made. Each has its cons and pros. In this work, compared to our baseline, we improved the accuracy of the depth map from a single image by 12% (0.474

meter) and achieving 3.464 meters in RMSE metric for KITTI dataset. In the next section, we talk about these methods and sensors being used to capture depth.

1.2 Methods of Acquiring Depth

1.2.1 RGB-D cameras

One of the most common ways of acquiring depth using sensors is to use RGB-D cameras, e.g. Kinect, Xtion Pro. These cameras have Infrared Radiation(IR) projector, which projects unique known patterns of IR dots in a known direction, and there is another camera placed by known baseline capturing these patterns from the scene. By matching the pattern seen by the camera with the projected patterns, it can look up and see which direction this pattern was projected and by the use of trigonometry, depth can be calculated. Although these sensors work well, they have some limitations. Because of IR projection and sun rays having IR in its bandwidth, they can only work indoors. Kinect v2 tried to overcome this limitation by using time of flight cameras and built-in ambient light rejection feature, but still, it is not working well in the direct sunlight situations. The depth range for Kinect and Xtion pro is around 0.4m - 4.5m, 0.8m - 3.5m, respectively.

1.2.2 LiDAR, Radar, and Sonar

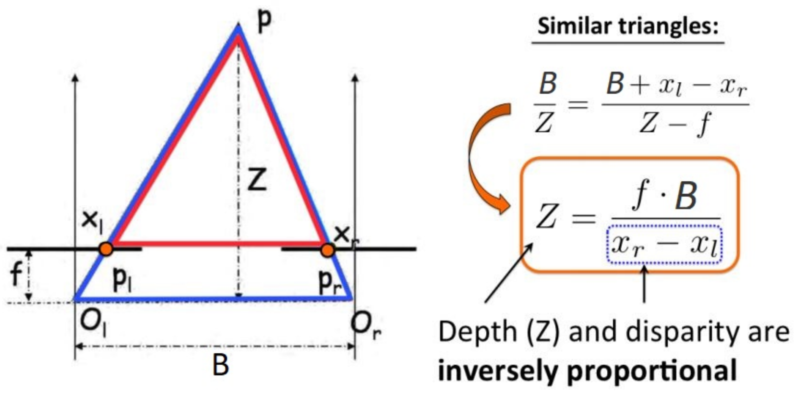
These sensors have been developed mostly to work for outdoor environments. These sensors project waves and measure the time that the wave takes to return to its source. Then the distance can be calculated using the following formula:

$$distance = \frac{speed\ of\ wave \times time\ of\ travel}{2} \tag{1.1}$$

The difference between LiDAR, Radar, and Sonar is the type of wave that is being emitted. LiDAR is using a laser while Radar and Sonar are using radio wave and sound wave, respectively. Due to the frequency difference, each has its usage in different applications. Compared to others, LiDAR is more accurate, and hence, they are expensive. Initially, LiDARs were only one beam, but nowadays multi-beam LiDARs are available at the expense of higher price. Due to the higher frequency of the laser, LiDAR has problems in foggy environments. While Radar can penetrate fog better, its accuracy is limited. Sonar is being used mostly for underwater and medical applications.

1.2.3 Using Image

Another way of getting depth is to use *stereo cameras.* By using two cameras, first, we need to find the correspondence between two images, and by knowing the baseline and camera focal length, we can calculate the depth of that pixel. Figure 1.1 shows the trigonometry in stereo to get the depth of a pixel.



p
Z
x_l
x_r
f
p_l
p_r
O_l
O_r
B
Similar triangles:
$\frac{B}{Z} = \frac{B + x_l - x_r}{Z - f}$
$Z = \frac{f \cdot B}{x_r - x_l}$
Depth (Z) and disparity are **inversely proportional**

Figure 1.1: Stereo Trigonometry. f is focal length and disparity is $x_r - xl$. O_l and O_r are the center of the left and right camera respectively. P is a point in space. p_l and p_r are the projection of point P on the camera plane, and B is the known baseline [23].

In summary, in the stereo camera, depth could be derived using the formula below.

$$depth = \frac{baseline \times focal\ length}{disparity} \tag{1.2}$$

Alternatively, many algorithms have been proposed to calculate the depth of a pixel using *multiple images.* Stereo cameras fall under this category as well. Some well-known algorithms that calculate depths using multiple images are the depth from defocus (DfD), shape from shading (SfS), simultaneous localization and mapping (SLAM) also known as structure from motion (SfM).

By taking photos from the same point of view with different known light source directions, we can solve an optimization to calculate the shape of an object. These methods are called Shape from Shading.

Depth from focus/defocus also estimates the 3D surface of a scene from multiple images of that scene. These images are captured from the same point of view using different camera parameters by typically changing the focal length of the camera to change the image plane position. The sharpness of a pixel with a different focus of the camera can measure the depth of a pixel.

Simultaneous localization and mapping (SLAM) known as structure from motion (SfM) in the computer graphics field also measures the depth of a pixel using an optimization process. First corresponding point in multiple images needs to be established. By knowing how much the camera has moved between frames (camera poses), we can solve an optimization term called a geometric loss or photometric loss to find the 3D position of that pixel.

Another way of capturing depth is to use *single image.* Given a single image, we can perceive depth from it and find out which object is closer to which one. This is possible because our brain has learned sizes of different objects and how they should look like, and it also has learned the effect of occlusions.

By using learning methods, researchers [24], [46] have shown that single image depth estimation is possible, but the initial accuracy achieved was low. With the rise of deep learning, the accuracy was boosted significantly in the field of single image depth estimation. The focus of this thesis is about single image depth estimation using deep learning. Hence, in the next section, we explore deep learning.

1.3 Deep Learning

Deep learning is a part of machine learning, which involves a broad range of neural network architectures. It uses connected neuron nodes to simulate the process of the brain. One of the differences between the deep neural network versus classical machine learning is that there is no need to hand-design the features. Features are automatically being optimized in the process of deep neural network training. Some of the architectures of the neural network architectures are fully connected neural networks, recurrent neural networks, convolutional neural networks, and deep belief networks. These neural networks have been applied to many fields and proven to be successful with high accuracy, e.g. in computer vision, audio recognition, natural language processing, and medical image processing.

Some of the most common properties of these neural networks are:

- Multilayer graph-based architectures with multiple nodes (i.e. neurons) in each layer. Each neuron might be connected to other neurons (or input x_i) with weighted edges w_{ij}. The output of each neuron is the linear multiplication of the edges with its input followed by a nonlinear activation function.

- The cost $C = f(X, W, Y)$ for a given set of Y target, is a function of weighted edges W, input X.

- The cost function is minimized using nonlinear optimization frameworks, e.g. gradient descent. The weighted edges are being updated based on the back-propagation to have a lower cost of the objective given the input [55].

Deep neural networks can have millions of parameters which need to be optimized. Therefore, they have high flexibility of representing complex function f to map input X to target Y. In the following section, we discuss some of the specific deep neural network architectures.

1.3.1 Deep Neural Networks

Fully Connected Networks

Fully connected Neural networks are an architecture which has fully connected layers. In fully connected layers, each neuron has weighted connections from all previous neurons. Neural networks which are based on only fully connected layers are called fully connected neural networks.

For instance, Figure 1.2 shows a simple fully connected neural network with one fully connected layer. It has three layers:

- **input layer X_i:** it includes two neurons and acts as an input to the network.
- **fully connected layer:** a layer of four neurons, which are all connected to previous neurons and neurons after.
- **output layer y_i:** a layer consists of two neurons and acts as an output layer. For instance, one could be a probability of the input being cat and another might be a probability of the input being dog.

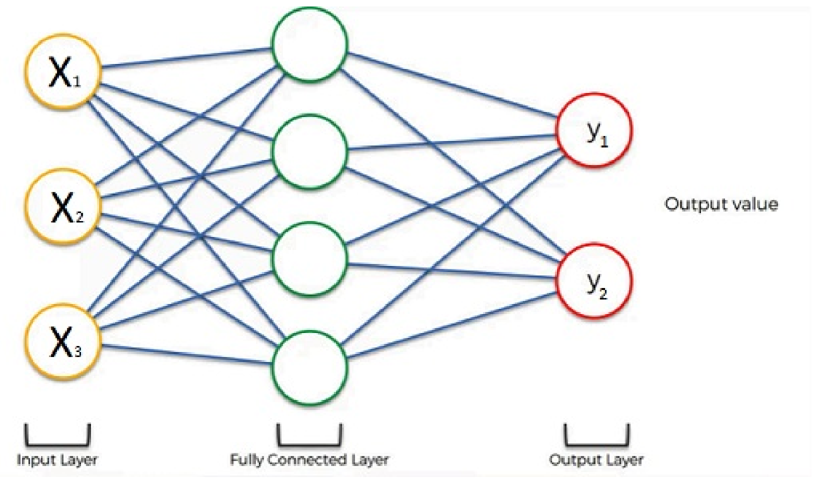


Figure 1.2: Sample of fully connected neural network. It includes three neurons X_i as input layer and a fully connected layer with four neurons and an output layer with two neurons y_i. Neurons are connected with blue lines (edges) where each has a weight of w_{ij} [52].

Convolutional Neural Networks

Convolutional neural network (CNN) is a type of deep neural network mostly applied to the image data. These networks include convolutional layers. These convolutional layers have filters which convolve the input with the filter using dot product followed by a nonlinear function known as activation layer.

Some of the layers and units which might be used in the CNNs are fully connected layers, convolutional layers, activation layers, and pooling layers. In the following section, we will discuss some of the components of CNNs.

Convolutional Layers: In mathematics, convolutions are integral over two functions to measure how much the two functions overlap when one passes the other one. Each convolutional layer has these two common attributes:

- Input shape is usually a tensor of shape $(batch\ size) \times (width) \times (height) \times (depth)$.
- Convolutional kernels known as filters have a size of $width \times\ height \times depth$. Their width and height are hyper-parameters, and the depth should match the depth of its input tensor. Each filter is convolved across the width and height of the input volume and computes the dot product between the elements of the filter and the input. The output is a 2-dimensional activation map. By having m filters, we will have m 2-dimensional activation maps. Therefore, by concatenation of all of them, the output of this layer is a tensor with a depth size of m, where m is the number of the filters.

Pooling Layers: Pooling layer is a nonlinear layer for downsampling. There are many strategies for pooling layer, and max-pooling is the most commonly used. The pooling layers are used to reduce the spatial size of the representation after successive convolutional layers. Max-pooling gets the maximum value in a window and outputs that number for that particular position of the window for the output. Pooling layers are applied individually for each slice of the image depth. Max pooling is usually defined by a window size of $n \times n$ with a stride of m where the stride is the number of pixels each time to jump over to find the next position of the window.

Figure 1.3 shows an output of an image slice when a max-pooling layer of 2×2 with stride 2 is done on that slice.

Activation Layers: Activation layers are usually nonlinear function, which decides whether a neuron should fire or not. Nonlinear activation functions make the model flexible to fit and generalize to a variety of data. Hence

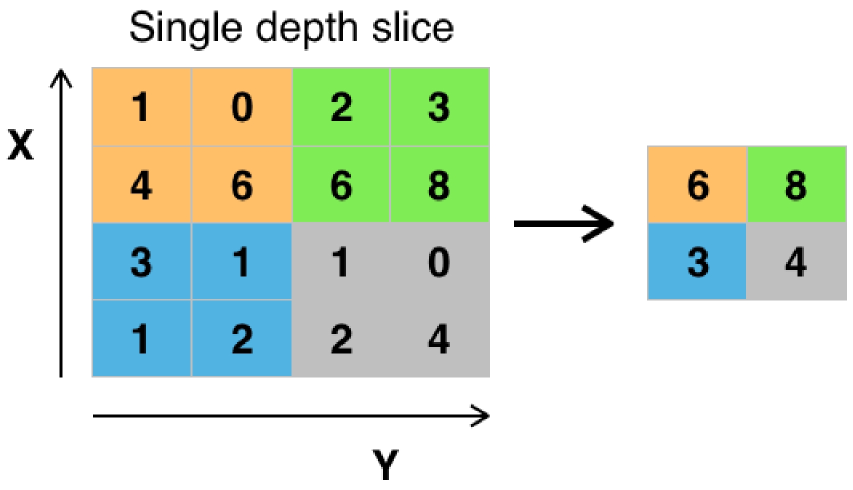


Figure 1.3: Max-pooling with a 2×2 filter and stride of 2 [60].

the model can represent complex mapping functions. Many nonlinear activation functions have been proposed, e.g. sigmoid, tanh, SELU, ReLU. Among all of the activation functions, and RELU and sigmoid are the most commonly used. Figure 1.4 shows the sigmoid versus ReLU function. Sigmoid is often used before the output, especially if the output is a probability making the output range fall into (0,1) while ReLU is often used after convolution layers.

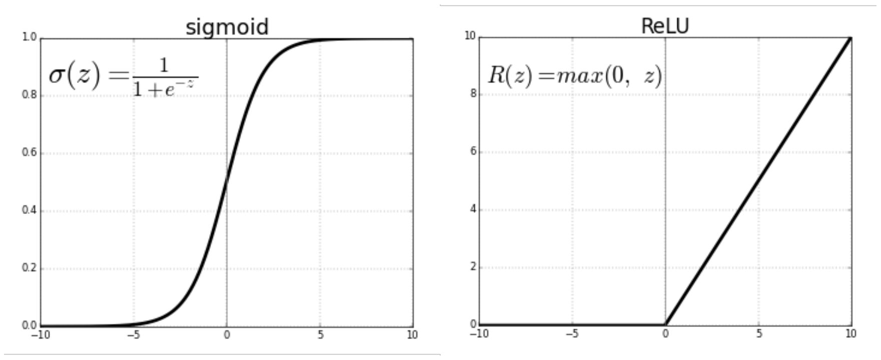


Figure 1.4: Sigmoid vs ReLU activation Layers [54].

Now that we have discussed the different units in convolutional neural networks, we bring an example of a CNN in figure 1.5. In this example, the input is an image of a digit between 0-9 and output is ten numbers each are the probability of the input being a digit from 0-9.

In the first layer, conv_1, n1 $5 \times 5 \times 1$ convolution filters are being applied to the input resulting in an output tensor with a shape of $24 \times 24 \times n1$ given the input dimensions are $28 \times 28 \times 1$. Then a 2×2 max-pooling process is applied, reducing the dimensions to half, making it $12 \times 12 \times n1$. Again n2 convolution filter with size of $5 \times 5 \times n1$ is applied resulting in an output of $8 \times 8 \times n2$ followed by a 2×2 max-pooling. The output will become $4 \times 4 \times n2$. Afterwards, by flattening the tensor, the shape of the tensor will become $n2 \times 16$. Subsequently, a fully connected layer of n3 units is applied, and at the end, an output layer of 10 neurons each represents a digit exists. The output of each neuron is a probability of the input image being that digit.

