DOES ESTIMATED DEPTH HELP OBJECT DETECTION?

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

BEDRETTİN ÇETİNKAYA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

AUGUST 2019

Approval of the thesis:

DOES ESTIMATED DEPTH HELP OBJECT DETECTION?

submitted by **BEDRETTİN ÇETİNKAYA** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oguztüzün
Head of Department, **Computer Engineering** _____

Assist. Prof. Dr. Emre Akbaş
Supervisor, **Computer Engineering, METU** _____

Assoc. Prof. Dr. Sinan Kalkan
Co-supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Assist. Prof. Dr. Gökberk Cinbiş
Computer Engineering, METU _____

Assist. Prof. Dr. Emre Akbaş
Computer Engineering, METU _____

Assist. Prof. Dr. Hacer Yalım Keleş
Computer Engineering, Ankara University _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Bedrettin Çetinkaya

Signature :

ABSTRACT

DOES ESTIMATED DEPTH HELP OBJECT DETECTION?

Çetinkaya, Bedrettin
M.S., Department of Computer Engineering
Supervisor: Assist. Prof. Dr. Emre Akbaş
Co-Supervisor : Assoc. Prof. Dr. Sinan Kalkan

August 2019, 74 pages

With the widespread use of RGB-D cameras, depth information has improved solutions of many computer vision problems including object detection. Object detection can exploit depth information and different encodings obtained from the depth map. Although previous works proved that depth information can be used to improve object detection results, this thesis investigates the effects of depth map to object detection from different aspects in detailed experiments. To clarify these effects, we examine the following three questions: (i) Should depth be used in its raw form or should it be processed to obtain different encodings and color spaces? (ii) How and when should the depth information be integrated into the object detection pipeline? (iii) how does estimated depth affect object detection results? In addition, we propose a novel method to integrate depth features into the processing pipeline of a modern two-stage object detector. Compared to previous methods, our method produces better results and uses fewer parameters. In this thesis, we also explore new loss functions to better handle the consistency between RGB and depth discontinuities. We proposed both hand-crafted and learning-based loss functions which we call the" bound loss". Using the bound loss, we were able

to improve the mean average and absolute errors for depth estimation.

Keywords: Deep Learning, Convolutional Neural Network, RGB-D Object Detection, Depth Estimation

ÖZ

TAHMİNİ DERİNLİK HARİTALARI NESNE TANIMLAMAYA YARDIMCI OLUR MU?

Çetinkaya, Bedrettin
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi Emre Akbaş
Ortak Tez Yöneticisi : Doç. Dr. Sinan Kalkan

Ağustos 2019, 74 sayfa

RGB-D kameraların kullanımının yaygınlaşmasıyla birlikte, derinlik bilgisi nesne tanımlama da dahil olmak üzere birçok bilgisayarlı görü problemlerinin çözümlerini iyileştirmektedir. Nesne tanımlama, derinlik bilgisinden ve derinlik bilgisinden elde edilen farklı kodlamalardan yararlanabilir. Her ne kadar önceki çalışmalar, derinlik bilgisinin nesne tanımlama sonuçlarını iyileştirmek için kullanılabileceğini kanıtlasa da, bu tez derinlik haritalarının nesne tanımlaya etkilerini farklı açılardan detaylı deneylerde inceler. Bu etkileri açığa kavuşturmak için, aşağıdaki üç soruyu çalıştık: (i) Derinlik bilgisi ham haliyle mi kullanılmalıdır ya da farklı kodlamaları ve renk alanlarını elde edecek şekilde işlenmeli midir? (ii) Derinlik bilgisi nesne tanımlama hattına ne zaman ve nasıl entegre edilmelidir? (iii) Tahmin edilen derinlik haritaları, nesne tanımlama sonuçlarını nasıl etkiler? Bunun yanında, derinlik bilgisini modern iki aşamalı nesne dedektör yapısına entegre eden yeni bir metot öneriyoruz. Önceki çalışmalarla kıyaslandığında, bizim yöntemimiz daha iyi sonuçlar vermektedir ve daha az sayıda parametre kullanmaktadır. Bu tezde ayrıca RGB ve derinlik kesikliklerininin tutarlılığını daha iyi işleyen yeni bir yitim fonk-

siyonu keşfettik. Sınır yitimi adını verdiğimiz el yapımı ve öğrenme tabanlı yitim fonksiyonları önerdik. Sınır yitimini kullanarak genel ortalama ve kesin hatalarını derinlik tahmin etme için iyileştirebildik.

Anahtar Kelimeler: Derin Öğrenme, Evrişimsel Sinir Ağları, RGB-D Nesne Tanımlama, Derinlik Tahmin Etme

Dedicated to my wife, family and friends.

ACKNOWLEDGMENTS

First of all, I would like to thank my supervisors, Prof. Dr. Emre Akbaş and Prof. Dr. Sinan Kalkan for their great interest and supports.

I am also grateful to my wife Dilan, my mother Suna, my father Fırat and my sister Rumeysa for their support.

This work is partially supported by the Scientific and Technological Research Council of Turkey under Grant No. 117E054.

The numerical calculations reported in this paper were fully/partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources).

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

SIFT	Scale-Invariant Feature Transform
DoG	Difference of Gaussian
HOG	Histogram of Oriented Gradients
HHA	Horizontal Disparity, Height, Angle
DHA	Depth, Height, Angle
SVM	Support Vector Machine
CNN	Convolutional Neural Network
RGB	Red, Green, Blue
RGB-D	Red, Green, Blue, Depth
mAP	mean Average Precision
MS-COCO	Microsoft COCO Object Datasets
YOLO	You Look Only Once
ROI	Region of İnterest
SSD	Single Shot Detection
MVS	Multi-view Stereo
YOLO	You Look Only Once
GAN	Generative Adverserial Network
CRF	Conditional Random Field
FPN	Feature Pyramid Network
ReLU	Rectified Linear Unit
UCM	Ultrametric Contour Map
DEN	Depth Estimation Network
RPN	Region Proposal Network
GPU	Graphics Processing Unit

CPU	Central Processing Unit
RAM	Random Access Memory

CHAPTER 1

INTRODUCTION

With technological advances, computers have been able to do many tasks that humans could do. Ordinary tasks that seem simple to humans such as object classification/detection/tracking, event detection, modeling objects, etc. are much more challenging for computers, because humans have a complex neurological system that contains millions of specialized neurons and almost never-ending learning process. On the other hand, computers have limited physical such as rams, GPUs and CPUs and virtual resources like algorithms than humans. Despite all these challenges, the computer vision field adapts to the tasks that human beings can do to computers. When these tasks are automated by computers, life becomes easier and the standard of living can increase enormously. Self-driving cars and detecting medical diagnosis are some of the examples that increase the quality of life. Both examples require object detection systems that can directly affect humans' life. In other words, improving object detection systems and creating more accurate object detectors can be vital, when human life is concerned. Even though other object detection systems don't have a vital effect on human life, unlike the previous examples, they have a large impact on daily life. For example, plant/animal diagnosis and face detection systems can ease the humans' life. Therefore, the problem of object detection has been gained the attention of researchers for many years.

While the pioneer solutions for object detections relied on hand-crafted algorithms, machine/deep learning-based solutions have become more dominant on this problem. Convolutional neural networks are one of most used machine learning approaches for object detection. CNN has an excellent learning capability and what

it learns is strongly related to provided inputs. This means that different types of input can generate more powerful neural networks. In this thesis, we investigate the CNN-based object detection solutions for different input types such as RGB, depth map and HHA encoding.

1.1 Motivation and Problem Definition

Over the past few years, RGB-D cameras that are able to capture the color and depth information simultaneously have become more affordable and available. These cameras have enabled researchers to use both sources of information for various problems such as scene labeling [38], object recognition [29], odometry estimation [26], depth and surface normal estimation [6] and etc. Object detection is one of the problems that can use depth information and it is well-known that the depth information, when combined with the color information, helps improve detection accuracy over baseline models that use the only color [19, 17, 3]. What is lesser known is whether depth, as estimated from the color image itself would improve accuracy and how to integrate it into the detection pipeline. This question becomes more important as ground-truth depth information is not always available in object detection datasets, e.g. widely used PASCAL VOC 2007 [9], 2012 [10], MS-COCO [32] datasets do not have it. With the recent developments of very successful single-image depth estimation methods [15, 30, 21, 51], the use of estimated depth for object detection has become more interesting. In this thesis, we explore different aspects of adding estimated depth information into the object detection pipeline. Specifically, we study the following questions: (i) should depth be used as it is or is it more useful when transformed into various other modalities such as horizontal disparity, height above ground, surface normal, etc. (ii) how and when should the depth information be integrated into the object detection pipeline? Also, we try to enhance the estimated depth map using a new gradient-based loss function. This loss function checks relations between the RGB image gradient and the estimated depth map gradient and prevents large differences in the estimated depth map. Table 1.1 explains penalized and non-penalized cases.

Table 1.1: Bound loss penalizing logic. ✗ means non-penalizing, ✓ means penalizing. ↑ means a large pixel value and ↓ means a small pixel value.

Pixels on RGB Gradient	Pixels on Depth Gradient	Penalizing
↑	↑	✗
↑	↓	✗
↓	↑	✓
↓	↓	✗

1.2 Proposed Methods and Models

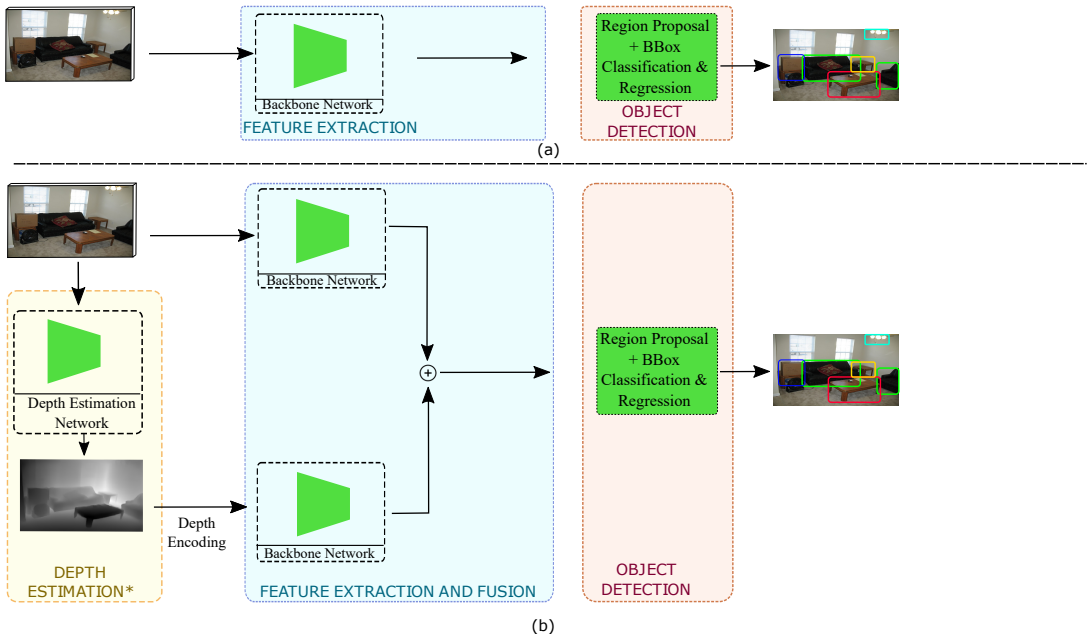


Figure 1.1: Case a shows overview of the Faster R-CNN and case b shows an overview of our proposed architecture.