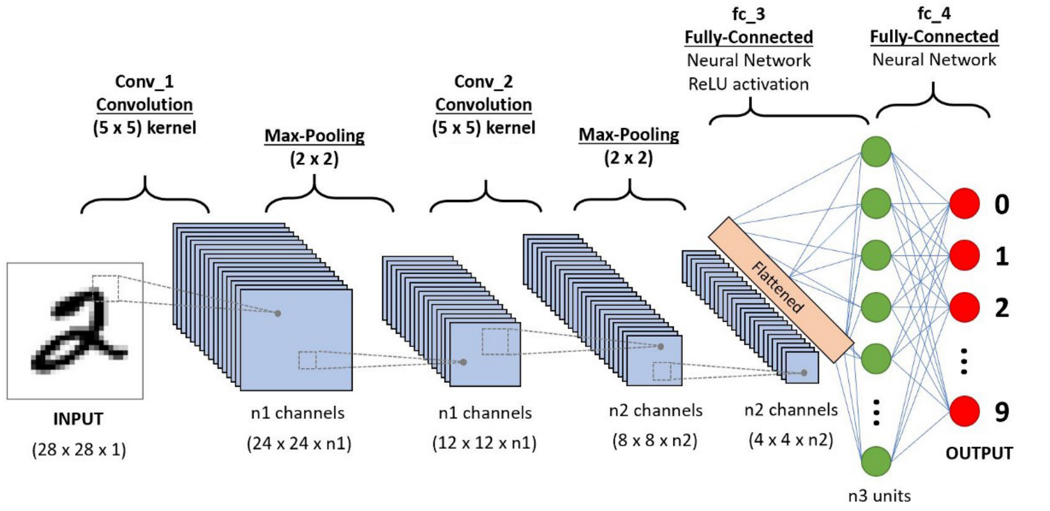


Figure 1.5: A sample of CNN [45].Input is an image of a digit between 0 to 9 and output is the probability of that image being 0-9 for each digit. $n1$, $n2$, $n3$ are the number of filters in layer one and two respectively. $n3$ is the number of neurons in fully connected layer [45]. This network includes two convolutions followed by a max-pooling, and at then end, fully connected layers to estimate the probability of that image belong to which class.

Generative Adversarial Networks

Generative Adversarial Networks (GANs) are known as a class of machine learning where two neural networks are contesting with each other. These two neural networks are called generator and discriminator. The generator tries to generate new data that falls in the training data distribution while discriminator fights back the generator by trying to discriminate between the generated one and the actual real data coming from the training distribution. With optimization, both networks will get stronger and reach a point where discriminator cannot distinguish between generated data and the actual real data anymore. Then in most cases, the discriminator is being removed, and only the generator is being used to generate new samples from training data.

Figure 1.6 shows a simple idea of GANs where generator generates an image from random noise while discriminator tells if its input is from fake images coming from the generator or it is from real images coming from the training set. By letting these two fight against each other eventually generator will fool the discriminator by generating real looking images, and that means the generator has learned the distribution of the training set properly.

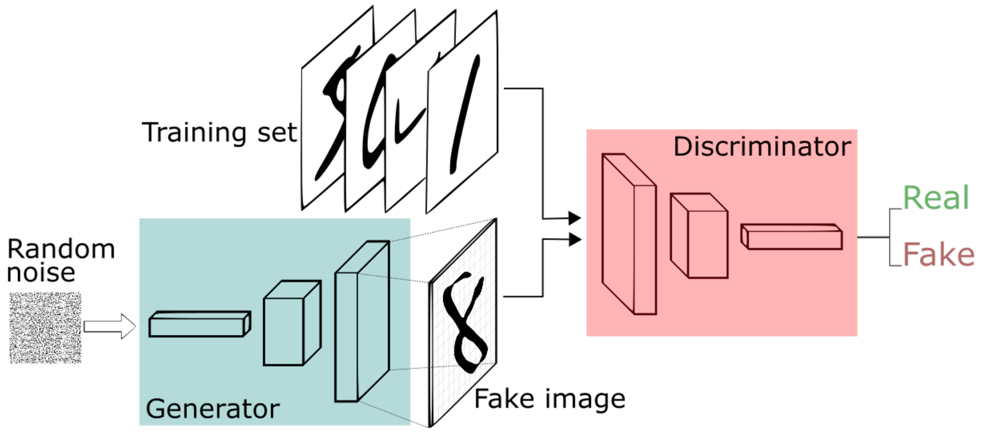


Figure 1.6: A sample of GAN [47]. The generator tries to generate a fake image in the distribution of the training set from random noise. Discriminator tries to fight back the generator by discriminating the fake image from the generator and the real image from the training set

1.3.2 Deep Learning in Depth Estimation

Deep learning has shown promising results in many classification and regression tasks in a variety of fields and is considered the state-of-the-art method in many fields. In recent years, many researchers are taking advantage of deep neural networks in their tasks, and depth estimation is not an exception. Some use it for an end to end depth learning, e.g. direct learning of depth from video [70], or stereo matching [67] while others are using it as a mini-module to improve the traditional multiple view geometry, e.g. feature detection and matching [18].

Besides, deep learning has shown its use in depth completion task, e.g. densifying of LiDAR data [43], filling missing data for RGB-d cameras using inpainting methods [68] as well.

One of the disadvantages of the deep learning models compared to traditional based methods is the extra cost of GPU and the time consumption during inference. Some researchers have proposed methods to reduce model size and complexity to run the single image depth estimation models in real-time for robots [13] beside the fact that costs of GPUs are dropping over time and their power of calculation is increasing with improvements in technology and hardware designs.

With the rise of deep learning, notable achievements in terms of accuracy and robustness have been obtained in the study of single image depth estimation. We can train deep neural networks for single image depth estimation using supervised, unsupervised, and semi-supervised methods.

Supervised methods in single image depth estimation use ground truth derived from LiDAR data. It is time-consuming and expensive to obtain dense ground-truth depth, especially for the outdoor scenes. LiDAR data is also sparse relative to the camera view, and it does not share the same field of view with the camera in general. Consequently, supervised methods are unable to produce meaningful depth estimation in the non-overlapping regions with the image (see Figure 1.7). In contrast, unsupervised methods learn dense depth prediction using the principle of reconstruction from stereo views; hence, depth can be estimated for the entire image. However, the accuracy of unsupervised depth estimation is limited by that of stereo reconstruction.

1.4 Thesis Statement

Unsupervised methods use either stereo cameras or video sequences for training. One of the limitations of the unsupervised single image depth estimation methods that use stereo cameras is that they output disparity and, as a result,

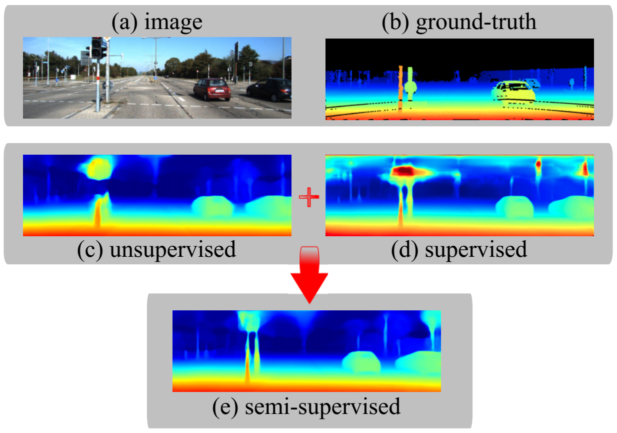


Figure 1.7: Qualitative comparison of supervised, unsupervised, and semi-supervised. Using stereo only (c) leads to the noisy depth map. Using LiDAR only (d) results in inaccurate for the top part of the image because there is no ground-truth available. Our semi-supervised method (e) fuses both LiDAR and Stereo and can predict depth more accurately. Ground truth LiDAR (b) has been interpolated for visualization purpose.

all the data in training should come from the same camera setup and baseline. If the cameras are different, then this will lead to different disparity values for the similar images taken by two different cameras. This inconsistency in output is similar to feeding same cat images taken from two different cameras to a classification network and one time saying this is a cat image and the second time saying this is a dog image. We show this inconsistency will confuse the network in the training phase.

As a solution to this problem, we propose our model to output inverse depth instead of disparity. Unlike disparity, inverse depth is invariant to camera intrinsic and baseline. In unsupervised training, in order to leverage massive data over the internet, it is essential that the output of the proposed model be invariant to the camera settings.

In this thesis, we present our research in single image depth prediction using semi-supervised training that outperforms the state-of-the-art. The focus of our study is the outdoor scene. We propose a novel semi-supervised loss function that uses the left-right consistency term originally proposed in [20]. Our network uses LiDAR data for supervised training and rectified stereo images for unsupervised training, and in the testing phase, our network takes only one image to perform depth estimation.

Another focus of our study is the impact of ground truth depth information on the training of our model, when network training is performed with the projected raw LiDAR data and the annotated depth map recently provided by KITTI [53], respectively. We discover that the commonly used projected raw LiDAR contains noisy artifacts due to the displacement between the LiDAR and the camera, leading to poor network performance. In contrast, we use the more reliable preprocessed annotated depth map for training, and we can achieve a significant reduction of prediction error.

In summary, in this thesis, we propose a semi-supervised deep neural network for depth estimation from a single image, with state-of-the-art performance.

1.5 Thesis Contribution

Our work makes the following three main contributions.

- We show outputting inverse depth is better than outputting disparity in unsupervised training of single image depth prediction, which makes the depth estimation model invariant to the camera setup.
- We show including a left-right consistency term in the loss function improves performance in semi-supervised single image prediction.
- We provide empirical evidence that training with the annotated ground truth derived from LiDAR leads to better depth prediction accuracy than with the raw LiDAR data as ground truth.

We make our semi-supervised deep neural network available to the community.

1.6 Organization of the Thesis

This thesis is organized as follows.

- **Chapter 1.** ***Introduction:***

  We discussed the problem of acquiring depth and our motivation, deep learning and its applications in depth estimation, thesis statement, and contribution in the field of single image depth estimation.

- **Chapter 2.** ***Related Works:***

  We reviewed related works to our research focusing on single image depth prediction for outdoor scenes.

- **Chapter 3.** ***Method:***

  We presented our deep neural network architecture, loss function used for training.

- **Chapter 4.** ***Datasets:***

  We talked about commonly used datasets for outdoor depth estimation and their properties.

- **Chapter 5.** ***Experiments:***

  We provided the most common evaluation metrics in depth estimation field and discussed the implementation details of our training procedure and the qualitative and quantitative result of our proposed method. We also discussed and experimented how each factor contributed to our improvement.

- **Chapter 6.** ***Conclusion:***

  We talked about the conclusion of this thesis and the ideas developed in this field. We also suggested some ideas for future work to make the result more accurate.

Chapter 2

Related Works

Over the past few years, numerous learning-based methods have been proposed for the problem of single image depth estimation. In this chapter, we survey papers related to single image depth estimation. Given a single image, we are interested in the depth value for every pixel. We can categorize the work of depth estimation into shallow methods and deep methods. Shallow methods are trying to find the best model that can map input image space to the output depth, whereas deep methods are doing the same thing except for the model which is a convolutional neural network instead.

2.1 Shallow Methods

One of the first methods in the single image depth estimation area was the work of D. Hoiem et al. [24] known as Automatic Photo Pop-up. It creates a 3D model from an image by labelling each region of an outdoor image as ground, vertical, or sky and based on the ground-vertical boundary in the image. Then by an estimate of the horizon's position, it figures out where to fold and cut the image.

Another early work in this field is Saxena et al. [46] known as Make3D. They apply a superpixel algorithm on the image and fit each region a plane and estimate the 3d position and orientation of planes corresponding to each region. Both methods, Automatic Photo Pop-up [24] and Make3D [46], are considered a local method and they rely on hand-crafted features. Therefore, they have poor generalization and accuracy compared to deep learning-based approaches.

2.2 Deep Methods

With the advances of deep learning, the field of single image depth estimation has been pushed to the next level of accuracy. Since then, almost all the proposed methods are using deep learning-based models, e.g., convolutional neural network. One advantage of using these models is that these models make coherent global predictions. Based on methods of the training, we can categorize deep methods into three subcategories: supervised, unsupervised, semi-supervised.

2.2.1 Supervised

Supervised methods use ground truth depth, usually from LiDAR in outdoor scenes, for training a network. Eigen et.al. [12] was one of the first who used such a method to train a convolutional neural network. Figure 2.1 shows the architecture of their proposed method. They had two stages of training. First, they generate the coarse prediction and then use another network to refine the coarse output to produce a more accurate depth map.

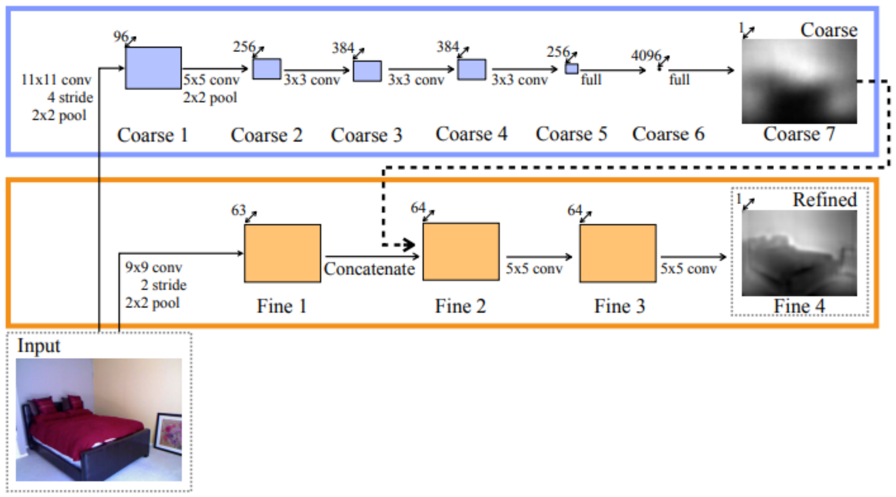


Figure 2.1: Eigen et al. model architecture [12]. It consists of two stage prediction, a coarse prediction followed by a refined prediction. *Full* and *conv* refers to fully connected layers and convolutional layer, respectively.

Following [12], several techniques have been proposed to improve the accuracy of convolutional neural networks such as CRFs [34], inverse Huber Loss as a more robust loss function [32], joint optimization of surface normal and depth in the loss function [25], [42], [58], fusion of multiple depths maps using Fourier transform [33], and formulation of depth estimation as a problem of classification [6], [14].

Lee et al. [33] proposed a method where they used multiple crops of the input image to produce multiple depth maps followed by fusion of multiple depth maps using Fourier transform. They also proposed a new loss, called depth-balanced Euclidean loss, where it solves the problem of the tendency of L2 loss to backpropagate more for higher depth values. Similarly, Xu et al. [62] fused multi-scale information of the CNN by using a structured attention guided network.

Among those, the work of Fu et al. [14] is considered state of the art in supervised methods. Figure 2.2 shows the architecture of Fu et al. [14]. They treated the regression problem as a classification task by discretizing the depth values into k intervals in log space, and for every pixel, they predict k probability of that pixel belong to every interval and then merge all of those probabilities and get the final class of that pixel to determine which interval that pixel belongs to. One of their contributions was to use dilated convolutions as well. Although they outperform other supervised and unsupervised methods by a huge margin, their result for the top portion of the image is noisy resulting from the difference in the camera's and the LiDAR's field of view. There is no groundtruth for the top portion of the images in KITTI. Therefore, noisy result won't affect the quantitative results.

Recently Generative Adversarial Networks have shown a promising result in domain transfer applications [28], [71]. Some researchers proposed supervised training of depth using GAN [5], [10], [22], [69] by distinguishing real or fake images, e.g., Zheng et al. [69] and Atapour et al. [5] proposed using synthetic data to train the network and convert real images to a synthetic domain and predict depth in the synthetic domain while some of them proposed unsupervised learning of depth using GAN [2], [4], [30], [41].

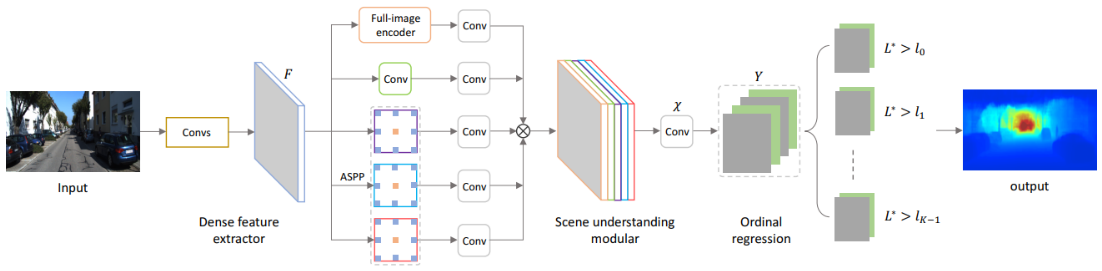


Figure 2.2: Fu et al. model architecture [14]. For each pixel x, they predict k probabilities of $L^* > l_k$ where L^* denoted the predicted depth for that pixel and l_k are the k discritized depths.

2.2.2 Unsupervised

Collecting manually labelled data is a time consuming and expensive task, especially for outdoor environments, where LiDAR ground truth data is sparse and noisy and requires expensive hardware. Uhrig et al. [53] proposed an automatic way of post-processing LiDAR data to remove noise and to make it a little bit denser but still collecting data is always time-consuming, and training will be limited to those datasets. In deep learning, the most important factor is the amount of data, and that is where the importance of unsupervised training shines. Recently, the unsupervised methods in depth estimation have shown a promising result.

Xie et al. [61] proposed unsupervised deep neural networks method of view synthesis to convert 2D videos and images to a stereoscopic 3D format, i.e., generate right eye image for left image input. Similarly, Garg et al. [15] demonstrated an unsupervised method in which the network is trained to minimize the stereo reconstruction loss; i.e., the loss is defined such that the reconstructed right image (i.e., obtained by warping the left image using the predicted disparity) matches the right image. Later on, Godard et al. [20] extended the idea by enforcing a left-right consistency that makes the left-view disparity map consistent with the right-view disparity map. The unsupervised training of our model is based on [20]. Figure 2.3 shows the input-output of the proposed Godard et al. method [20]. Given a left view as input, the model in [20] outputs two disparities of the left view and the right view, while we are outputting only one depth map for one input image in the form of inverse depth instead of disparity. As a result, we treat both left and right images equivalently. This allows us to eliminate the overhead of the post-processing step in [20]. By making these changes, our unsupervised model outperforms [20] as will be discussed in Section 5.3.

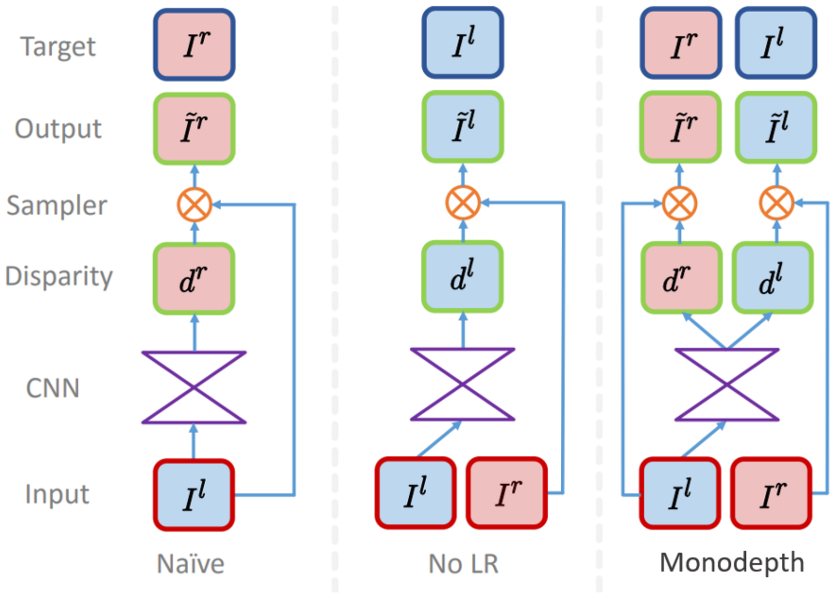


Figure 2.3: Inverse warping strategies [20]. The left column, Naïve, CNN outputs disparity aligned with image right for input image left. The middle column, NO LR, fixes this issue but there is no constraint on the left-right consistency [15]. The right column is the warping strategy proposed by Godard et al. [20]

Since visual odometry, optical flow, and depth are very related tasks, some researchers proposed methods of jointly optimizing them [19], [72]. At the same time as Godard et al., Zou et al. [70] trained a network using monocular videos. Figure 2.4 shows their proposed encoder-decoder architecture. There are two networks: one for depth prediction, and another for Pose estimation and explainability mask. Explainability mask is a mask to define the occluded and moving regions in monocular videos. They use the visual odometry between frames to warp the consecutive frames into each other and have their loss as the photometric loss. Then they calculate loss only on regions where explainability mask is valid, i.e., non-occluded and static. Accounting for occluded regions is a well-explored method in optical flow estimation [26], [27], [40], [59]. In the field of depth estimation, Gue et al. [21] proposed using a stereo matching network to find a proxy of depth and occlusion mask and used those outputs as a helper to train another network for single image depth estimation.

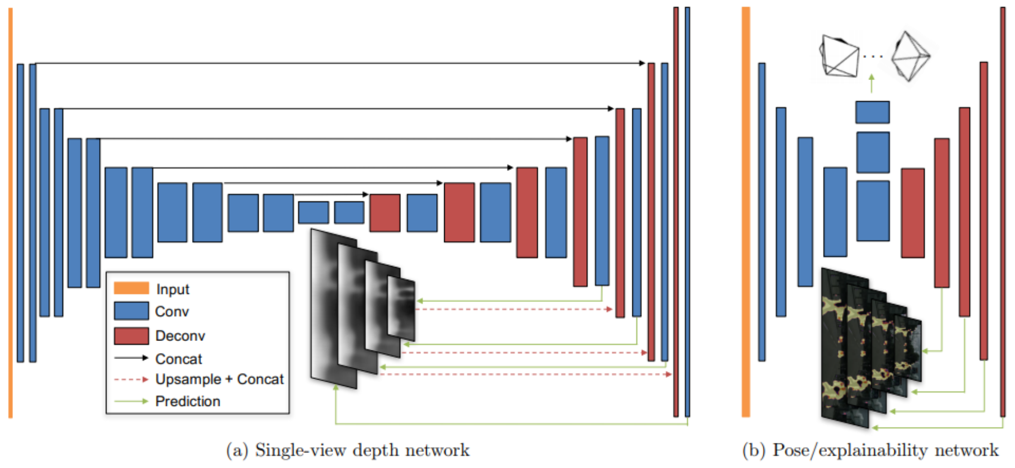

Input
Conv
Deconv
Concat
Upsample + Concat
Prediction
(a) Single-view depth network
(b) Pose/explainability network

Figure 2.4: SFM-Learner architecture [70]

Following the work of Godard et al. [20], Zhan et al. [66] proposed a method of unsupervised learning of depth and visual odometry by reconstruction of features. First, they warp the image, then they get the features of the warped image and then by having a loss over that they were able to train their network. Similarly, in the field of optical flow, which is same as the disparity in case of stereo, instead of warping the image and then get the features, Sun et al. [50] proposed first to warp the features, and then have volume loss over warped features. More recently, Yan et al. [35] and Yang et al. [63] trained a network for depth estimation and used it to boost the accuracy of traditional visual odometry.

Mahjourian et al. [37] proposed a method of estimating depth and ego-motion from monocular video using 3D geometric constraints. In addition to the 2D photometric loss, they convert depth map to point clouds then they used ICP to warp the point clouds and have a 3D loss as well. Yin et al.

proposed a jointly unsupervised learning of depth, optical flow, and ego motion [64]. They also propose an adaptive geometric consistency loss to increase the robustness about non-Lambertian regions where the photometric loss is not valid. At the same time, Wang et al. [57] proposed using a traditional differentiable visual odometry instead of using a network estimator for a more reliable estimation of poses during training.

2.2.3 Semi-supervised

Unlike unsupervised methods, there has not been much work on semi-supervised learning of depth. Smolyanskiy et al. [48] proposed a method of semi-supervised learning for stereo disparity estimation, but not for single image depth estimation. Recently, Luo et al. [36] and Guo et al. [21] proposed a method that consists of multiple sequential unsupervised and supervised training stages; hence their method could be categorized as a semi-supervised method although unlike us, they did not use LiDAR and stereo images at the same time in training.

Closest to our work is Kuznietsov et al. [31], who proposed adding the supervised and unsupervised loss terms in the final loss together resulting in using LiDAR and stereo at the same time in training. One of the main differences between [31] and ours is that we explicitly enforce the left-right consistency term first proposed by [20]. Having this term makes the prediction consistent between left and right. Another difference is that their supervised loss term was directly defined on the depth values, whereas we defined it on inverse depth instead. As discussed in [31], a loss term on depth values makes the training unstable because of the high gradients in the early stages of the training. To remedy the situation, Kuznietsov et al.[31] proposed to gradually fade in the supervised loss to achieve convergence whereas our method does not have this problem and does not need to fade in supervised or unsupervised loss terms. In Section 5.3, we show qualitatively and quantitatively that we can obtain better accuracy than [31], which is considered the state-of-the-art in semi-supervised single image depth estimation, as the result of the above considerations.

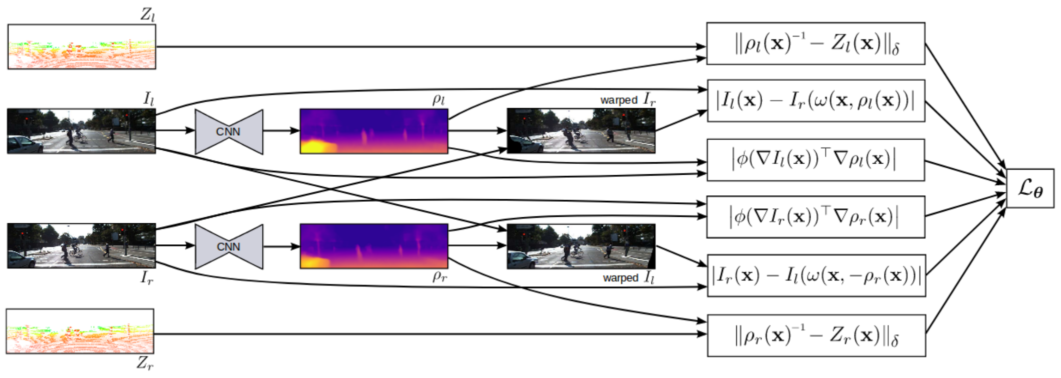

Z_l
I_l
CNN
ρ_l
warped I_r
I_r
CNN
ρ_r
warped I_l
Z_r
$\|\rho_l(\mathbf{x})^{-1} - Z_l(\mathbf{x})\|_\delta$
$|I_l(\mathbf{x}) - I_r(\omega(\mathbf{x}, \rho_l(\mathbf{x}))|$
$|\phi(\nabla I_l(\mathbf{x}))^\top \nabla \rho_l(\mathbf{x})|$
$|\phi(\nabla I_r(\mathbf{x}))^\top \nabla \rho_r(\mathbf{x})|$
$|I_r(\mathbf{x}) - I_l(\omega(\mathbf{x}, -\rho_r(\mathbf{x}))|$
$\|\rho_r(\mathbf{x})^{-1} - Z_r(\mathbf{x})\|_\delta$
$\mathcal{L}_{\boldsymbol{\theta}}$

Figure 2.5: components of the semi-supervised loss proposed by Kuznietsov et al. [70]. As shown the supervised loss is defined

Chapter 3

Method

In this thesis, for training, we use rectified stereo images as unsupervised and ground-truth depth data as supervised training. In the following section of this chapter, we will talk about our model architecture and the loss function used for training. To the best of our knowledge, we are the first one to use left-right consistency proposed by Godard et al. [20] in a semi-supervised framework of single image depth estimation.

3.1 Model Architecture

We use encoder-decoder architecture with skip connection. Skip connections were used to back-propagate the gradients better and also to use the early captured features of image for the output. Figure 3.1 shows an overview of our network. Our network inputs one image and outputs inverse depth corresponding to that image at four scales, and in our loss section, we defined loss for each scale. Each scale is half size of the next scale in width and height size. The dimensions of the four scale outputs are 256×512, 128×256, 64×128, and 32×64, respectively. Our model has over 59M parameters.