Our work is not the first to integrate estimated/ground-truth depth into object detection. However, to the best of our knowledge, ours presents the most comprehensive exploration of the problem. Previous work studied the problem in a limited way, i.e. they integrated depth only at a later stage and they increased the number of base object detector parameters at least twice. In contrast, our model

(see Figure 1.1 case b) which we obtained after thorough experimentation, combines RGB and depth/HHA features at an early stage (i.e. right after the backbone network) which yields better detection accuracy with a smaller number of parameter compared to previous works. To explore the architecture in figure 1.1 case b, we detected potential concatenation points that allow integrating the depth map to RGB images and we carried out experiments for each point. After choosing the best architecture, we examine how depth map types, which are ground-truth and estimated, and different depth encodings affect object detection results. These encodings include depth map in gray-scale and jet color space and HHA representation which encodes horizontal disparity, height from the ground and angle of the surface normal with the direction of gravity by using depth map. We additionally show that even if camera parameters are unknown, HHA representation can be obtained from the estimated depth and that it improves detection accuracy. After all of these experiments, we evaluate our method on different object detection datasets including PASCAL VOC 2007, NYU-D2 and SUN RGB-D. Also, we test different DENs and choose the best one, because the estimated depth map strongly influences HHA encoding quality. For chosen DEN, we also train it with different combination of indoor and outdoor scenes.

Also, the gradient-based loss function was used in previous depth estimation works [21, 6, 46]. However, they did not consider a relation between RGB and depth map gradient changes. By considering this relation, we propose a new loss function which we call the" bound loss". Bound loss penalizes discrepancies between RGB and depth map gradients. If a pixel on RGB gradients has large value and its matching pixel on depth gradients has small value, then bound loss penalizes this inconsistency based on their differences. The main purpose is to make RGB and depth compatible. Theoretically, pixels on RGB and depth gradients should be matches. However, depth sensors such as Kinect v2 and Real Sense 1 may generate depth maps that have pixels with small shifting errors on object boundaries. In other words, generated depth maps may have alignment problems with RGB images and this problem decreases the effectiveness of bound loss.For this reason, the contribution of bound loss was limited. To overcome this problem, we designed a learning-based depth estimation module. Similar to bound loss, this module aims

to estimate a compatible depth map with RGB image. While bound loss penalizes estimated depth maps at a pixel level, the proposed learning-based module penalizes them at a patch level. We train this model on a set of matched and unmatched pairs of RGB and depth maps. Specifically, a matched-pair means that RGB patch and the depth patch come from the same spatial location. An unmatched-pair means that they come from random, unmatched (different) spatial locations. In this way, we expected to solve the alignment problem. However, the neural network architecture that we used were not as successful as we expected in discriminating matched and unmatched patches.

1.3 Contributions and Novelties

Our contributions are as follows:

- We experiment with several state-of-the-art single image depth estimation methods and investigate which estimator helps object detection the most.
- We propose a simple way of integrating depth into a two-stage object detection pipeline, which outperforms the previous methods while using a fewer number of parameters.
- We provide thorough experimentation on whether/how to process and integrate depth into the object detection pipeline. From our experiments, we conclude that depth should be concatenated at just before the RPN layer or the bounding box and classification layers.
- We propose novel loss functions to train the depth estimator, which improves the results both qualitatively and quantitatively.

1.4 The Outline of the Thesis

In chapter 2, we provide related work and necessary background on object detection and depth estimation problems.

Chapter 3 presents the details of the proposed object detection method/architecture and bound loss. It includes the architecture of our method, depth estimation network used in this work, different types of depth encoding, details of bound loss and its integration to other loss functions.
Chapter 4 presents the results of experiments These experiments composed of four parts: Architecture experiments, depth estimator network experiments, depth estimator training set experiments and datasets experiments for mAP performance.
Chapter 5 provides a summary of our work and discussion.

CHAPTER 2

RELATED WORK AND BACKGROUND

In this chapter, we provide the necessary information and review related work about the object detection and depth map estimation problems.

2.1 Introduction

Even if RGB-D cameras are widely used in many tasks such as depth map generation [21, 6, 2], ground-plane detection [49], human pose estimation [41, 53], object tracking [35] and etc., there are only a small number of datasets that published with RGB images and depth map together. Collecting RGB images and depth maps requires great effort and in case of a missing depth map of RGB images can restrict training data for the above-mentioned topics. Due to these reasons, depth map estimation from a single RGB image is one of the most important problems in computer vision. With the advance of deep learning, the depth map can be estimated from single RGB images successfully.

Also, object detection is one of the widely studied problems in computer vision and one of the central problems that need to be addressed in a wide spectrum of applications. It requires to overcome many challenges like object occlusions, different scene light conditions, different types of noises and blur. Also, it incorporates many tasks like localization, classification, scalability, etc. that are shared across many different problems in Computer Vision.

The object detection problem is composed of two tasks: localization and classification. The localization task requires tightly covering all pixels of each object a

bounding box. Classification task requires estimating which class these bounding boxes belong to. Figure 2.1 illustrates these tasks.

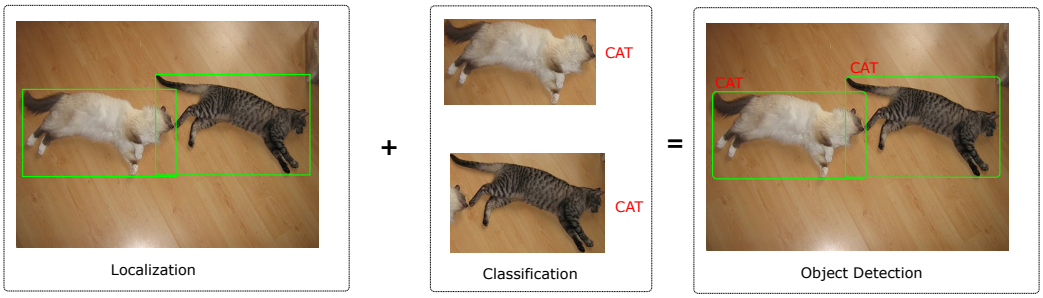


Figure 2.1: The two tasks in object detection. Image is taken from Pascal VOC 2007.

There are different approaches for object detection and we can divide it into two main groups which are RGB and RGB-D object detection. Most of the object detectors try to detect objects on RGB images, only a few of them use depth/HHA information together with RGB images. RGB detectors can be divided into two approaches, which are "classical" that rely on hand-crafted features and "neural" that employ convolutional neural networks. Also, neural approaches can be divided into two groups which are two-stage and single-stage detectors. This taxonomy can be shown in Figure 2.2. We review RGB detection methods in Section 2.3 and RGB-D methods in Section .

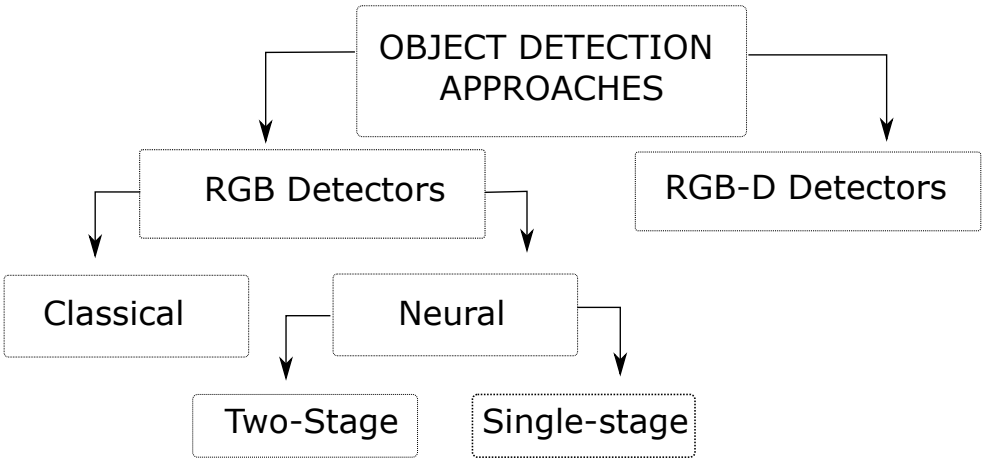


Figure 2.2: Taxonomy of object detection approaches.

2.2 Depth Map Generation

2.2.1 Depth Sensors

Different types of depth sensors are available for different tasks. Even though each sensor has different characteristics, most popular sensors are as follows:

- Microsoft Kinect [50]
- Intel RealSense [27]
- Asus Xtion

Different versions of these sensors are also available, such as v1, v2 for Microsoft Kinect and D400, T265 for Intel RealSense. Even if each sensor uses its own method for generating the depth map, we divided algorithms which are used to generate depth map in sensors into two groups. The first one is that using IR pattern to project each pixel to a 2D environment, then applying stereo matching algorithms. The second method is using time-of-flight (ToF). It aims to find the distance between the camera and each pixel of the scene using round trip time of the light signal. These methods provide raw depth and a raw depth map can contain holes or missing depth pixels because of shadowing or occlusions. To enhance raw depth, post-processing methods like edge/boundary preserving, hole filling are applied.

Song et al. [42] provides a detailed comparison between these sensors.

2.2.2 Estimation From a Single RGB Image

We can review depth estimation methods in two groups: (i) Unsupervised and (ii) Supervised learning methods.

2.2.2.1 Unsupervised Learning Methods

Unsupervised learning-based depth estimation models use stereo pair images or monocular video frames. Each method uses a different combination of objective

functions.The following methods are an example of the state-of-the-art unsupervised depth estimation models.

Gard et al. [12] proposed a type of auto-encoder architecture. Their method takes stereo image pairs. The left image is used for encoding the predicted inverse depth map and the right image is used for decoding warped RGB image output. Then, using the left image and warped RGB image, it calculates a reconstruction error.