3.1.1 Encoder

Resnet50 was chosen for the encoder section. The schematic of our model architecture is shown in Figure 3.1. Figure 3.2 and 3.3 show the two types of resblocks in our resnet50 encoder architecture type A and type B. As shown in Figure 3.1 we have 16 resblocks. Resblock 3, 7, 13, and 16 are type B with

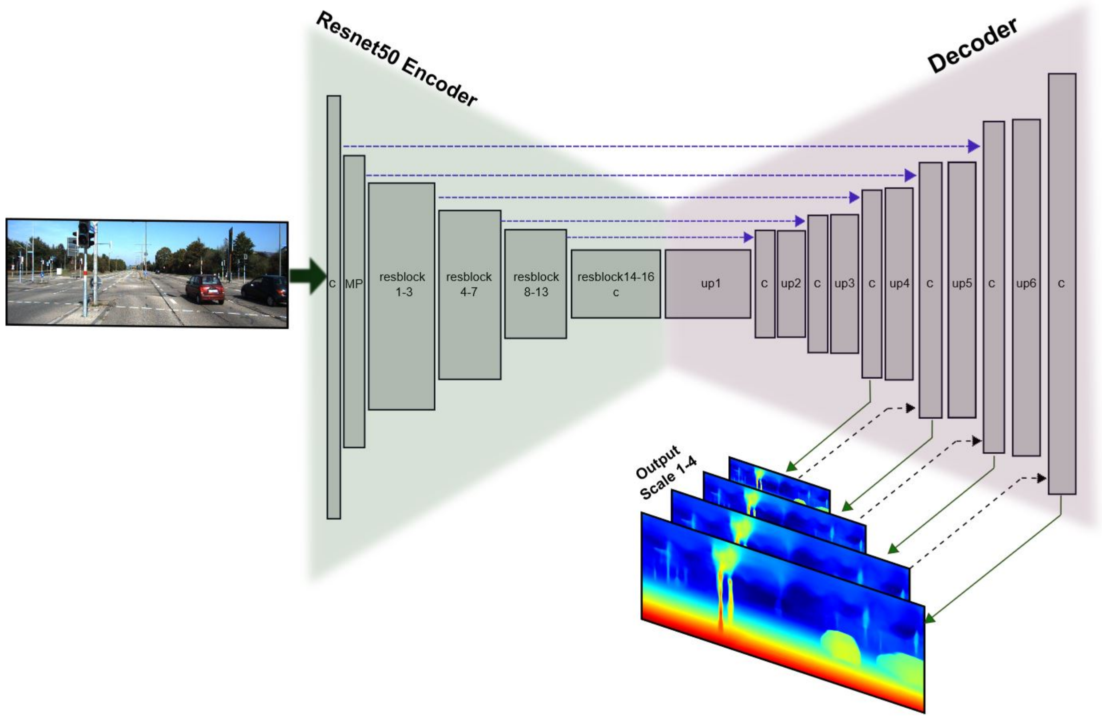


Figure 3.1: Overview of the schematic of our model architecture. C, MP, and up are convolution and max-pooling and upsampling layers, respectively. Blue arrow means there is going to be concatenation after. Green arrows consist of a layer of convolution. The black dashed arrow consists of resizing the image two times and concatenate it afterwards. Upsampling layer includes resizing two times and then convolution. We output four different scales and use these intermediate outputs for our loss.

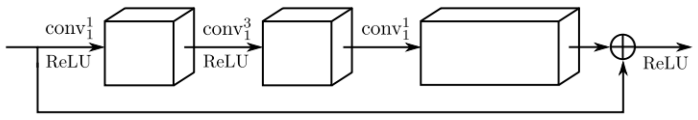


Figure 3.2: Type A residual block with stride 1. $conv_s^k$ denotes convolution with stride s and filter size $k \times k$. The residual is obtained from three consecutive convolutions. The output channel is same as the input [31].

stride $s = 2$, and the rest are type A with stride $s = 1$.

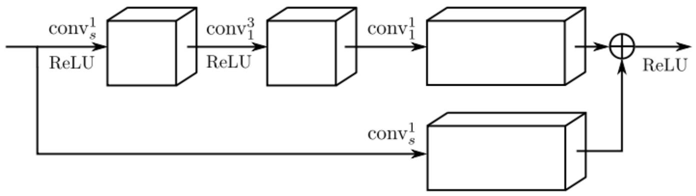


Figure 3.3: Type B residual block with stride 2. $conv_s^k$ denotes convolution with stride s and filter size $k \times k$. The residual is obtained from three consecutive convolutions. The first convolution applies on stride $s = 2$. Another additional convolution with stride $s = 2$ is applied and concatenated with the output of the three consecutive convolution. The output channel is double (for $s = 2$) as the input [31].

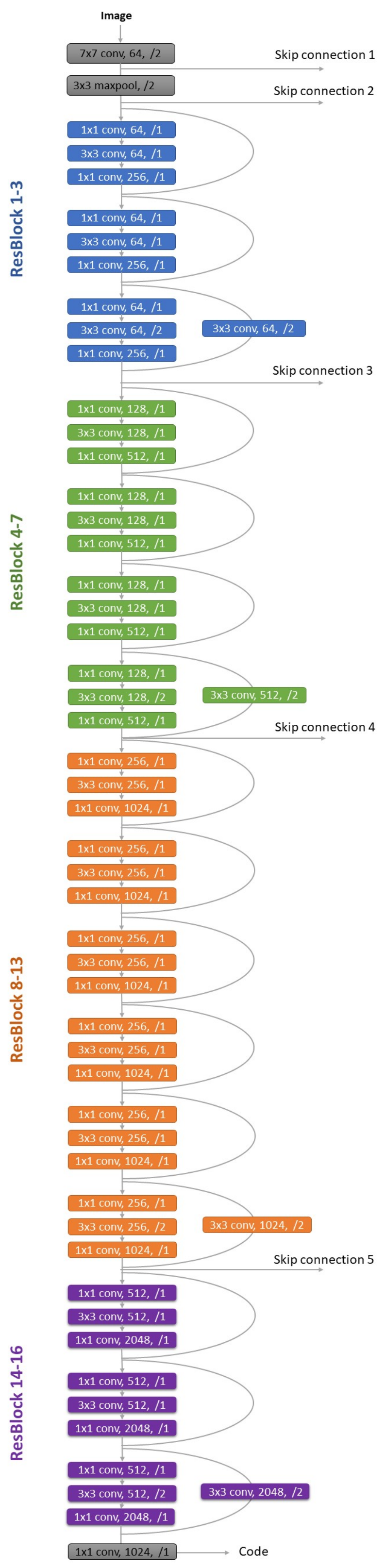


Figure 3.4: Resnet50 Encoder Architecture. "$k \times k$ $conv, i, /s$" denotes i number of convolutions with kernel size $k \times k$ and stride s. Skip connection and code will be connected to the decoder part.

3.1.2 Decoder

We use multiple upsampling layers which consist of nearest neighbour resizing followed by a convolution. We concatenate the skip connections and output the inverse depth at different scales. Figure 3.5 shows our detailed architecture of our decoder.

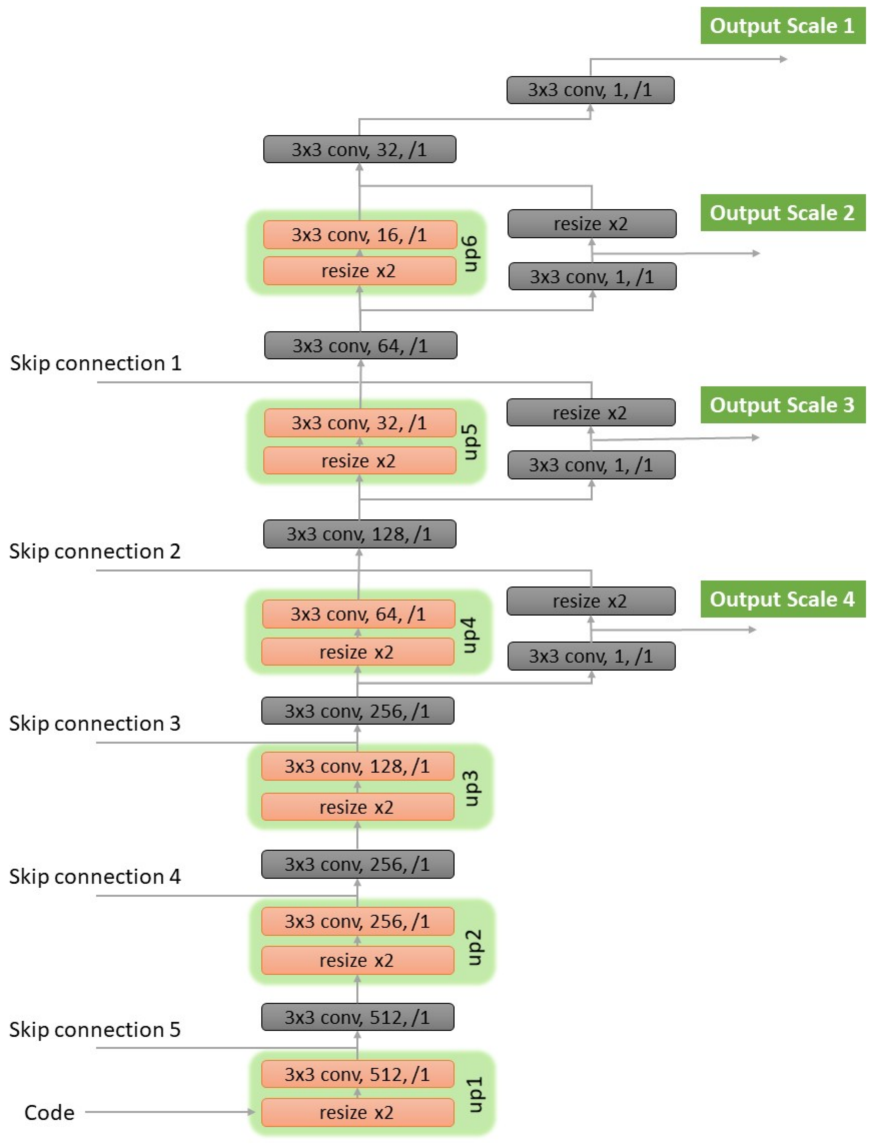


Figure 3.5: Decoder Architecture. Code and skip connections come from encoder. "$k \times k\ conv, i, /s$" denotes i number of convolutions with kernel size $k \times k$ and stride s. For resizing module, we used nearest neighbour method.

3.2 Loss Terms

One of the most critical factors in deep learning is defining the loss function. Minimizing the loss function should directly be aligned with the objective. Carefully designing the loss function is very important, especially in unsupervised training.

Figure 3.6 shows the different loss terms we use in our training phase, to be described in the current section.

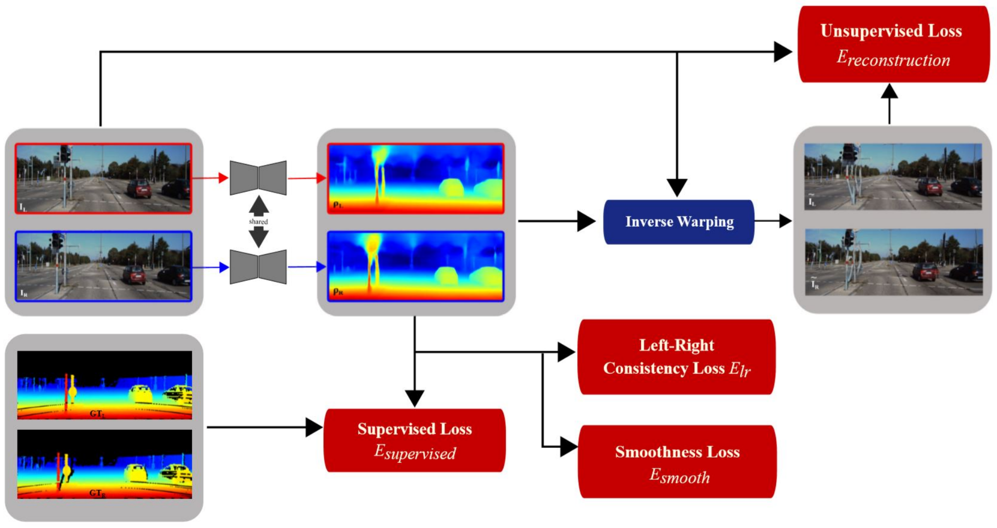


Figure 3.6: Overview of the schematic of our proposed loss. There are four terms in our loss $E_{reconstruction}$, $E_{supervised}$, E_{lr}, E_{smooth}, $E_{supervised}$. Subscript L and R refers to left and right image, respectively. ρ refers to output of our network inverse depth. We use bilinear sampler in the inverse warping function.

We define L_s for each output scale s. Hence the total loss is defined as $L_{total} = \sum_{s=1}^{4} L_s$.

$$L_s = \lambda_1 E_{reconstruction_s} + \lambda_2 E_{lr_s} + \lambda_3 E_{supervised_s} + \lambda_4 E_{smooth_s} \tag{3.1}$$

where λ_i are scalars and the E terms are defined below. By setting λ_i, we can define how much each loss term contribute to the final loss.

3.2.1 Unsupervised Loss $E_{reconstruction}$

We use photometric reconstruction loss between left and right image. Similar to other unsupervised methods, we assume photometric constancy between left-right images. Inverse warping has been used to get the estimated left/right image and then the estimated image is compared with its corresponding real image. In the inverse warping, bilinear sampler is used to make the pipeline differentiable. For comparison, we use the combination of the structural similarity (SSIM) and L1 used by Godard et al. [20], and the ternary census transform used in [38], [49], [65]. SSIM and the ternary census transform can compensate for the gamma and illumination change to some extent and result in improved satisfaction of the constancy assumption. Our unsupervised photometric image reconstruction loss term E_u is defined as follows:

$$\begin{aligned} E_{reconstruction} &= \sum_{k \in \{l,r\}} f(I^k, \tilde{I}^k) \\ f(I, \tilde{I}) &= \frac{1}{N} \sum_{i,j} \alpha_1 * \frac{1 - SSIM(I_{ij}, \tilde{I}_{ij})}{2} + \\ &\quad \alpha_2 * ||I_{ij} - \tilde{I}_{ij}||_1 + \\ &\quad \alpha_3 * census(I_{ij}, \tilde{I}_{ij}) \end{aligned} \tag{3.2}$$

where I^l, I^r, $\tilde{I}^l$, and $\tilde{I}^r$ are the left image, right image and their reconstructed images, respectively. N is the total number of pixels. α_1, α_2, and α_3 are scalars that define the contribution of each term to the total reconstruction loss.

SSIM

Structural similarity (SSIM) measures how two images are similar to each other. It gets two input images and outputs in a range of (-1,1]. Zero means they are totally different, and one means two images are equal, and negative one means two images are inverted versions of each other.

SSIM tries to simulate how the human visual system (HVS) works by calculating three different properties of an image in a $N \times N$ window where

$N = 11$ as default. These properties are luminance(l), contrast(c), structure (s).

$$\begin{aligned} & l(x,y) = \tfrac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \\ & c(x,y) = \tfrac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \\ & s(x,y) = \tfrac{2\sigma_{xy} + c_2}{2\sigma_x\sigma_y + c2} \\ & c_1 = (k_1 L)^2 \qquad c_2 = (k_2 L)^2 \end{aligned} \tag{3.3}$$

$$SSIM = l(x,y)^\alpha \times c(x,y)^\gamma \times s(x,y)^\delta \tag{3.4}$$

where μ and σ^2 denote the average and variance of pixel values in the window, respectively. L is the dynamic range of pixel values ($2^{(number\_of\_bits)} - 1$), and $k_1 = 0.01$, $k_2 = 0.03$ as default. By choosing $\alpha = \gamma = \delta = 1$ the equation 3.5 will become:

$$SSIM = \frac{(2\mu_x\mu_y + c_2)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{3.5}$$

Census

Census operator is a robust way of patch representation first proposed by Zabih and Woodfill [65]. It is a nonlinear transformation which compares a pixel with its local neighbourhood pixels.