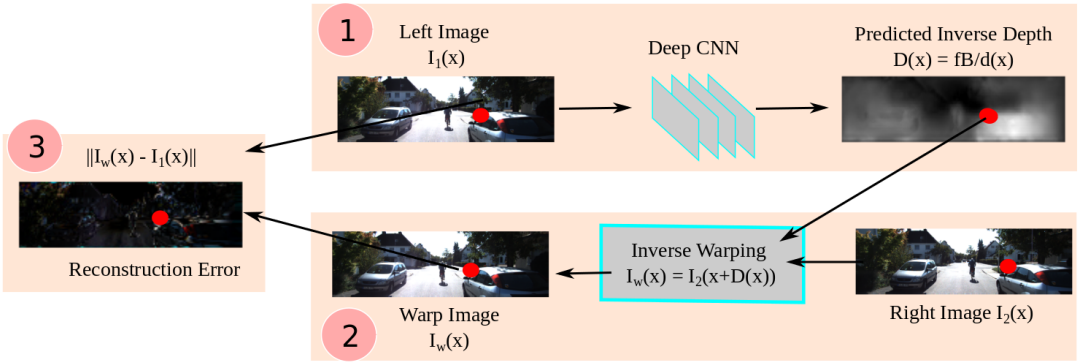


Figure 2.3: Garg et al. method's pipeline. Image is taken from [12].

Figure 2.3 shows their methods pipeline.

Godard et al. [15] uses epipolar geometry constraints and enforces the neural network for producing consistent disparity map from left and right images. For this purpose, they combine three different loss functions, respectively, appearance matching loss, disparity smoothness loss, left-right disparity consistency loss. Following formulas show their loss functions:

Final Loss:

$$C_{ds}^{l} = \frac{1}{N}\sum_{i,j}\left|\partial_x d_{ij}^{l}\right| e^{-\left\|\partial_x I_{ij}^{l}\right\|} + \left|\partial_y d_{ij}^{l}\right| e^{-\left\|\partial_y I_{ij}^{l}\right\|} \tag{1}$$

where l is a left image, r is a right image and α is a coefficent.

Apperance Matching Loss:

$$C_{ap}^{l} = \frac{1}{N}\sum_{i,j}\alpha\frac{1-\mathrm{SSIM}\left(I_{ij}^{l},\tilde{I}_{ij}^{l}\right)}{2} + (1-\alpha)\left\|I_{ij}^{l}-\tilde{I}_{ij}^{l}\right\| \tag{1.1}$$

,where $SSIM$ is Stuructural Similarity Index proposed by Wang et al. [47],I_{ij}^{l} is

input image pixel and $\tilde{I}^l_{ij}$ is its reconstruction.

Disparity Smoothness Loss:

$$C^l_{ds} = \frac{1}{N}\sum_{i,j}\left|\partial_x d^l_{ij}\right| e^{-\left\|\partial_x I^l_{ij}\right\|} + \left|\partial_y d^l_{ij}\right| e^{-\left\|\partial_y I^l_{ij}\right\|} \tag{1.2}$$

,where ∂_x and ∂_y are image gradients, d^l_{ij} is depth map pixel.

Left-Right Disparity Consistency Loss:

$$C^l_{lr} = \frac{1}{N}\sum_{i,j}\left|d^l_{ij} - d^r_{ij+d^l_{ij}}\right| \tag{1.3}$$

,where d^l_{ij} is disparity map pixel.

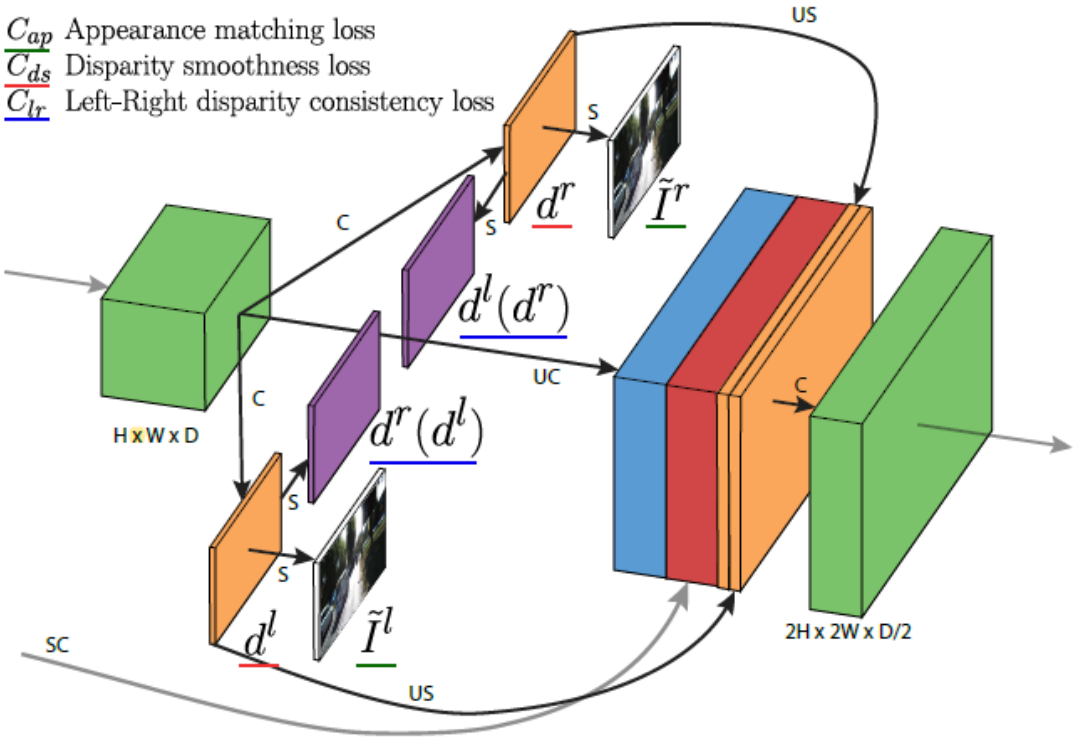


Figure 2.4: Godard et al. loss functions combination. Image is taken from [15].

Figure 2.4 shows used loss functions and their combinations as a loss module.This same module is repeated at each of the four different output scales. C: Convolution, UC: Up-Convolution, S: Bilinear Sampling, US: Up-Sampling, SC: Skip Connection Image is taken from [15].

Aleotti et al. [2] propose a generative adversarial network (GAN) to estimate the depth map. MonoGan consists of two modules, generator, and discriminator. While a generator produces depth maps and warps images, discriminator classify images as real or fake. In this way, the generator tries to fool the discriminator by learning a better-produced depth map and discriminator learns to improve its ability. Figure 2.5 shows the MonoGan pipeline.

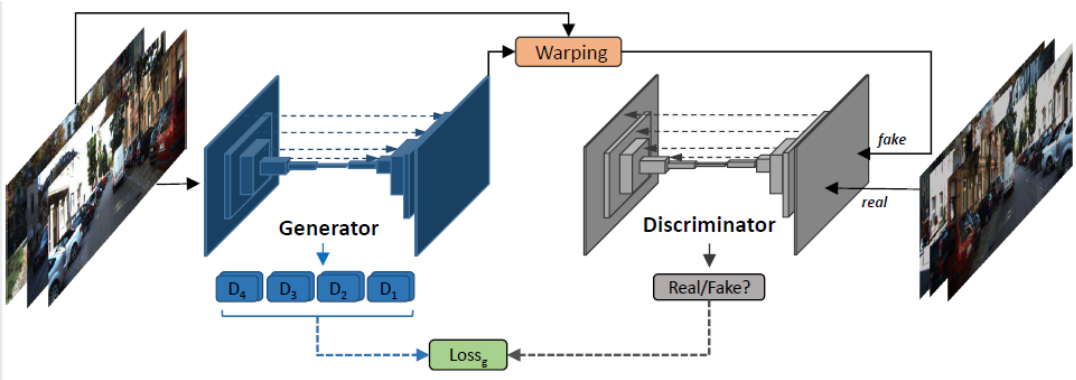


Figure 2.5: MonoGan pipelines. Image is taken from [2].

Zhou et al. [52] propose a joint training method for depth map and camera pose estimation from unstructured monocular video sequences. Even if they trained both tasks together, at the inference each task can be run separately. Their network takes three inputs frame $I_{(x-1)}$, I_x and $I_{(x+1)}$. Depth network only uses I_x frame and pose networks uses all three frame. Then, outputs of the two networks are used to reconstruct the target view by employed photometric reconstruction loss. Figure 2.6 shows this method of overview.

Kendall et al. [25] proposes an architecture that process stereo pair images and aims to learn 3d cost volume. Their network can learn context information using 3-D convolutions and deconvolutions operation applied to the produced cost volume. They used the soft argmin cost function, which is differentiable and enables regressing sub-pixel disparity values from 3d cost volume. Figure 2.7 shows their method overview.

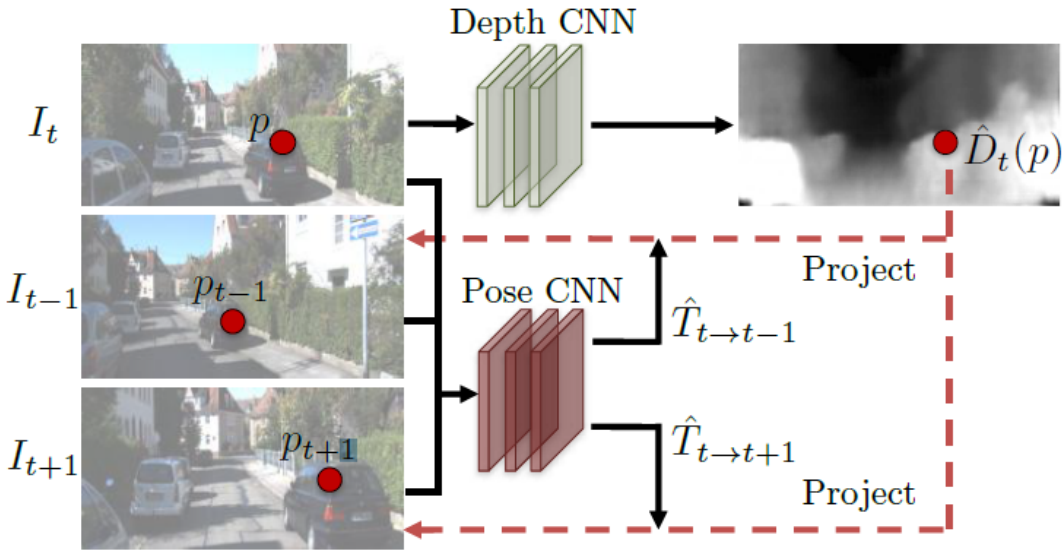


Figure 2.6: Zhou et al. method pipeline. Image is taken from [52].

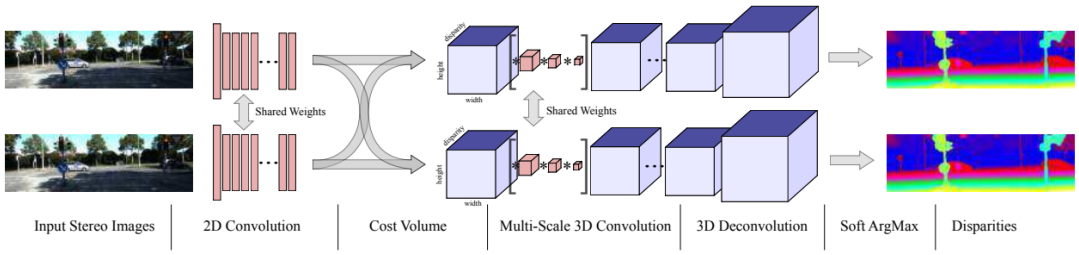
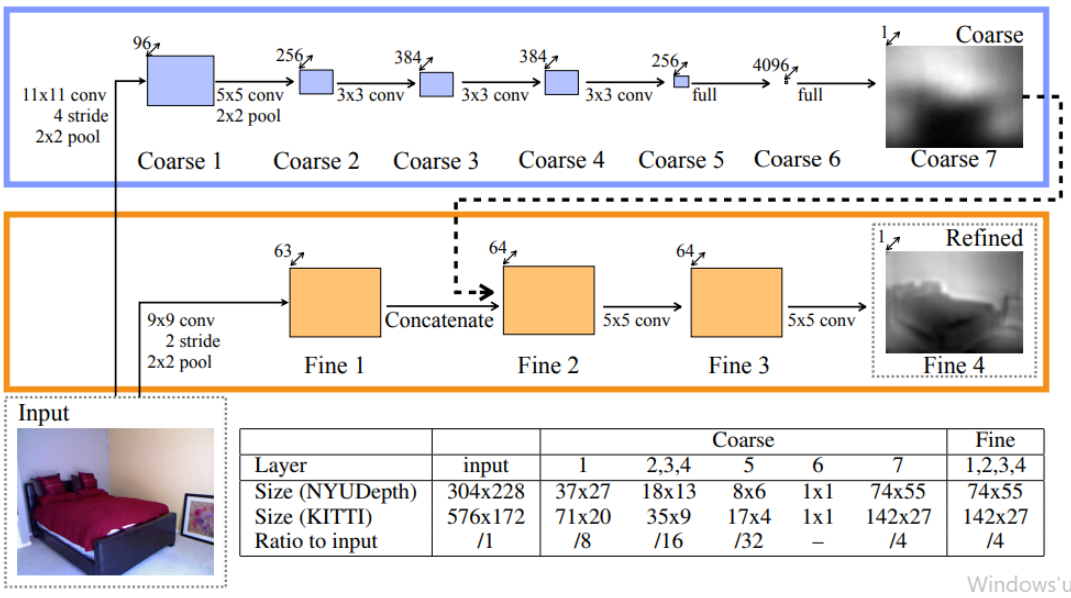


Figure 2.7: Kendall et al. method pipeline. Image is taken from [25].

2.2.2.2 Supervised Learning Methods

Eigen et al. [7] propose an architecture that composes two modules. The first module predicts the depth map at the global level and the second module is used to refine the first module output at the local level. Their method is based on regression of depth map and they used a scale-invariant loss to prevent scale-depended errors. Figure 2.8 shows their method of architecture.

Liu et al. [33] treated the depth estimation problem as a continuous conditional random field (CRF) learning problem. Their method composes of two parts which are unary and pairwise, respectively. After images are segmented into superpixels, the unary network takes these resized superpixels.Then, the neighborhood of superpixel which is the input of unary network are determined and similarities be-



		Coarse					Fine
Layer	input	1	2,3,4	5	6	7	1,2,3,4
Size (NYUDepth)	304x228	37x27	18x13	8x6	1x1	74x55	74x55
Size (KITTI)	576x172	71x20	35x9	17x4	1x1	142x27	142x27
Ratio to input	/1	/8	/16	/32	–	/4	/4

Figure 2.8: Eigen et al. method pipeline. Image is taken from [7].

tween neighborhood superpixels and selected superpixels are found. These similarities are an input of the pairwise network. Finally, negative log-likelihood is calculated by using outputs of two networks. Figure 2.9 shows their method overview.

Li et al. [30] propose a method to create virtual unlimited depth datasets. They use state-of-the-art structure from motion (Sfm) and multi-view stereo methods (MVS) to create depth dataset which is called MegaDepth dataset. Also, using semantic segmentation produced depth datasets are enhanced. After these steps, they train the depth estimation network using huge depth data.

Junjie Hu et al. [21] propose a network that focuses on object boundaries. Their network consists of four parts: encoder, decoder, multi-scale feature fusion, and refinement module. Also, they use a combination of three different loss functions. The first one is called depth loss that penalizes the difference between ground truth and estimated depth data. The second one is called gradient losses that penalize error around edges. The final loss is called normal loss that penalizes difference normals to scene surface between ground-truth and estimated depth map. Figure 2.10 shows their architecture overview.

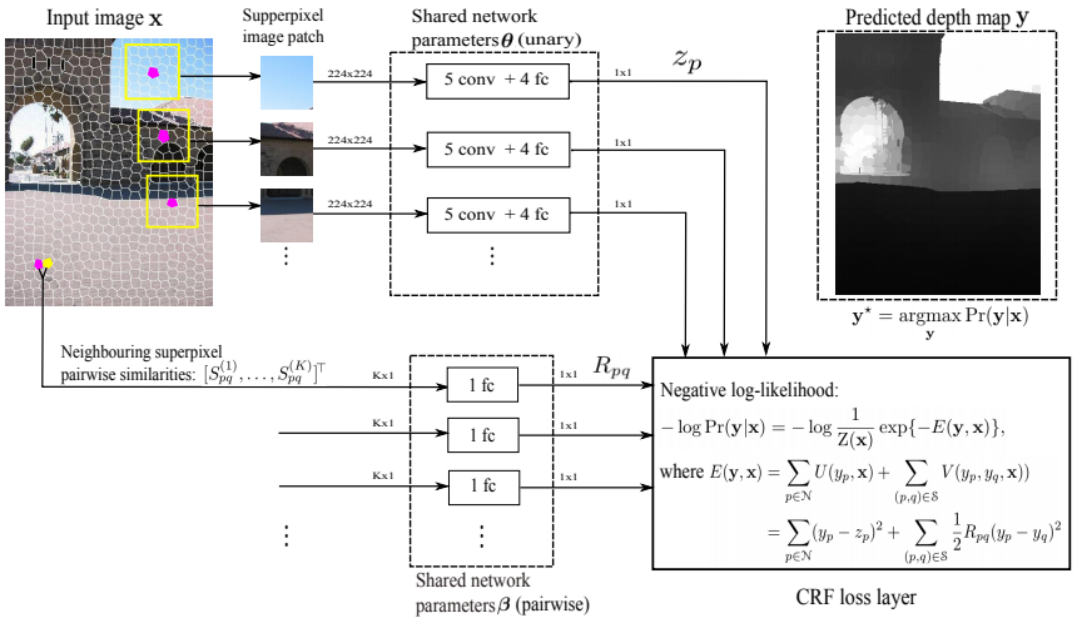


Figure 2.9: Liu et al. method pipeline. Image is taken from [33].

2.3 RGB Object Detection

2.3.1 Classical Approaches

Classical approaches to object detection [34, 5] included two stages: (i) Region selection or proposal, and (ii) Object classification. For region selection, a window is slid over the image, or "interesting regions" are identified in the image. These windows or regions are assumed to be likely to include an object or objects in them. For object classification, classical approaches used hand-designed features extracted from the selected region and mapped those features to one of the object categories. As features, many alternatives such as SIFT [34], HOG [5] and for classification, methods like SVM [4], AdaBoost [11] have been used successfully.

SIFT: Scale-Invariant Feature Transform (SIFT) tries to find feature points of images, then computes its descriptor using these points. It composes of following four main parts: (i) Feature point detections, (ii) Feature points localization, (iii) Orientation assignment and (iv) Feature descriptor generation.

By using the Difference of Gaussian (DoG) detector, subpixels local maxima or minima are detected and these extrema points are localized by using Taylor series

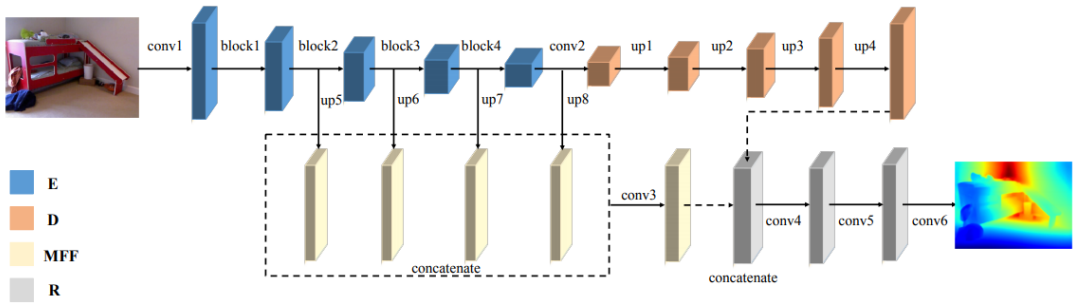


Figure 2.10: Li et al. method pipeline. E shows encoder module, D shows decoder module, MFF shows multi-scale feature fusion module, R shows refinement module. Image is taken from [21].

expansion. Then, directions and magnitudes are assigned to each extrema points by taking gradients of the region that cover the neighborhood of these points. Finally, the descriptor is generated from key-points neighborhood orientation histogram.

HOG: Histogram of Oriented Gradients(HOG), provides key-points and descriptors to detect objects. It divides images to the grid and computes the direction of the gradients of each grid cell. In this way, pixels that have smaller orientation, are eliminated and pixels that have greater orientation become dominant.

The above-mentioned algorithms, SIFT and HOG, only generate images key-points. To determine which key-points represent which categories/classes, classifiers are needed. Support Vector Machine (SVM) [1] is one of the successful classifiers. It provides separation of features by determining the best hyperplane with maximum margin.

2.3.2 Neural Approaches

With advances in deep learning, neural approaches have been applied to object detection with superior performances. It can be divided into two categories, 2.3.2.1 and 2.3.2.2 according to using region proposal network.

2.3.2.1 Two-Stage Detectors

This approach consists of two stages which are region proposal and classifying selected region into objects, respectively. The following models are examples of state-of-the-art two-stage object detectors.