Ternary census transform is defined in Figure 3.7. For a pixel P and its local neighbourhood pixel P', we calculate the difference and if the difference is less than ϵ we assign 1 to it and 0 if the difference is greater than ϵ and 2 for less than $-\epsilon$. Then we concatenate all numbers. For example, the ternary census transformed value for the middle pixel x shown in Figure 3.7 is going to be 21002222.

To calculate the census loss between I_1 and I_2, we first use ternary census transform operator to find every pixel representation, and then we use the hamming distance between ternary representations at each pixel. Final cencus loss is the sum of hamming distances between two images over all pixels.

$$\xi(P,P') = \begin{matrix} 0 & P-P' > \varepsilon \\ 1 & |P-P'| \le \varepsilon \\ 2 & P'-P > \varepsilon \end{matrix}$$

$$\begin{array}{|c|c|c|}\hline 124 & 74 & 32 \\ \hline 124 & 64 & 18 \\ \hline 157 & 116 & 84 \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|}\hline 2 & 1 & 0 \\ \hline 2 & x & 0 \\ \hline 2 & 2 & 2 \\ \hline \end{array} \rightarrow 21002222$$

Figure 3.7: Ternery cencus transform. P is middle pixle in the window and P' is the surrounding pixel. The ternary census transform of pixel x is 21002222 [49].

3.2.2 Left-Right Consistency Loss E_{lr}

To ensure the equal contribution of both left and right images in the network training, we feed left and right images independently to the network, and then we jointly optimize the output of the network such that the predicted left and right depth maps are consistent. As explained in [20], the left-right consistency loss attempts to make the inverse depth of the left (or right) view the same as the projected inverse depth of the right (or left) view. This type of loss is similar to forward-backward consistency for optical flow estimation [38]. We define our left-right consistency loss as follows:

$$E_{lr} = \frac{1}{N}\sum_{i,j} ||\rho^l_{ij} - \rho^r_{ij+d^l_{ij}}||_1 + ||\rho^r_{ij} - \rho^l_{ij+d^r_{ij}}||_1, \tag{3.6}$$

where ρ_l and ρ_r are the predicted inverse depth for left and right images, respectively. d^l and d^r are predicted disparities corresponding to left and right images, respectively. The conversion of inverse depth ρ to disparity d is calculated using (3.7):

$$d = baseline * f * \rho, \tag{3.7}$$

where f is the focal length of the camera, and *baseline* is the linear distance between the two cameras.

3.2.3 Supervised Loss $E_{supervised}$

The supervised loss term measures the difference between the ground truth inverse depth Z^{-1} and the predicted inverse depth ρ for the points Ω where the ground truth is available.

$$E_{supervised} = \sum_{k\in\{l,r\}} \frac{1}{M_k} \sum_{i,j\in\Omega_k} ||\rho^k_{ij} - {Z^{-1}}^k_{ij}||_1 \tag{3.8}$$

where Ω_l and Ω_r are the points where the ground truth depths are available for the left and right images, respectively. M_l and M_r are the total number of the pixels that ground truth is available for left and right images, respectively.

3.2.4 Smoothness Loss E_{smooth}

As suggested in [20], [31], the smoothness loss term is a regularization term that encourages the inverse depth to be locally smooth with a L_1 penalty on inverse depth gradients. We define our smoothness regularization term as:

$$E_{smooth} = \frac{1}{N} \sum_{k\in\{l,r\}} \sum_{i,j} |\partial_x \rho^k_{ij}| e^{-|\partial_x I^k_{ij}|} + |\partial_y \rho^k_{ij}| e^{-|\partial y I^k_{ij}|} \tag{3.9}$$

Since the depth is not continuous around object boundaries, this term encourages the neighbouring depth values to be similar in low gradient image regions and dissimilar otherwise.

Chapter 4

Datasets

In deep learning, one of the essential factors is having a dataset that can cover the distribution of real-life scenarios. While many datasets have been publicly available for indoor scenes, e.g., Tum RGB-D dataset, Scannet dataset, there has not been many datasets for outdoor scene due to expensive LiDAR equipment. In this chapter, we will discuss the datasets that most people have been training their network on for outdoor scenes.

4.1 Cityscape Dataset

Cityscape dataset [9] was released with semantic, instance-wise, dense pixel annotations of 30 classes. It was initially used for semantic segmentation task, but more recently, researchers have been using stereo images of this dataset to train single image depth prediction neural network unsupervised. It has about 25000 stereo images. Unfortunately, there is no LiDAR ground truth depth available. The depth data is provided by doing the stereo matching. Stereo matching algorithms are not accurate, and due to the triangulation, stereo depth accuracy is not good enough for far objects. This dataset has been recorded while driving in 50 cities during several month and summer, spring, and fall season with good weather conditions.

The image resolutions are 2048×1024 with baseline around 20cm-23cm different in each sequence. A sample pair of stereo images is provided in Figure 4.1. As shown the left portion of the right image includes noisy data. Hence, we take the center image of each image and crop the noisy part and car hood section on the bottom.

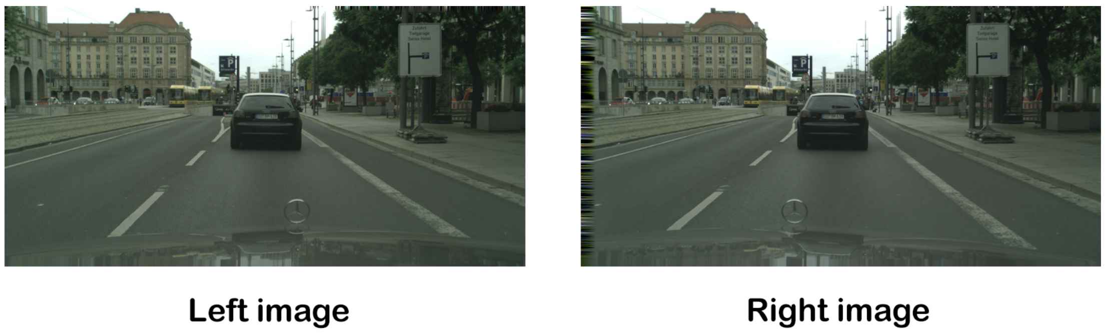


Figure 4.1: Cityscape stereo images sample. Images are being cropped such that left portion of right image and carhood section are not included.

4.2 Make3D Dataset

Make3D [46] consists of only RGB and depth pairs. The resolution of images is 1704×2272. Due to different aspect ratio of this dataset with KITTI dataset, we crop the images to have the same aspect ratio with KITTI. This dataset provides 400 images with aligned depth maps. Since the ground truth depth maps of this dataset are not perfectly aligned, and it is not considered a big dataset for deep learning, this dataset is not being used for training a deep neural network. However, this dataset is still being used for generalization tests. Figure 4.2 shows sample images from this dataset, including their aligned colour coded depth maps.

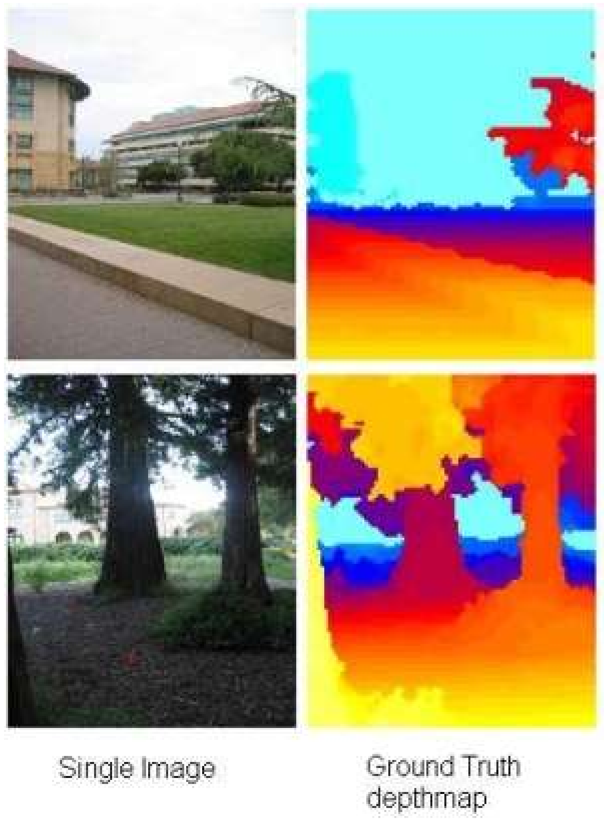


Figure 4.2: A sample of Make3D dataset and aligned depth maps. More red means more closer and more blue means furthur away pixels [46].

4.3 KITTI

This dataset [17] is the main dataset for training and evaluation of outdoor single image depth estimation tasks that people have been using, and therefore we will be using this dataset as our main dataset. This dataset was built for stereo, optical flow, visual odometry, 3D object detection and 3D tracking, and depth estimation tasks. They equipped a standard car with two high-resolution colour cameras, two monochrome cameras, GPS, IMU, and an accurate LiDAR, i.e., Velodyne. The Velodyne LiDAR only provides information for the bottom half of the RGB images due to the field of view difference. This dataset is captured by driving around cities in rural areas and highways. Therefore, it is an appropriate dataset for self-driving tasks. Up to 15 cars and 30 pedestrians are visible in each image.

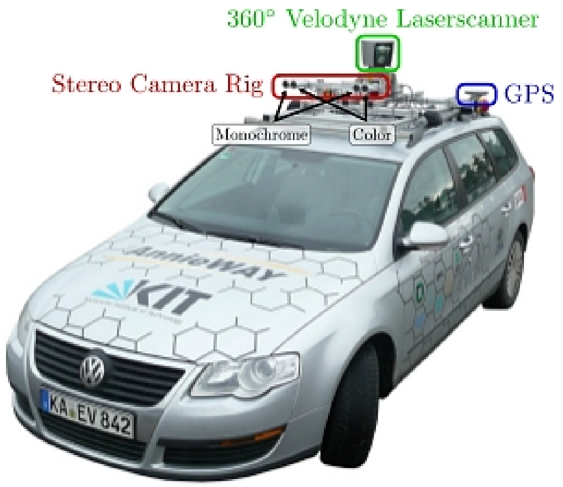


Figure 4.3: KITTI data acquisition setup [16].

Image resolutions of KITTI dataset varies from a sequence to another, e.g., 1242×375 , 1241×376 , 1224×370, 1226×370 , and 1238×374. To deal with different resolution, we simply resize the images and their corresponding depth maps to fit our network input size making all images the same resolution.

Besides the fact that raw dataset is available, we can also download the splits for each benchmark as well. We will discuss some splits and benchmarks related to our thesis in the Section 4.3.1. The KITTI also provides an online system to rank algorithms based on the accuracy of the evaluation metrics for

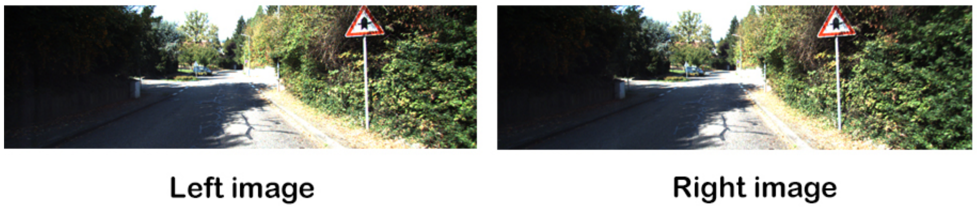


Figure 4.4: A sample stereo images of KITTI dataset.

each benchmark. Everyone can submit their results for the non-public testing splits in each benchmark.

4.3.1 KITTI Splits

There are various splits in each benchmark. The most common splits in the field of depth estimation are Eigen Split and KITTI splits.

Eigen Split

This split was created by Eigen et al.[12] and after that, most of the researchers follow this split to compare their algorithm with other researchers. They used 56 scenes from the “city,” “residential,” and “road” categories of the raw data and 28 scenes were used for the training set and other 28 scenes was used for testing. The training set has 800 images per scene. When the acceleration of the car is low, they exclude those shots to avoid stationary duplicate images. In total, the training set has 20K unique images, and the testing set has 697 unique images. For evaluation, since projected Velodyne LiDAR is noisy due to occlusions, the motion of the vehicles, etc., we used more accurate pre-processed depth map from KITTI depth benchmark called ”Annotated Depth Map.” Since depth benchmark only covers 652 out of 697 images, we evaluate based on these 652 accurate depth maps. We will talk about this more in Section 4.3.2.

KITTI Stereo 2015 Split

This split is also known as KITTI split used in [20]. It refers to KITTI stereo benchmark, which is a benchmark with the purpose of disparity estimation.

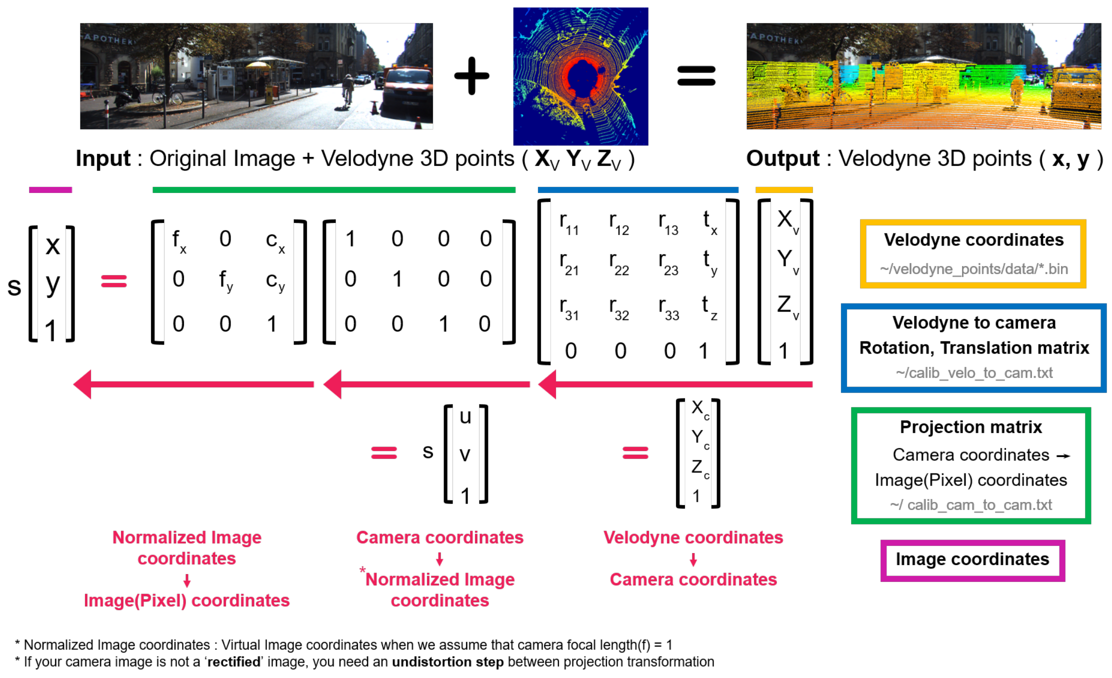
The training set includes 33 scenes with a total of 29,000 images. The testing set includes 28 scenes with a total of 200 high-resolution disparity maps. These disparity maps are manually labeled by inserting CAD models. Although the quality of these disparity maps is better than the projected Velodyne LiDAR, the CAD models result in incorrect values for transparent surfaces, e.g., car windows, and they are not perfectly aligned with the image as well.