- R - CNN
- Fast R-CNN
- Faster R-CNN

R-CNN: R- CNN [14] is a pioneer of this two-stage approach. To propose regions, it uses a selective search algorithm [45] that generates a region which has the highest probability of containing the object. Selective Search uses bottom-up approaches which means merging small regions to obtain larger regions. It defines initial regions, then calculates similarities between each region in terms of color, texture, size, and filling. Using this similarity information, it iteratively merges most similar regions. When images become a single region, iteration halts. This region is called the region of interests, ROI, and the typical number of ROI is 2k. After generating ROI, backbone network, AlexNet [28] extract features from this ROI. To classify each ROI, class-specific SVMs are trained and scores these extracted features. Figure 2.11 shows R-CNN pipelines.

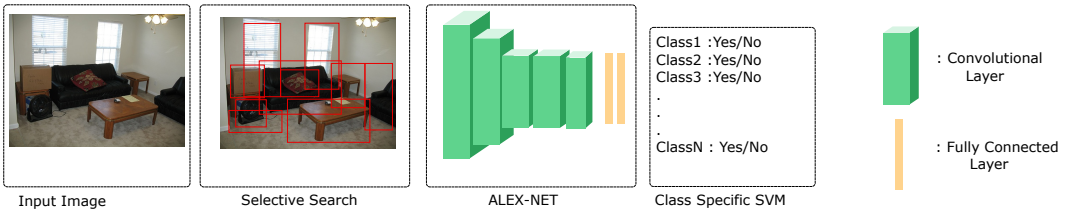


Figure 2.11: R-CNN pipeline.

Even if R-CNN performed a great revolution for object detection, It has the following disadvantages:

- Complex training procedures

- Expensive training and test times
- Non-trainable object proposal algorithm

Fast R-CNN: To overcome some of these disadvantages, Fast R-CNN [13] was proposed. Unlike R-CNN, Fast R- CNN extracted features from an input image and then each ROI is identified on these extracted features. In other words, ROIs share a convolutional features map and this operation speeds whole training procedure. Another change in Fast R-CNN is the ROI pooling layer. It performs a max-pooling operation to obtain fixes sized feature vector. As a final step, it has two outputs, respectively class score obtained from softmax function and bounding box coordinate obtained from the related fully connected layer. With all of these changes, Fast R-CNN simplified training procedure, decreased training and testing times and increased mean average precision. Figure 2.12 shows the Fast R-CNN pipelines.

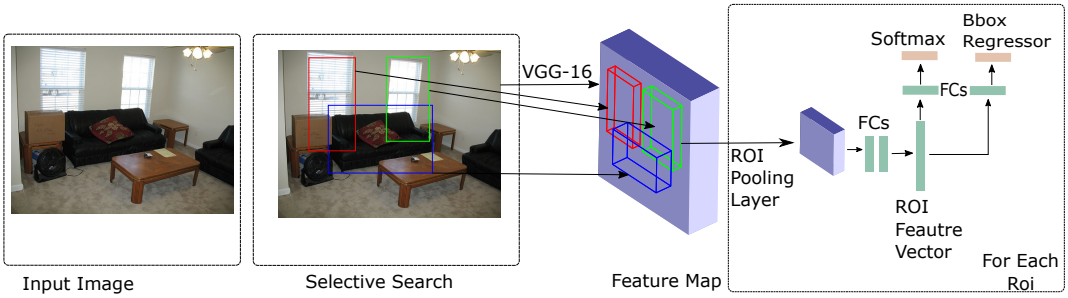


Figure 2.12: Fast R-CNN pipeline.

Faster R-CNN: Although Fast R- CNN partially solved the speed problem of R-CNN, the region proposal algorithm, selective search, still slowed down training and testing speeds. Faster R-CNN [37] solved this problem by replacing selective search with CNN based region proposals which means that ROI is the output of a trainable process. The remaining parts of Faster R-CNN are the same as Fast R-CNN. Figure 2.13 shows Faster R-CNN pipelines.

2.3.2.2 Single-Stage Detectors

Unlike two-stage detectors, they try to detect objects without the object proposal step. The following models are examples of the state-of-the-art one stage detec-

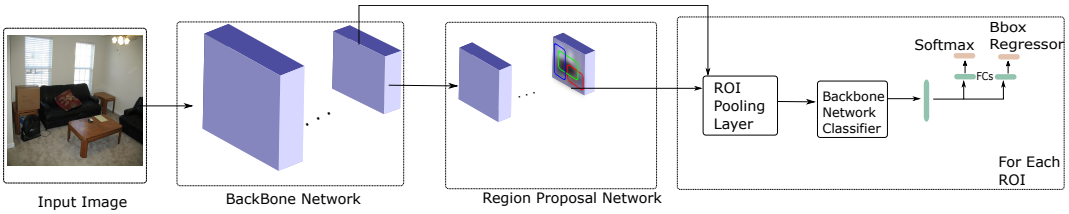


Figure 2.13: Faster R-CNN pipeline.

tors.

- YOLO
- RetinaNet
- SSD

YOLO: YOLO [36] divides an image into square grids and tries to detect the object by using these grid cells. Each grid cell can detect only one object. This rule can lead to undetectable objects for specific cases such as more than one object in one grid, overlapping and too close to each other objects, etc. Also, grid cells are responsible for predicting a fixed number of bounding boxes and probabilities for each class. In light of all this information, YOLO is more appropriate for a real-time object detection task because of its inference speed and simple pipeline. Figure 2.14 shows the YOLO pipeline.

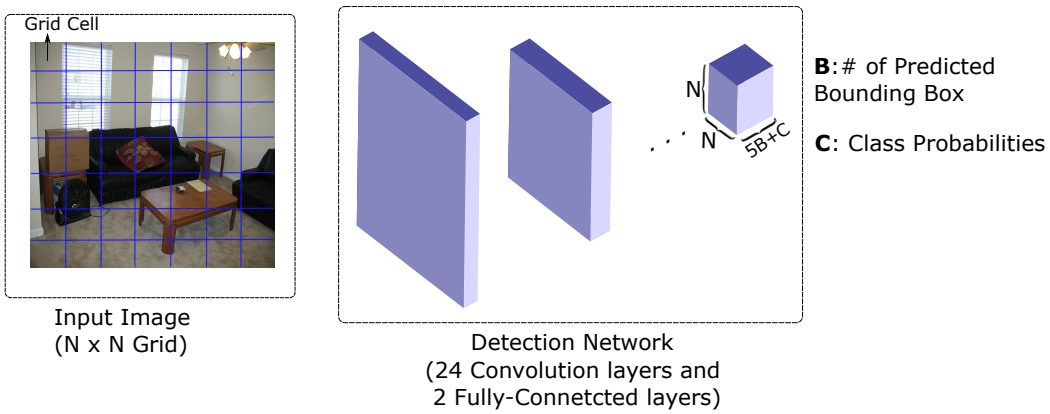


Figure 2.14: YOLO pipeline.

RetinaNet: RetinaNet [31] is a simple network that contains two main parts, backbone, and subnets. Unlike other object detection models, the backbone network is Feature Pyramid Networks [**?**]lin2017feature), called as FPN. FPN includes features at different scales and combines these features into the pyramid shape. In this way, low-level features are not only responsible for generating high-level features, but also contribute directly classification and bounding box regression task. It is shown in case b of Figure 2.15. Also, there two subnets for classification and bounding box regression tasks. Classification and Bounding Box Subnet contains four convolution layers followed by RELU, separately. Focal loss is used for classification and L1 loss is used for bounding box regression. Figure 2.15 shows the RetinaNet pipeline.

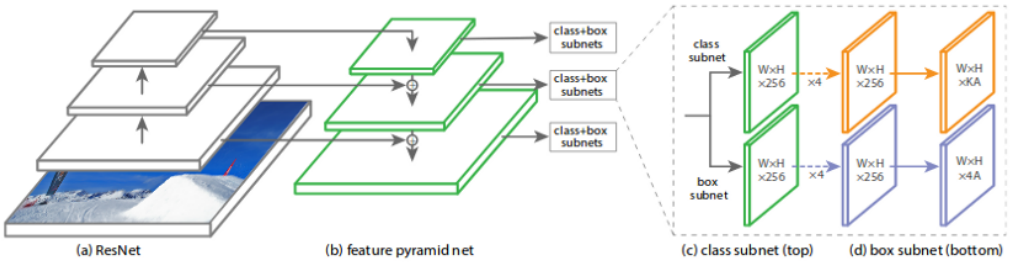


Figure 2.15: RetinaNet pipeline [31].

SSD: Single Shot Detection (SSD) is designed for real-time object detection. It consists of two parts: (i) Feature Extraction and (ii) Object Detection using the extracted feature. For extracting features from images, it uses the VGG-16 network as a backbone. Then, these features pass through five convolutional layers. The output of each layer directly connected to the final layer of the SSD network. This means it uses multi-scale feature maps. While SSD decreases inference time, it performs worse than the two-stage object detector for small scale objects.

2.4 RGB-D Object Detection

By definition and owing to perspective projection, compared to RGB images, depth information can provide more cues about the local surface structure of the objects

as well as the layout of the scene. An object detector can utilize both cues for both determining the regions likely to contain objects as well as in classifying regions into objects. This has been demonstrated by many studies.

With the introduction of R-CNN [14], many studies attempted integrating depth into deep object detectors [3, 17, 19]. The work by Gupta et al. [17] was one of the first to do so. They utilized the depth information for extending both the region proposal stage as well as the classification stage. To be specific, they used RGB-D images to compute 2.5D contours from which they estimated regions that are likely to contain objects. For the object classification stage, they first trained two CNN models for feature extraction; one for extracting features from the RGB information and one from the depth information. Then, they trained a linear SVM for classifying these features into objects. Another crucial contribution of Gupta et al. was to use horizontal disparity (D), height from the ground (H) and angle of the surface normal with the direction of gravity (A) as the input to the feature extracting CNN model. They demonstrated that this performs better than providing depth directly.

It has also been demonstrated by Cao et al. [3] that depth estimated from an RGB image can be used to improve object detection from that RGB image, without relying on any depth detector. They used Conditional Random Field to estimate the depth of the scene from the RGB image and train two independent CNN networks to classify the regions using the RGB inputs and the estimated depth input. For encoding the depth information, they directly provided the logarithm of the estimated depth and refrained from using surface normals or D, H and A encoding used by [17], claiming that such cues are not informative since (i) the estimated depth is approximate and noisy, and (ii) information about the camera is required for some of these cues.

Hou et al. [19] provided a very informative analysis of the different mechanisms for integrating depth information into a deep object detector. Namely, they investigated the importance of the different visual inputs (RGB, depth, angle, height, contour, etc.) as well as the different levels (input-level vs. feature-level) for combining RGB and depth information for object detection. They argued and demon-

strated that (i) color and depth should not be combined at the input level since they carry different types of information, and (ii) processing D, H and A separately with separate networks perform better than processing DHA together with a single network.