4.3.2 Ground-truth LiDAR Projection

Since LiDAR measurements and the cameras are in different coordinates systems, we need to project LiDAR 3D points into the camera coordinates system. Figure **??** shows the steps we need for this transformations. The steps needed for this projection are as follows:

- Velodyne coordinates to camera coordinates: We need to use Velodyne to camera transformation matrix using calibration data.
- Camera coordinates to normalized image coordinates: Normalized image coordinates are the coordinates for a virtual image if we assume the focal length of the camera is at 1 meter. All the 3D points should be divided by their Z to force Z=1, as shown in Figure **??**.
- Normalized image coordinates to pixel coordinates: Finally, by multiplying the normalized image coordinates with the intrinsic camera matrix, we will be able to get the corresponding pixel coordinates.

Using this naïve way of projection will cause an occlusion artifact. Due to offset between the camera and LiDAR sensor, some LiDAR points are occluded from the camera point of view. Hence naïve projection will result in incorrect depth values for those points. We can preprocess projected LiDAR as described by Uhrig et al. [53] or use provided annotated depth map(c) for KITTI dataset. They used multiple adjacent frames to densify the depth map and use left-right consistency check combined with stereo matching algorithms consistency checks to filter out the outliers and noise in LiDAR data. They also use

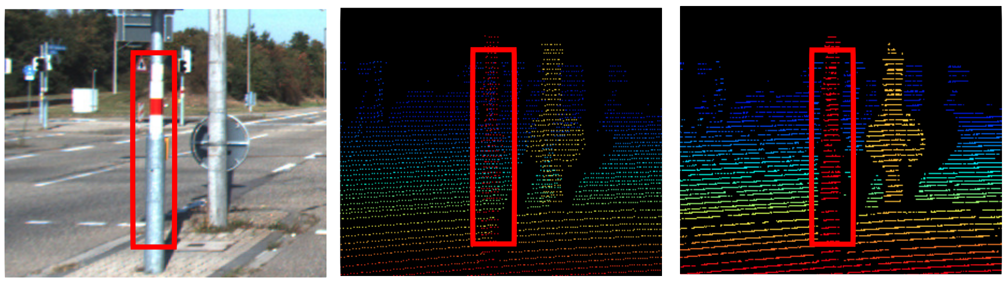


Input : Original Image + Velodyne 3D points (X_V Y_V Z_V)
Output : Velodyne 3D points (x, y)
Velodyne coordinates
~/velodyne_points/data/*.bin
Velodyne to camera Rotation, Translation matrix
~/calib_velo_to_cam.txt
Projection matrix
Camera coordinates → Image(Pixel) coordinates
~/ calib_cam_to_cam.txt
Image coordinates
Normalized Image coordinates → Image(Pixel) coordinates
Camera coordinates → *Normalized Image coordinates
Velodyne coordinates → Camera coordinates
* Normalized Image coordinates : Virtual Image coordinates when we assume that camera focal length(f) = 1
* If your camera image is not a 'rectified' image, you need an undistortion step between projection transformation

$$s\begin{bmatrix}x\\y\\1\end{bmatrix}=\begin{bmatrix}f_x&0&c_x\\0&f_y&c_y\\0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\end{bmatrix}\begin{bmatrix}r_{11}&r_{12}&r_{13}&t_x\\r_{21}&r_{22}&r_{23}&t_y\\r_{31}&r_{32}&r_{33}&t_z\\0&0&0&1\end{bmatrix}\begin{bmatrix}X_v\\Y_v\\Z_v\\1\end{bmatrix}$$

$$= s\begin{bmatrix}u\\v\\1\end{bmatrix} \qquad = \begin{bmatrix}X_c\\Y_c\\Z_c\\1\end{bmatrix}$$

the speed of the vehicle to untwist the LiDAR points and publish the post-processed depth maps as the annotated depth map. Figure 4.5 shows the comparison of the naïve projection and the projection using Uhrig et al. [53]. Later on in Section 5.3, we showed that significant accuracy boost could be gained if we use denser and more accurate depth map.



(a) image (b) projected raw LiDAR (c) annotated depth map

Figure 4.5: Qualitative comparison between (b) projected raw LiDAR containing occlusion artifacts due to the displacement between the camera and LiDAR and (c) annotated depth map without any occlusion artifact. We use annotated depth map dataset (c) for our training and evaluation. (b) shows the erroneous depth values for points (green pixels among red for the pole bounded by the red rectangle) that are occluded from the camera point of view but not LiDAR point of view.

Figure 4.6 shows the reason for the occlusion artifact due to the offset be-

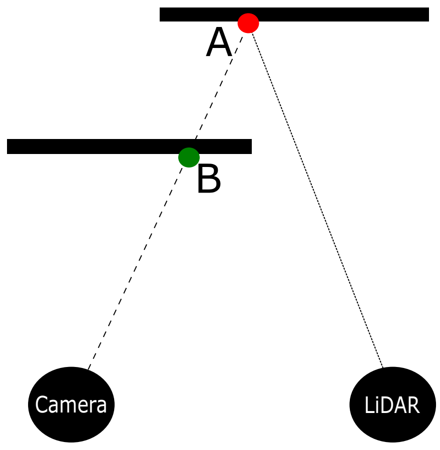


Figure 4.6: Occlusion artifact top-down view. Projection of point A from the LiDAR point of view will be point B in the camera point of view. Naive projection will assign point A's depth to it's projection, e.g., point B which is wrong

tween the LiDAR sensor and camera. Most of the points that LiDAR can see but they are occluded from the camera point of view will have this occlusion artifact, i.e., Point A. Some points, i.e., A and B, will have the same projection in the camera point of view. Therefore their depth value will overlap each other, and we can easily ignore the bigger value but in practice, the exact overlap of measurements will not often happen due to sparse LiDAR measurements.

Chapter 5

Experiments

For comparison, we use the popular Eigen split [12] in KITTI dataset [53] that has been used in the previous methods. Using this split, we notice the same problem mentioned by Aleotti et al. [2] that, when LiDAR points are projected into the camera space, an artifact results around objects that are occluded in the image but not from the LiDAR point of view. This is due to the displacement between the LiDAR and the camera sensors (see section 4.3.2). Recently Uhrig et al. [53] provided preprocessed annotated depth maps of KITTI by a preprocessing step on projected raw LiDAR data. They used multiple sequences, left-right consistency checks, and untwisting methods to carefully filter out outliers and densify projected raw LiDAR point clouds. Figure 4.5 shows the occlusion artifact in raw projected LiDAR and the corresponding annotated depth map dataset provided by [53]. Since the occlusion artifact is filtered out in the annotated depth ground truth, we train our model with this more accurate ground truth. The first and the third row of Table 5.2 show the effect of the training network with the projected raw LiDAR versus the annotated ground truth.

5.1 Evaulation Metrics

We use the standard metrics used by previous researchers [12], [20], [31]. Specifically, we use RMSE, RMSE$_{log}$, absolute relative difference (Abs Rel), squared relative difference (Sq Rel), and the percentage of depths (δ) within a certain threshold distance to its ground truth.

RMSE: $\sqrt{\frac{1}{N}\sum_{i=1}^{N} ||\rho(x_i)^{-1} - Z(x_i))||_2^2}$

RMSE$_{\mathbf{log}}$: $\sqrt{\frac{1}{N}\sum_{i=1}^{N} ||\log\rho(x_i)^{-1} - \log Z(x_i))||_2^2}$

ARD: $\frac{1}{N}\sum_{i=1}^{N} \frac{|\rho(x_i)^{-1} - Z(x_i))|}{Z(x_i)}$

SRD: $\frac{1}{N}\sum_{i=1}^{N} \frac{|\rho(x_i)^{-1} - Z(x_i))|^2}{Z(x_i)}$

ACCURACY $\%\ of\ x_i\ s.t.\ max(\frac{Z(x_i)}{\rho(x_i)^{-1}}, \frac{\rho(x_i)^{-1}}{Z(x_i)}) = \delta < thr$

where N is the total number of pixels with ground-truth depth available.

5.2 Implementation Details

We train our network from scratch using Tensorflow [1]. Our network and training procedure are identical to the Resnet50 network used by Godard et al. [20] except for the decoder part in which we have one output instead of two for each scale. Similar to Godard et al. [20], all inputs are resized to 256×512. The output of the network, i.e., inverse depth, is limited to 0 to 1.0 using the sigmoid function. We use Adam optimiser [29] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ with initial learning rate of $\lambda = 10^{-4}$, and that remains constant for the first 15 epochs and being halved every 5 epochs for the next 10 epochs for a total of 25 epochs. The hyperparameters for loss are chosen as $\lambda_1 = 1$, $\lambda_2 = 1.0$, $\lambda_3 = 150.0$, $\lambda_4 = 0.1$, $\alpha_1 = 0.85$, $\alpha_2 = 0.15$, and $\alpha_3 = 0.08$.

5.3 Results

5.3.1 The Effect of Each Term

Disparity vs. Inverse Depth

To investigate the effect of changing the output from disparity to inverse depth, Table 5.1 experiment has been done. The model is trained on cityscape dataset and has been tested directly on 200 images of KITTI Stereo 2015 split. In Monodepth architecture [20], there is an assumption of baseline and focal length being constant and similar for the entire training split which does not hold for cityscape dataset. Cityscape dataset includes different stereo baseline and different camera intrinsic. By changing the output of the model to inverse depth, we are able to overcome this issue by feeding the network with persistent output. This output, i.e., inverse depth, is not related to camera settings and as shown in Table 5.1, training on inverse depth will lead to further improvement in accuracy because, by training using inverse depth, our output becomes independent of the camera setup.

Output	Abs Rel	Sq Rel	RMSE	RMSE$_{log}$
disparity	0.296	3.909	9.518	0.321
inverse depth	**0.284**	**3.338**	**8.914**	**0.307**

Table 5.1: The effect of the having output as disparity versus inverse depth. The model is trained on cityscape dataset and tested on KITTI split. Training model based on inverse depth output allows us to overcome the limitation of all training data being from the same stereo camera setup and will lead to more accurate results.

Naive Lidar Projection vs. Annotated Depth Map and Adding Left-right Consistency term

To investigate in detail the effect of using left-right consistency term in the loss function and using the annotated LiDAR ground truth, we recorded result by just changing one variable in Table 5.2, where 200 images of KITTI Stereo 2015 split [39] were used in this controlled experiment. First and third rows show the effect of using annotated depth map. Training network using annotated depth maps significantly improved the accuracy on all metrics. Second row and third rows show the effect of adding left-right consistency term in our semi-supervised framework. The result shows we get a small improvement by adding this term to our final loss. Our proposed method is shown in bold in Table 5.2

Training Data		Loss Term				
Projected Raw LiDAR	Annotated Depth map	Left-Right Consistency	Abs Rel	Sq Rel	RMSE (meter)	RMSE$_{log}$
✓		✓	0.120	1.154	5.614	0.204
	✓		0.110	0.973	5.373	0.191
	✓	✓	**0.108**	**0.949**	**5.369**	**0.190**

Table 5.2: The effect of the left-right consistency term and using annotated depth map in our semi-supervised training. The results are evaluated on 200 images of KITTI Stereo 2015 split [39]. The second and the third row show that exploiting left-right consistency helps to achieve better accuracy. The first and the third rows show training on annotated depth map significantly reduces error. The result from our method is shown in bold.

5.3.2 Comparison Against Other Methods

We evaluate our method based on the official KITTI annotated depth map rather than noisy projected raw LiDAR. Table 5.3 contains the quantitative evaluation of the projected raw LiDAR based on the provided annotated depth map ground truth if a depth value of a pixel exists in the both annotated depth map and projected raw LiDAR (54.89% of the LiDAR points have been evaluated). The large error for projected raw LiDAR suggests that raw LiDAR is not as accurate as annotated depth maps and should not be considered as ground truth for evaluation.

Table 5.3 shows the quantitative comparison with the state of the art methods in Eigen split using reliable annotated depth maps for training and testing. Although supervised methods, e.g., DORN [14], can achieve better quantitative performance according to some metrics than semi-supervised methods, they produce an inaccurate prediction of the top portion of the image, which can be seen in Figure 5.2, where the LiDAR's field of view is different from that of the camera.

By treating left and right images equivalently and defining our loss symmetrically, we eliminate the post-processing step needed in [20]. As shown in Table 5.3, our unsupervised model outperforms our baseline unsupervised model [20]. Besides, from Table 5.3 among the evaluated semi-supervised methods, our method outperforms [31], considered the state-of-the-art, with respect to the majority of the performance metrics.

method	type	Dataset	lower is better				higher is better		
			Abs Rel	Sq Rel	RMSE (meter)	RMSE_{log}	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
raw LiDAR	-	-	0.010	0.126	1.209	0.054	0.993	0.996	0.998
DORN [14]	S	*K*	**0.080**	**0.332**	**2.888**	**0.120**	**0.938**	**0.986**	**0.995**
SemiDepth(Ours)	S	*K*	0.096	0.552	3.995	0.152	0.892	0.972	0.992
Monodepth [20](Resnet50)	U	*C+K*	0.085	0.584	3.938	0.135	0.916	**0.980**	**0.994**
MonoGAN [2](Resnet50)	U	*C+K*	0.096	0.699	4.236	0.150	0.899	0.974	0.992
SemiDepth(Ours)	U	*C+K*	**0.082**	**0.551**	**3.837**	**0.134**	**0.920**	**0.980**	0.993
Kuznietsov et al. [31]	Semi	*I+K*	0.089	0.478	3.610	0.138	0.906	0.980	0.995
SVSM FT [36]	Semi	*I+F+K*	**0.077**	**0.392**	3.569	0.127	0.919	0.983	0.995
SemiDepth (full) (Ours)	Semi	*C+K*	0.078	0.417	**3.464**	**0.126**	**0.923**	**0.984**	**0.995**

Table 5.3: Quantative evaluation on 652 (93.5%) test images of Eigen Split from the KITTI Dataset. We use official annotated depth map dataset as ground truth instead of noisy projected raw LiDAR. U, S, Semi means unsupervised, supervised and semi-supervised training, respectively. Results of [2], [20] are achieved without the post-processing step. *I*, *C*, *K*, and *F* refer to ImageNet [11], Cityscapes [9], KITTI [53] and FlyingThings3D datasets, respectively. *I* indicates that an encoder is initialized with a pre-trained model trained on ImageNet. All evaluations are using crop from [15]. Depth is capped at 80.0 meters.