2.4.1 Summary of the Literature and Our Contributions

Looking at the existing studies, we see that all previous works increase the number of parameters at least twice to integrate depth/HHA encoding. Also, their base object detection network is not the state-of-the-art method.

Table 2.1 shows the common properties of RGB-D object detector networks.

Table 2.1: Method Comparison. *x* in Number of Model Parameters row represents the number of parameters in the base network. NA means "not applicable". D, H, A mean each channel of HHA encoding respectively, horizontal Disparity, Height Above Ground and Angle With Gravity. UCM means Ultrametric Contour Map. (+) means separate network input. Early in Concatenation Type column means concatenation before the RPN module. Late in Concatenation Type column means concatenation after the RPN module.

	[3]	[17]	[19]	[19]	Ours
Base Network	Fast R-CNN	R-CNN	R-CNN	Fast R-CNN	Faster R-CNN
Backbone Network	VGG-16	VGG-16	AlexNet	VGG-16	VGG-16 or ResNet-101
Number of Model Parameters	2x	2x	≥ 5x	≥ 4x	≤ 2x
Model Input Type	RGB + Gray-scale DepthMap	RGB + DHA	RGB + D + H + A + UCM	RGB + D + H + A	RGB + (Depth or DHA)
Concatenation Type	Late	Late	Early	Early	Early
Tested RGB Datasets	VOC 2007	None	None	None	VOC 2007
RGB Datasets mAP Improvement	✓	NA	NA	NA	✗
Tested RGB-D Datasets	NYUD2 B3DO	NYUD2	NYUD2 SUN RGB-D	NYUD2	NYUD2 SUN RGB-D
RGB Datasets mAP Improvement	✓	✓	✓	✓	✓

CHAPTER 3

METHOD & MODEL

In this chapter, we first describe our proposal on how to integrate depth into the common object detection pipeline. Then, we present, bound losses, our novel loss functions for training single-image depth estimation models.

3.1 Overview

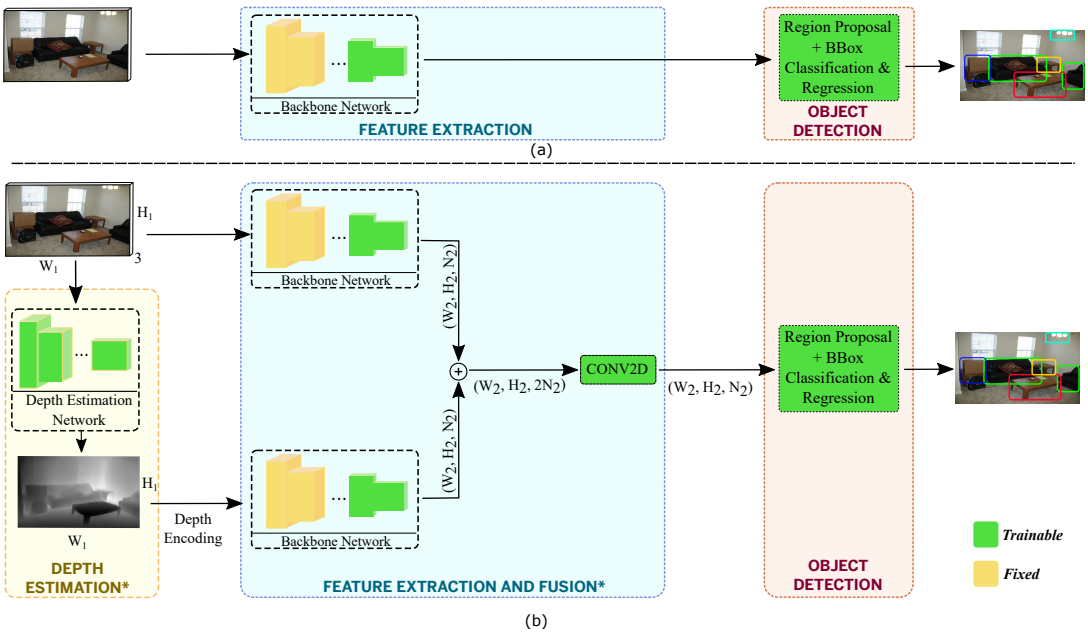


Figure 3.1: Case a shows overview of the Faster R-CNN and case b shows an overview of the proposed architecture.

As illustrated in Figure 3.1, our method is composed of three main steps:

- Depth estimation (Section 3.3).

- Extracting and fusing features from the RGB image and estimated depth (Section 3.2.4).

- Object detection from fused RGB and depth features (Section 3.2.5).

3.2 Encoding Depth

Similar to color, depth can be encoded in different ways, which significantly affect the overall performance when they are used (see, e.g. [17]). In this section, we describe the widely used depth encoding schemes which we investigated in the paper – see also Figure 3.3 for how these encodings reflect different aspects of the 3D structure.

3.2.1 Gray-scale Encoding

In this encoding, the depth values are converted to gray-scale intensity values linearly as follows:

$$g(d) = \frac{d - d_{min}}{d_{max} - d_{min}} \times 255, \tag{31}$$

where d_{min}, d_{max} refer to the minimum and maximum depth values of all the scenes in the dataset.

3.2.2 Jet Color Space Encoding

People perceive colorful images better than gray-scale images, so there are many different colormaps as seen in figure 3.2. Each colormap has different characteristics and is used for different purposes. Also, it is known that encoding depth in the jet color space yields better performance for many 3D object recognition tasks [8, 44]. This is expected since the jet color space provides a bigger range for the depth values and makes changes in the depth more distinctive (see also Figure 3.3). For conversion into the jet color space, the grayscale encoding values are used as indices for a jet colormap with 256 entries.

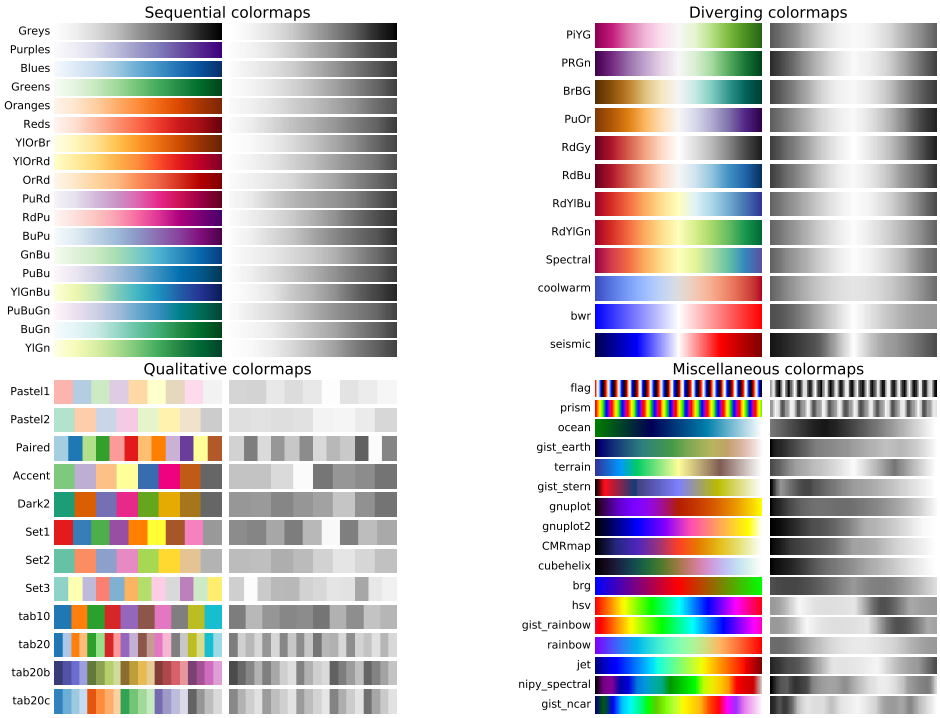


Figure 3.2: Example colormaps [22].

3.2.3 Disparity Height Angle (DHA) Encoding

Recently, it has been shown that explicitly encoding horizontal disparity (D), height (with respect to the ground – denoted with H) and angle with the vertical direction (A) yields better depth representations [17]. DHA encoding, also called HHA in the literature, can be formally defined as follows:

$$D = p_x - p_x^c, \tag{32}$$

where p_x is the x coordinate of pixel p, and p^c is the corresponding pixel in camera c.

As for H (the height from the ground), since it is tricky to obtain the knowledge about the ground, the height from the lowest point in the scene is usually taken [16].

To compute angle with gravity (A), an iterative procedure is used as in[16]: An initial estimate for the gravity direction is taken as the vertical axis, with respect to which all surface normals are clustered into surfaces that are approximately paral-

lel or orthogonal to the gravity direction. After clustering, a new gravity direction estimated with respect to the parallel and orthogonal clusters. These steps are iterated to minimize so that the gravity direction is as parallel as possible to the parallel surfaces and as orthogonal as possible to the orthogonal surfaces.

Before calculating the DHA encoding, the grayscale depth encoding is zero-centered and normalized with a standard deviation.

3.2.4 Extracting and Fusing Features from RGB and Depth Data

The RGB input (I) and the estimated depth encoding (D) are separately fed to a backbone network (X to be specific), yielding two sets of features, $\phi(I) \in \mathbb{R}^{W_2 \times H_2 \times N_2}$ and $\phi(D) \in \mathbb{R}^{W_2 \times H_2 \times N_2}$. The concatenation $\phi(I) \oplus \phi(D)$ is passed through a convolutional layer to reduce dimensionality to $W_2 \times H_2 \times N_2$.

3.2.5 Object Detection from Fused RGB-Depth Features

In this work, we use Faster R-CNN, which is one of the state-of-the-art two-stage object detectors; however, our contributions are not specific to Faster R-CNN and they can be integrated into any object detector that is based on features extracted from a backbone network. We use the region proposal network and the bounding box regression & classification stages as they are (except for the alternative architectures in the ablation experiments), and therefore, we only briefly remind these stages here.

In Faster R-CNN, the first stage, Region Proposal Network, estimates possible object regions with respect to a fixed set of bounding box configurations (called anchor boxes). The regions passing this first stage are then classified into object categories and their bounding boxes are fine-tuned with respect to the region-pooled features.

3.2.6 Preprocessing and Postprocessing

After the pre-processing step, we made small changes in the original implementation. Point cloud's z value is equal to the thresholded ground-truth depth map:

$$Z_{i,j} = \begin{cases} T, & \text{if } \sum_{i=0}^{h} \sum_{j=0}^{w} D_{i,j} < T \\ D_{i,j}, & \text{otherwise} \end{cases} \tag{33}$$

where T is threshold value and equal to 100 in the original implementation, D is depth map, Z is point cloud's z value, h is depth map height and w is depth map width. We set this threshold T value to 200 for creating meaningful images visually. Also, height above ground channel is normalized with its maximum pixel value and then scaled 0-255 range. To create HHA encoding from estimated depth maps, if the camera parameter is unknown for RGB input image, Microsoft Kinect camera parameters are used in point cloud estimation step of HHA algorithm.
Example images for this encoding are shown in the last row of figure 3.3.