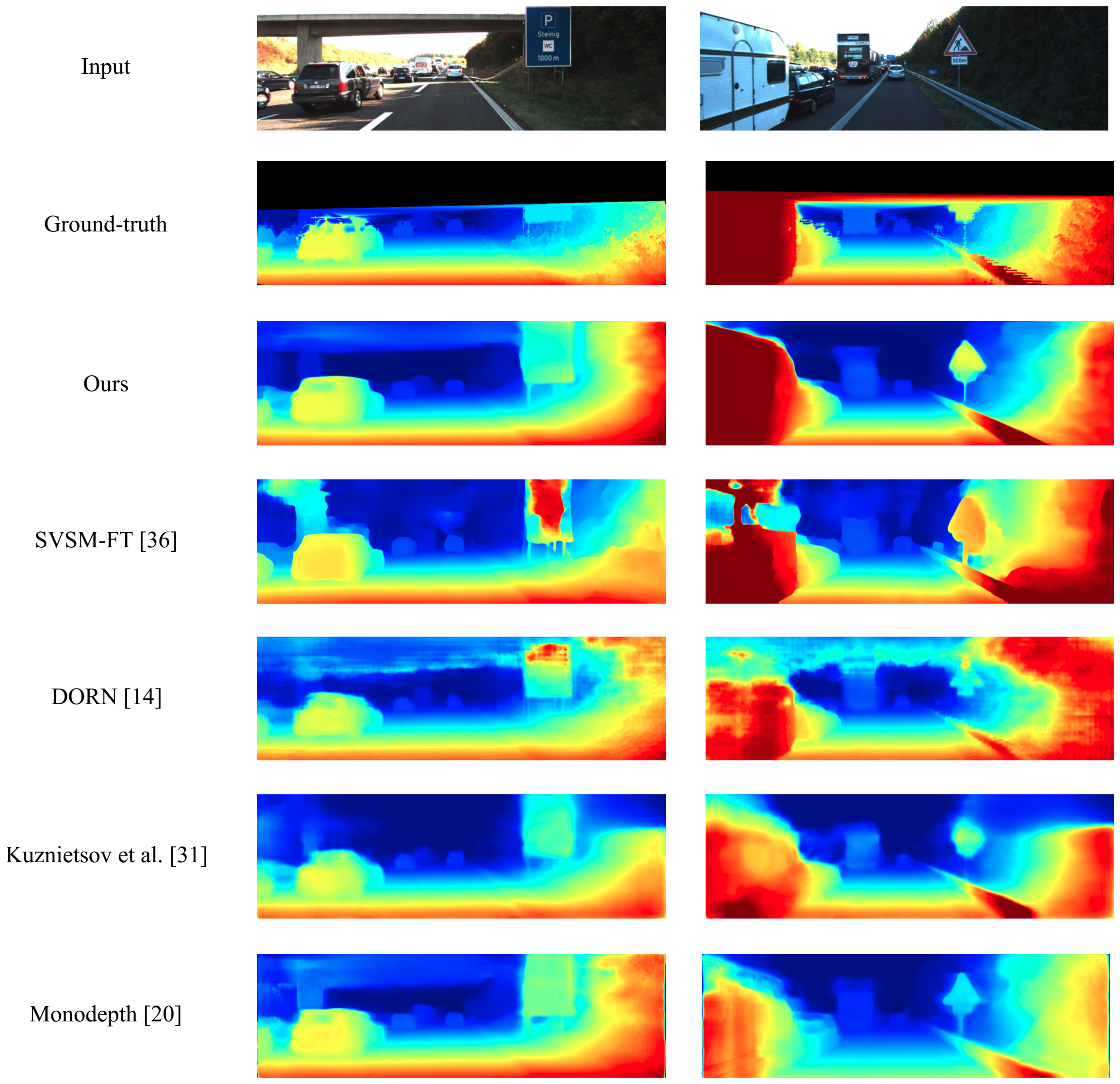


Figure 5.1: Qualitative comparison between state-of-the-art methods. We use interpolation in ground truth for visualization purpose.The depthmap is color coded. More red means closer pixels and more blue means further away pixels.

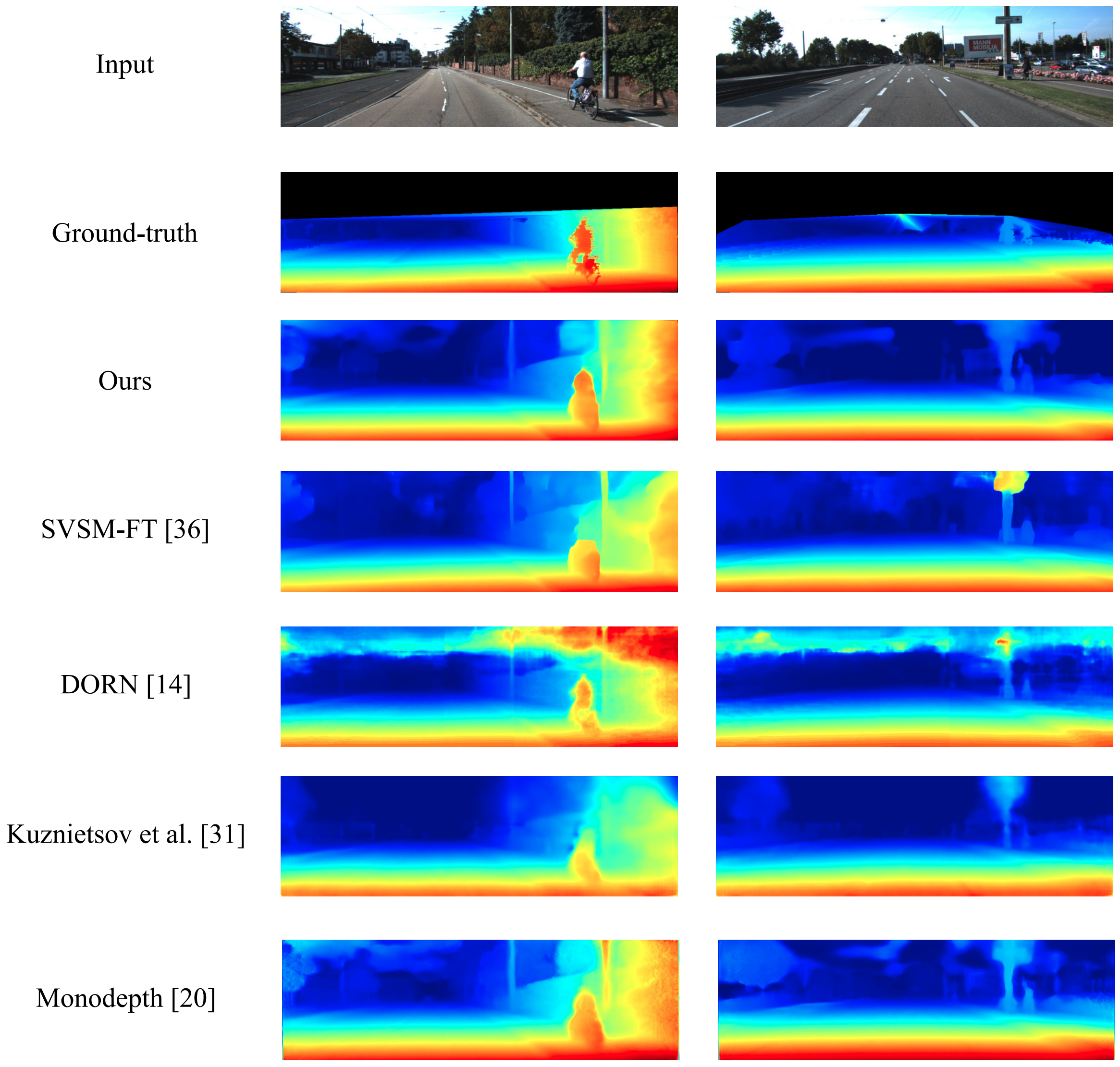


Figure 5.2: Qualitative comparison between state-of-the-art methods. We use interpolation in ground truth for visualization purpose.The depthmap is color coded. More red means closer pixels and more blue means further away pixels.

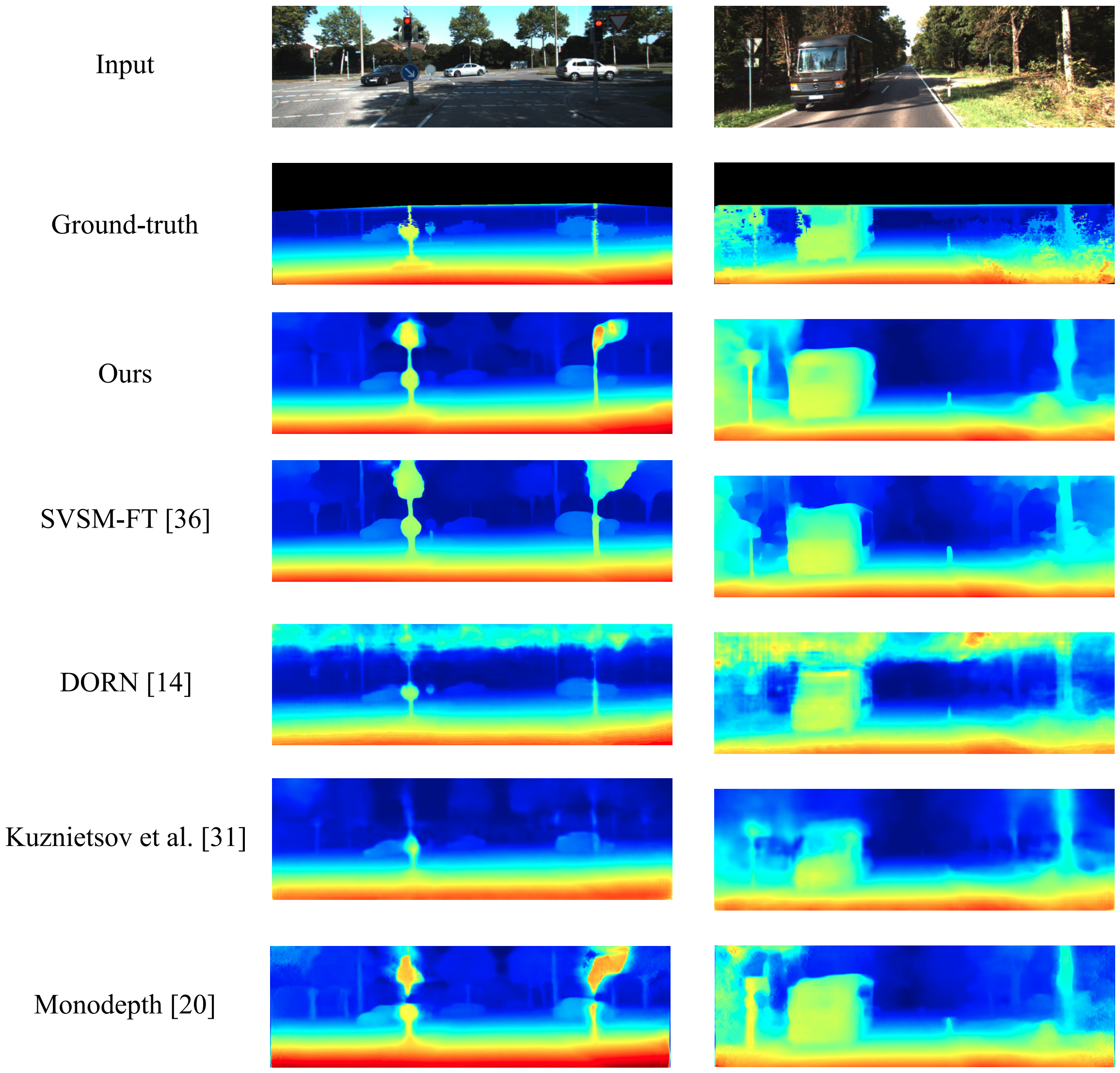


Figure 5.3: Qualitative comparison between state-of-the-art methods. We use interpolation in ground truth for visualization purpose.The depthmap is color coded. More red means closer pixels and More blue means further away pixels.

Figure 5.1, 5.2, and 5.3 show some samples of the results for 6 images. Each column corresponds to an image. The first row shows the images fed to the neural network as an input. The rest of the rows are depth maps associated with the image on its column. The depth maps are in the range of 1 meter to 80 meters and colour coded. The more hot colour, i.e. red, means the object is closer to the camera. The more cold colour, i.e. blue, means the object is far away from the camera. The top row is the image as input. The second column is the ground truth. Black colour means there is no depth data available to interpolate. Note that the very top colour coded portion of the ground truth might be wrong due to interpolation for visualization purposes. The third row shows our result given the input image. The fourth row is the result of Lue et al. [36] on their best-proposed model, i.e. SVSM fine-tuned. The fifth to seventh rows show the result of Fu et al. [14], Kuznietsov et al. [31], and Godard et al. [20], respectively.

In Figure 5.1, for both left and right images, by focusing on the traffic sign and the car, we can see our method is able to predict consistent and more reliable depth for the traffic sign. The depth map shows the car is closer than the traffic sign as it appears from the image. For some other methods, e.g. SVSM-FT [36] and DORN [14], the depth is not consistent for either the car or the traffic signs.

In Figure 5.2, for the left image, if we focus on the bicycle, we can see our method is producing better depth while other methods are missing correct depth value especially around the head of the cyclist. For the right image, if we focus on the top portion of the image, we can see we are able o produce better and meaningful results for the sky and tree on the left side of the image.

In Figure 5.3, for the left image, if we focus on the traffic light and the traffic sign, we can see ours is able to better predict depth for the poles and especially the top portion of the image. For the right image, our method can produce consistent depth for the side of the car while others are missing the boundaries or producing wrong depth values.

In general, unlike other methods [14], [31], our method is able to predict more accurate depth, especially for small objects, e.g., pole, traffic lights, and traffic signs. Furthermore, we predict more accurate depth maps for the top portion of the image. Adding the left-right consistency term and also training on annotated depth maps helped us achieve these improvements. Furthermore, by outputting inverse depth instead of disparity, we can train our deep neural network with different baseline setups without confusing the network. This helps convergence and results in slight improvement.