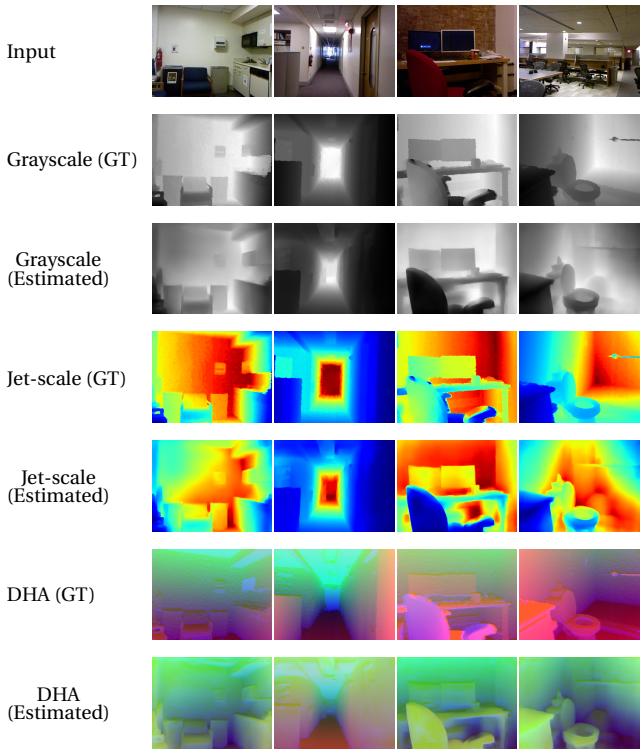


Figure 3.3: The encoding schemes considered in this paper. Images and ground truth depth maps are taken from the NYUD2 dataset [42].

3.2.7 Alternative Architectures

In addition to the architecture outlined in Section 3.1, we also considered and extensively evaluated alternative architectures (see Figure 3.4). The alternative architectures include:

- Integrating depth information just before the bounding box classification & regression networks (Figures 3.4(a) and (b)). We tested this in two ways:

    1. Directly providing depth encoding as it is, without any feature extraction (Figure 3.4(a)). To match the sizes, we first resize depth map to 64×64.

    2. Providing extracted features from depth (Figure 3.4(b)). This is achieved by three convolutional layers, each followed by ReLU non-linearity and batch normalization [23]. To arrange the sizes, the activations after these operations are resized using linear interpolation.

- Integrating depth information before the backbone network classifier (Figure 3.4(c)). This is achieved by three convolutional layers, each followed by ReLU non-linearity and batch normalization [23]. To arrange the sizes, the activations after these operations are resized using linear interpolation.

3.3 Depth Estimation Network (DEN)

In this thesis, the proposed depth estimation method by Hu et al. [21] was used and object detection experiments were done with this network outputs. Besides, we proposed the novel loss function, called bound loss, to enhance estimated depth. Similar to the approach of Cao et al. [3], we estimate depth from the RGB images and aim to improve object detection with the estimated depth. For estimating the depth map from a single RGB image, we adapt and extend the deep network proposed by Hu et al. [21]. They used a combination of three different loss functions mentioned in section 2.2.2.2. For proposed loss function,bound loss, the formula

is as follows:

$$L_b = \alpha \times |h(D)| \times \sigma(-\frac{\sqrt{h(R)^2 + h(B)^2 + h(G)^2)}}{\beta} - \theta) \tag{34}$$

where function h() is gradient operation, $\sigma()$ is sigmoid function, D is estimated depth map, R,G,B are image channels and α, β, θ are fine-tune parameters. We used Sobel operator for extracting image gradients. d_x and d_y are outputs of Sobel operator, so we have two loss functions for horizontal and vertical gradients.

First of all, to understand how proposed loss function works, ground-truth depth images were used instead of estimated depth and fine-tune parameters are chosen a small number, α = 1, β = 0.125, and θ = 1. Figure 3.5 shows a visualization of bound loss with corresponding depth and RGB images.

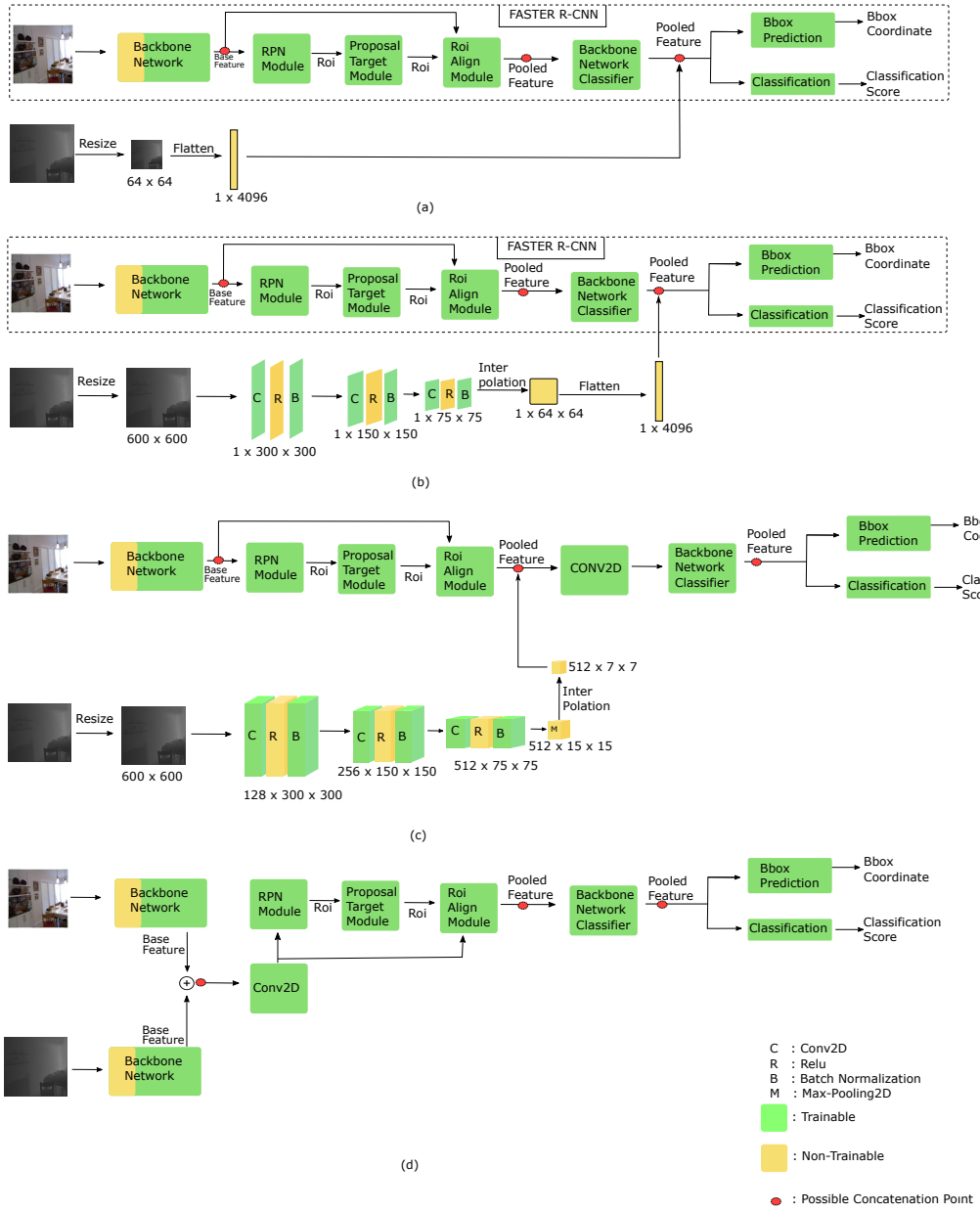


Figure 3.4: All architectures skeleton used in experiments. Case a shows raw depth map joining at Backbone Network Classifier output. Case b shows extracted depth features from small network joining with Backbone Network Classifier output. Case c shows extracted depth feature from small network joining with Roi Align Module output. Case d shows extracted depth extracted from Backbone Network joining with extracted RGB features from separate Backbone Network.

Figure 3.5: Visuzalition of bound loss function parts.

$$L_d = L_b = \alpha \times |h(D)| \times \sigma(-\frac{\sqrt{h(R)^2 + h(B)^2 + h(G)^2)}}{\beta} - \theta) \tag{35}$$

As shown in figure 3.5, bound loss aims to penalize pixels where depth and RGB gradient do not overlap. Theoretically, the expected behavior is that depth and RGB gradients overlap. However, they may not overlap, because used depth data is saved with Kinect v-2 and this depth sensor is not very sensitive especially in case of objects' border pixels. Therefore, bound loss enhances the estimated depth map less than expected. Also, another issue is the thickness of gradient pixels. Gradient pixels in depth and RGB images can be in different thickness and this can leads to penalizing of truly estimated pixels. That is why we added to fine-tune parameters and tried to get non-penalized loss visualization of ground-truth depth data which means loss function visualization consists of only black pixels.

Optimizing β and θ parameters require a lot of time if we try to optimize by hand. Instead of this, we trained differentiable parameters to find the best value of these parameters. In other words, we assigned our loss function value to a differentiable variable and tried to minimize it using the Adam optimizer. We have chosen an initial value of these parameters as 0.5 for both variables, and calculated our loss function value without α parameters. α parameter is only important when this loss function is integrated to other loss functions. Also, the optimizer could choose α parameter as zero to minimize our loss functions value. To avoid this, we eliminated this parameter to optimize our loss function parameters. For this training process, we tried to select the same settings with the actual depth estimator network. We used 5 epoch using Adam optimizer with a learning rate of 0.001. After this optimization of parameters, bound loss slightly improved the depth metrics. The results are presented in section 4.1.

Figure 3.10 shows the loss value vs epoch graph. As shown in this figure, we can minimize loss value by training differentiable parameters. Only one epoch is enough, more thane one epochs increase loss value. As shown in formula 311, the minimum value of our loss is zero and optimizing variable can reach the minimum value of our loss function at the end of the first epoch. We choose two different pairs of parameters that proximate to zero. These variables are shown in table 3.1.

There is no easy way to choose α parameter, so we tried different values. Related results are presented experiment section 4.1.