Chapter 6

Conclusion and Future work

In this thesis, we have presented our approach to semi-supervised training of a deep neural network for single-image depth prediction. Our network uses a novel loss function that uses the left-right consistency term, which has not been used in the previous semi-supervised training of depth-prediction networks. Besides, we have explained and experimentally confirmed that, for optimal prediction result, in either supervised or semi-supervised training, careful use of the LiDAR data as the ground truth is essential. We have shown in order to take advantage of huge datasets, our output should not be related to the camera settings, i.e. camera intrinsic and baseline. Hence by changing the output from disparity to inverse depth, we can properly train the neural network for different camera intrinsics. To evaluate depth maps, we have shown that naïve LiDAR projection has high error and should not be used for evaluation purposes. Extensive experiments have been conducted to evaluate our proposed training approach, and we are able to achieve state-of-the-art performance in depth prediction accuracy. Our network model is publicly available in both training and inference.

One of the challenges of this work, in general, deep learning, is the training time. After every change, we needed to wait for 12 hours in order for our model to be trained for 25 epochs. In future, we would like to improve the accuracy of depth estimation by using a higher resolution of the images as input, and instead of training from scratch, we would like to use different encoder networks architectures pre-trained with ImageNet classification task.

We would also like to incorporate surface normals into the depth prediction training.

The field of single image depth estimation is considered one of the most important and fast-growing fields in computer vision. The results are becoming more robust and accurate due to better exploitation of the human visual system and availability of the bigger datasets, and better neural network architectures. In the past recent years, researchers have been explicitly exploiting human visual system cues and geometric constraints in the training procedure to enhance accuracy. These exploits include using stereo cameras (i.e. when we use two eyes), depth from defocus (i.e. when we change eye focus for objects with different depths), using gravity as a prior for depth, using semantics to increase accuracy (i.e. knowing of an object type will define a prior about the depth of the object), using local planar continuity (i.e. knowing some points 3D positions on a plane will help to recognize other points' depths on the same plane, and surface normal smoothness in a local plane).

One of the most important visual cues yet to be investigated in the deep learning framework is the effect of the occlusions known as depth ordering via occlusion reasoning. Occlusions are important cues to our visual system. For example, if part of a building is occluded by a car that means the car is in front of the building, and this means the car is closer to the camera, and its depth is smaller than the depth of the building. Although the network might learn the occlusion effect during the training, this type of occlusion effect in a single image was never explicitly incorporated into the training. Another HVS clue which needs to be investigated in deep neural networks is incorporating the effect of the perspective vanishing points. Vanishing points give us strong priors about the depth along those lines.

One other important aspect of the single image depth estimation field is its closeness to the optical/scene flow and stereo matching fields. Methods in all these fields are very similar to each other, e.g. the design of the neural network architecture, handling occlusions in stereo warping, unsupervised training using image warping.

Generalization was always an important issue of deep learning solutions.

Single image depth estimation was first developed for indoor scenes, and then outdoor scenes due to initially the existence of dense depth ground truth for indoors. One of the unsolved problems in this field is the nonexistence of a single model which can perform accurately for both indoors and outdoors scenarios. Generalization for both indoor and outdoor scenes might be one of the future lines of the research in this field.

Although single image depth estimation accuracy is not close to LiDAR accuracy yet, given how fast this field is going forward, soon, we will not need expensive LiDAR and other depth sensors. We can sense 3D only with a regular camera.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283. 42

[2] F. Aleotti, F. Tosi, M. Poggi, and S. Mattoccia, "Generative adversarial networks for unsupervised monocular depth prediction," in *15th European Conference on Computer Vision (ECCV) Workshops*, 2018. 17, 41, 46

[3] R. Alkemade, "Depth perception for augmented reality using parallel mean shift segmentation," 2010. 1

[4] Y. Almalioglu, M. R. U. Saputra, P. P. de Gusmao, A. Markham, and N. Trigoni, "Ganvo: Unsupervised deep monocular visual odometry and depth estimation with generative adversarial networks," *arXiv preprint arXiv:1809.05786*, 2018. 17

[5] A. Atapour-Abarghouei and T. P. Breckon, "Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, vol. 18, 2018, p. 1. 17

[6] Y. Cao, Z. Wu, and C. Shen, "Estimating depth from monocular images as classification using deep fully convolutional residual networks," *IEEE Transactions on Circuits and Systems for Video Technology*, 2017. 17

[7] P. Chakravarty, K. Kelchtermans, T. Roussel, S. Wellens, T. Tuytelaars, and L. Van Eycken, "Cnn-based single image obstacle avoidance on a quadrotor," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'17)*, IEEE, 2017, pp. 6369–6374. 1

[8] K. Čopič Pucihar and P. Coulton, "Estimating scale using depth from focus for mobile augmented reality," in *Proc. 3rd ACM SIGCHI symposium on Engineering interactive computing systems*, ACM, 2011, pp. 253–258. 1

[9] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223. 33, 46

[10] A. CS Kumar, S. M. Bhandarkar, and M. Prasad, "Monocular depth prediction using generative adversarial networks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 300–308. 17

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009. 46

[12] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in neural information processing systems*, 2014, pp. 2366–2374. 16, 17, 37, 41

[13] S. Elkerdawy, H. Zhang, and N. Ray, "Lightweight monocular depth estimation model by joint end-to-end filter pruning," *arXiv preprint arXiv:1905.05212*, 2019. 11

[14] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'18)*, 2018, pp. 2002–2011. 17, 18, 45, 46, 50

[15] R. Garg, V. K. BG, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *Proc. European Conference on Computer Vision (ECCV'16)*, Springer, 2016, pp. 740–756. 18, 19, 46

[16] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2012, pp. 3354–3361. 36

[17] ——, "Are we ready for autonomous driving," in *Proc. CVPR*, pp. 3354–3361. 36

[18] G. Georgakis, S. Karanam, Z. Wu, J. Ernst, and J. Košecká, "End-to-end learning of keypoint detector and descriptor for pose invariant 3d matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1965–1973. 11

[19] C. Godard, O. Mac Aodha, and G. Brostow, "Digging into self-supervised monocular depth estimation," *arXiv preprint arXiv:1806.01260*, 2018. 19

[20] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*, 2017, pp. 270–279. 12, 18–21, 23, 29, 31, 32

[21] X. Guo, H. Li, S. Yi, J. Ren, and X. Wang, "Learning monocular depth by distilling cross-domain stereo networks," in *Proc. European Conference on Computer Vision (ECCV'18)*, Sep. 2018, pp. 484–500. 20, 21

[22] K. Gwn Lore, K. Reddy, M. Giering, and E. A. Bernal, “Generative adversarial networks for depth map estimation from rgb video,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1177–1185. 17

[23] J. Hays. (2017). Stereo matching, [Online]. Available: `https://www.cc.gatech.edu/~hays/compvision2017/lectures/15.pdf` (visited on 07/04/2019). 3

[24] D. Hoiem, A. A. Efros, and M. Hebert, “Automatic photo pop-up,” in *ACM transactions on graphics (TOG)*, ACM, vol. 24, 2005, pp. 577–584. 4, 15

[25] J. Hu, M. Ozay, Y. Zhang, and T. Okatani, “Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries,” *arXiv preprint arXiv:1803.08673*, 2018. 17

[26] J. Hur and S. Roth, “Mirrorflow: Exploiting symmetries in joint optical flow and occlusion estimation,” in *Proc. IEEE Internation Conference on Computer Vision (ICCV’17)*, 2017. 20

[27] E. Ilg, T. Saikia, M. Keuper, and T. Brox, “Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation,” in *Proc. European Conference on Computer Vision (ECCV’18)*, Sep. 2018. 20

[28] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR’17)*, 2017, pp. 1125–1134. 17

[29] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014. 42

[30] J. N. Kundu, P. K. Uppala, A. Pahuja, and R. V. Babu, “Adadepth: Unsupervised content congruent adaptation for depth estimation,” *arXiv preprint arXiv:1803.01599*, 2018. 17

[31] Y. Kuznietsov, J. Stückler, and B. Leibe, “Semi-supervised deep learning for monocular depth map prediction,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR’17)*, 2017, pp. 6647–6655. 21, 24, 25, 32, 41, 45, 46

[32] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, “Deeper depth prediction with fully convolutional residual networks,” in *International Conference on 3D Vision (3DV’16)*, IEEE, 2016, pp. 239–248. 17

[33] J.-H. Lee, M. Heo, K.-R. Kim, and C.-S. Kim, “Single-image depth estimation based on fourier domain analysis,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR’18)*, 2018, pp. 330–339. 17

[34] 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 *Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'15)*, 2015, pp. 1119–1127. 17

[35] S. Y. Loo, A. J. Amiri, S. Mashohor, S. H. Tang, and H. Zhang, "Cnn-svo: Improving the mapping in semi-direct visual odometry using single-image depth prediction," *arXiv preprint arXiv:1810.01011*, 2018. 1, 20

[36] Y. Luo, J. Ren, M. Lin, J. Pang, W. Sun, H. Li, and L. Lin, "Single view stereo matching," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, 2018, pp. 155–163. 21, 46, 50

[37] R. Mahjourian, M. Wicke, and A. Angelova, "Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, 2018, pp. 5667–5675. 20

[38] S. Meister, J. Hur, and S. Roth, "Unflow: Unsupervised learning of optical flow with a bidirectional census loss," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 29, 31

[39] M. Menze, C. Heipke, and A. Geiger, *Object scene flow*, 2018. 43, 44

[40] M. Neoral, J. Šochman, and J. Matas, "Continual occlusions and optical flow estimation," *arXiv preprint arXiv:1811.01602*, 2018. 20

[41] A. Pilzer, D. Xu, M. Puscas, E. Ricci, and N. Sebe, "Unsupervised adversarial depth estimation using cycled generative networks," in *International Conference on 3D Vision (3DV'18)*, IEEE, 2018, pp. 587–595. 17

[42] X. Qi, R. Liao, Z. Liu, R. Urtasun, and J. Jia, "Geonet: Geometric neural network for joint depth and surface normal estimation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, 2018, pp. 283–291. 17

[43] J. Qiu, Z. Cui, Y. Zhang, X. Zhang, S. Liu, B. Zeng, and M. Pollefeys, "Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image," *arXiv preprint arXiv:1812.00488*, 2018. 11

[44] D. Rao, Q. V. Le, T. Phoka, M. Quigley, A. Sudsang, and A. Y. Ng, "Grasping novel objects with depth segmentation," in *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS'10)*, IEEE, 2010, pp. 2578–2585. 1

[45] S. Saha. (2018). A comprehensive guide to convolutional neural networks, [Online]. Available: `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53` (visited on 06/24/2019). 9

[46] 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*, vol. 31, no. 5, pp. 824–840, 2009. 4, 15, 35

[47] T. Silva. (2018). An intuitive introduction to generative adversarial networks (gans), [Online]. Available: `https://www.freecodecamp.org/news/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394/` (visited on 06/24/2019). 10

[48] N. Smolyanskiy, A. Kamenev, and S. Birchfield, "On the importance of stereo for accurate depth estimation: An efficient semi-supervised deep neural network approach," *arXiv preprint arXiv:1803.09719*, 2018. 21

[49] F. Stein, "Efficient computation of optical flow using the census transform," in *Joint Pattern Recognition Symposium*, Springer, 2004, pp. 79–86. 29, 31

[50] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, 2018, pp. 8934–8943. 20

[51] K. Tateno, F. Tombari, I. Laina, and N. Navab, "Cnn-slam: Real-time dense monocular slam with learned depth prediction," in *PProc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*, 2017, pp. 6243–6252. 1

[52] S. Team. (2018). Convolutional neural networks (cnn) step 4 - full connection, [Online]. Available: `https://www.superdatascience.com/convolutional-neural-networks-cnn-step-4-full-connection/` (visited on 07/04/2019). 6

[53] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," in *International Conference on 3D Vision (3DV'17)*, 2017. 13, 18, 38, 39, 41, 46

[54] A. S. V. (2019). Understanding activation functions in neural networks, [Online]. Available: `https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0` (visited on 06/24/2019). 8

[55] S. Valipour, "Deep learning in robotics," 2017. 5

[56] N. Wadhwa, R. Garg, D. E. Jacobs, B. E. Feldman, N. Kanazawa, R. Carroll, Y. Movshovitz-Attias, J. T. Barron, Y. Pritch, and M. Levoy, "Synthetic depth-of-field with a single-camera mobile phone," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 64, 2018. 1

[57] C. Wang, J. M. Buenaposada, R. Zhu, and S. Lucey, "Learning depth from monocular videos using direct methods," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, 2018, pp. 2022–2030. 21

[58] X. Wang, D. Fouhey, and A. Gupta, "Designing deep networks for surface normal estimation," in *Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'15)*, 2015, pp. 539–547. 17

[59] Y. Wang, Y. Yang, Z. Yang, L. Zhao, and W. Xu, "Occlusion aware unsupervised learning of optical flow," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, 2018, pp. 4884–4893. 20

[60] wikipedia. (2019). Convolutional neural network, [Online]. Available: `https://en.wikipedia.org/wiki/Convolutional_neural_network` (visited on 06/24/2019). 8

[61] J. Xie, R. Girshick, and A. Farhadi, "Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks," in *Proc. European Conference on Computer Vision (ECCV'16)*, Springer, 2016, pp. 842–857. 18

[62] D. Xu, W. Wang, H. Tang, H. Liu, N. Sebe, and E. Ricci, "Structured attention guided convolutional neural fields for monocular depth estimation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, 2018, pp. 3917–3925. 17

[63] N. Yang, R. Wang, J. Stuckler, and D. Cremers, "Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry," in *Proc. European Conference on Computer Vision (ECCV'18)*, Sep. 2018, pp. 817–833. 1, 20

[64] Z. Yin and J. Shi, "Geonet: Unsupervised learning of dense depth, optical flow and camera pose," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, vol. 2, 2018. 21

[65] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *European conference on computer vision*, Springer, 1994, pp. 151–158. 29, 30

[66] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. Reid, "Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, 2018, pp. 340–349. 20

[67] F. Zhang, V. Prisacariu, R. Yang, and P. H. Torr, "Ga-net: Guided aggregation net for end-to-end stereo matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 185–194. 10

[68] Y. Zhang and T. Funkhouser, "Deep depth completion of a single rgb-d image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 175–185. 11

[69] C. Zheng, T.-J. Cham, and J. Cai, "T2net: Synthetic-to-realistic translation for solving single-image depth estimation tasks," in *Proc. European Conference on Computer Vision (ECCV'18)*, Sep. 2018. 17

[70] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*, 2017, pp. 1851–1858. 10, 19, 20, 22

[71] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*, 2017, pp. 2223–2232. 17

[72] Y. Zou, Z. Luo, and J.-B. Huang, "Df-net: Unsupervised joint learning of depth and flow using cross-task consistency," in *Proc. European Conference on Computer Vision (ECCV'18)*, Sep. 2018. 19