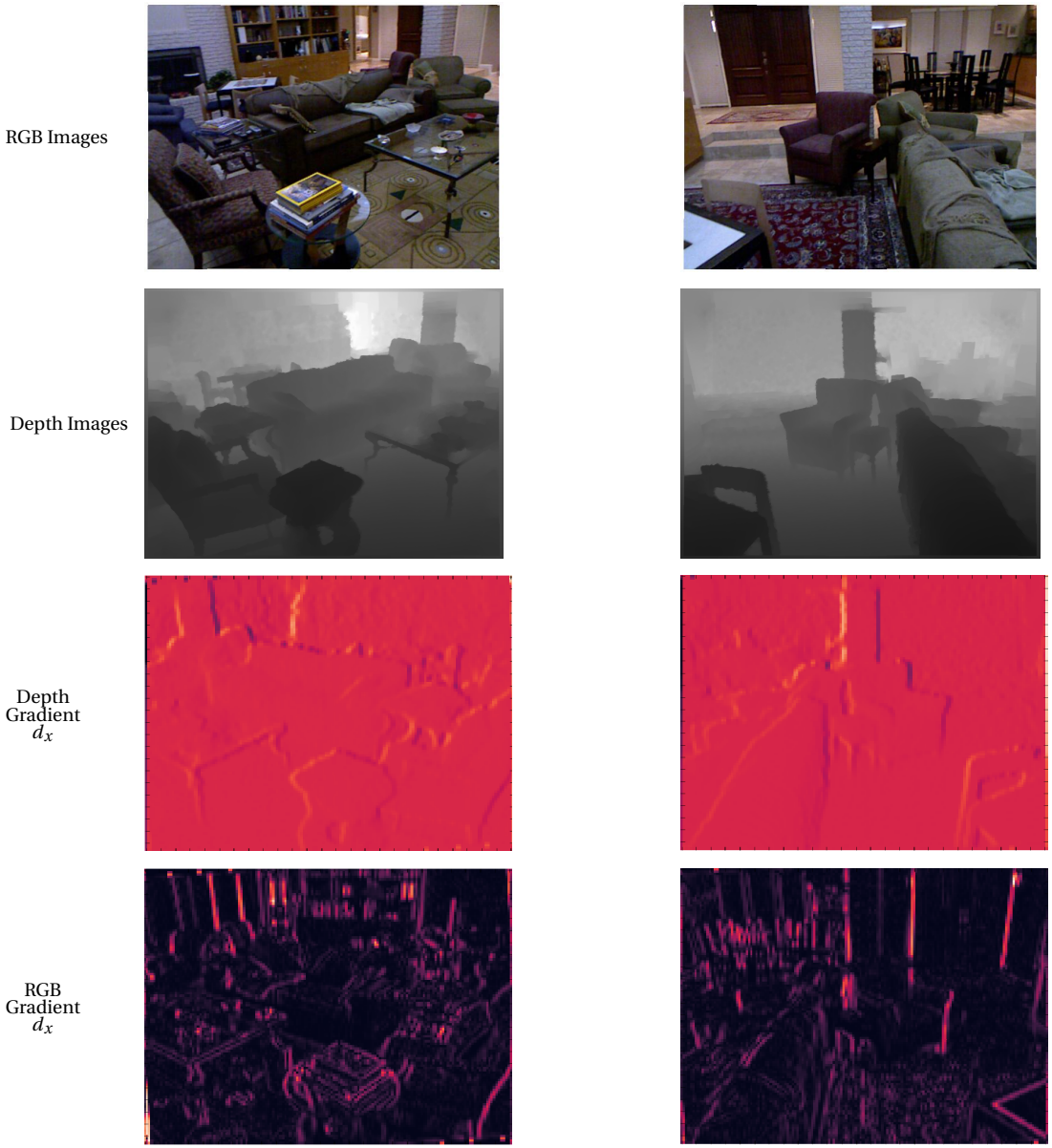


Figure 3.6: Visualization of Our Loss Function Part 1.

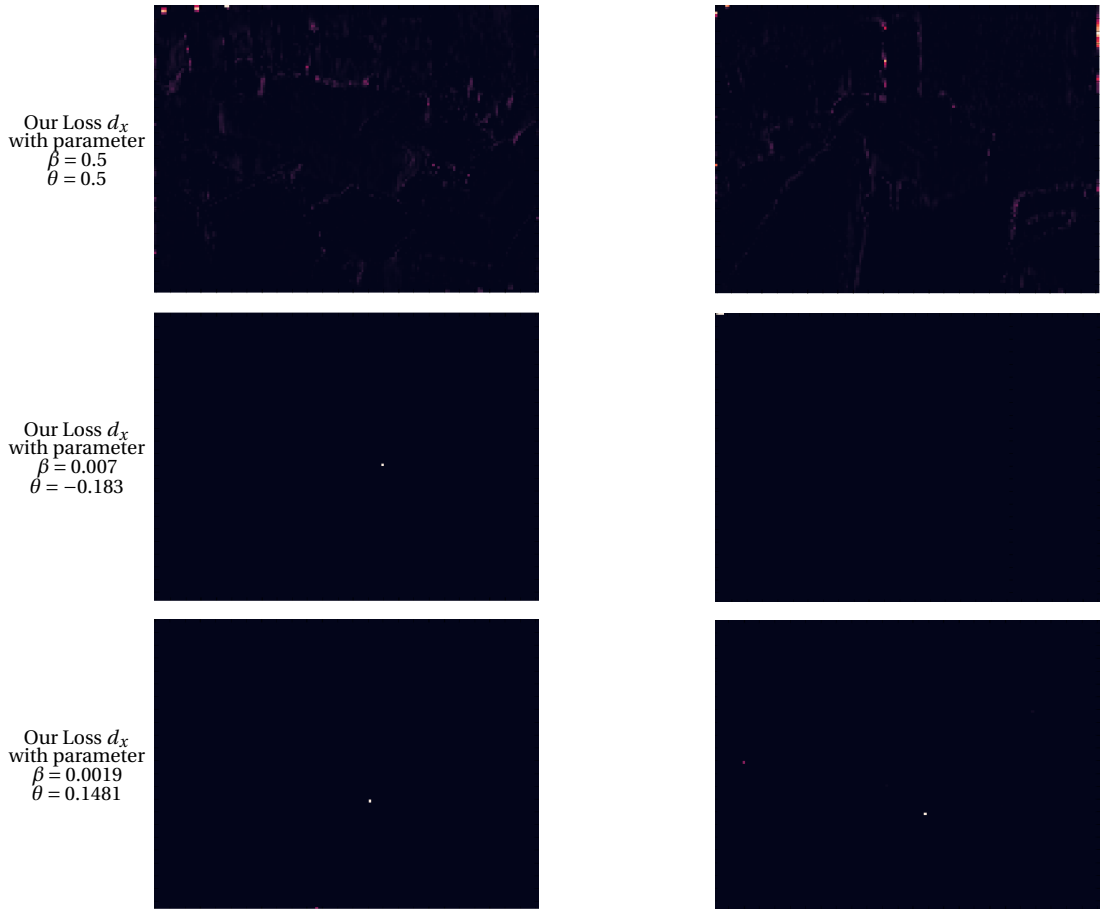


Figure 3.7: Visualization of Our Loss Function Part 2.

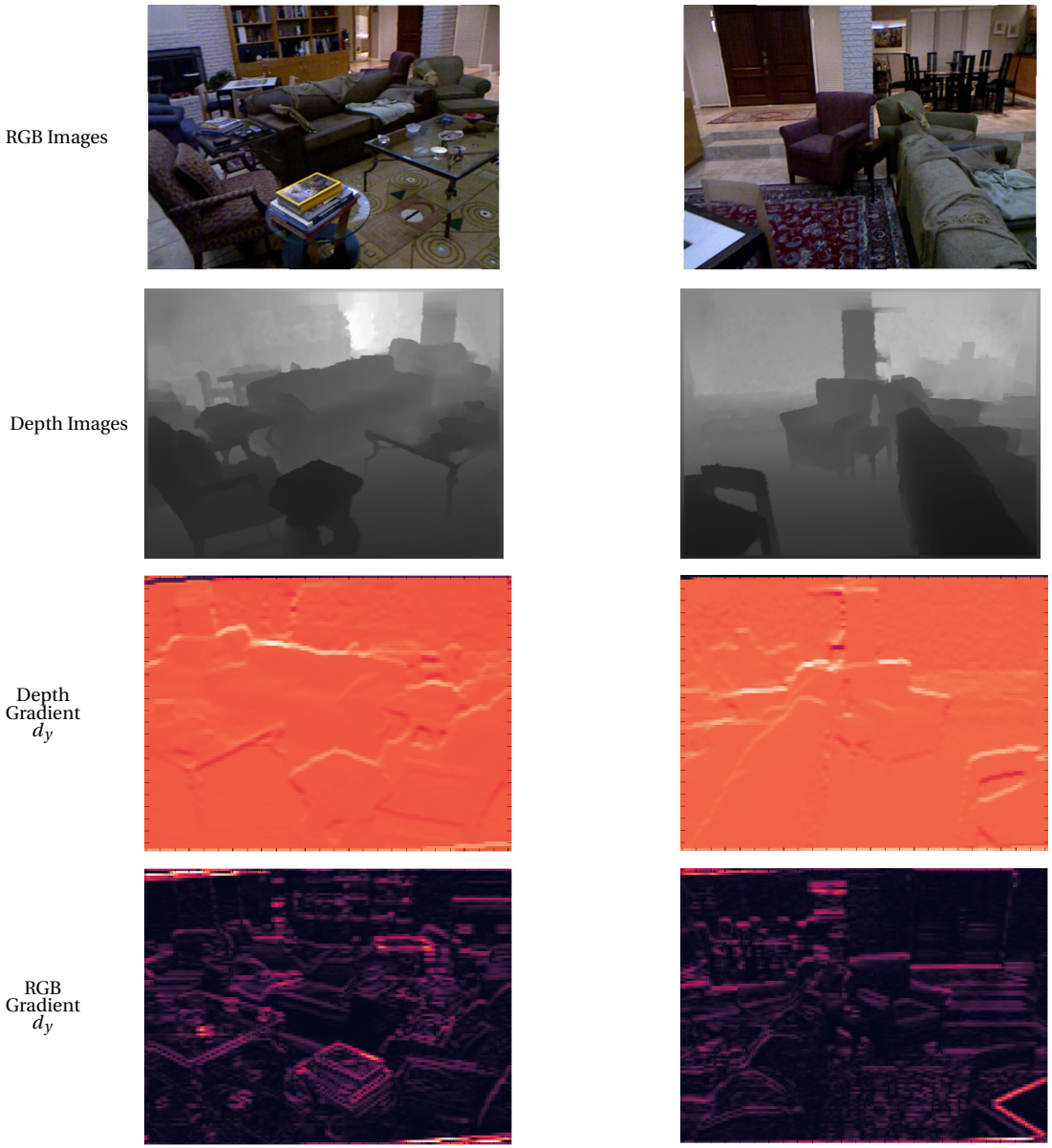


Figure 3.8: Visualization of Our Loss Function Part 1.

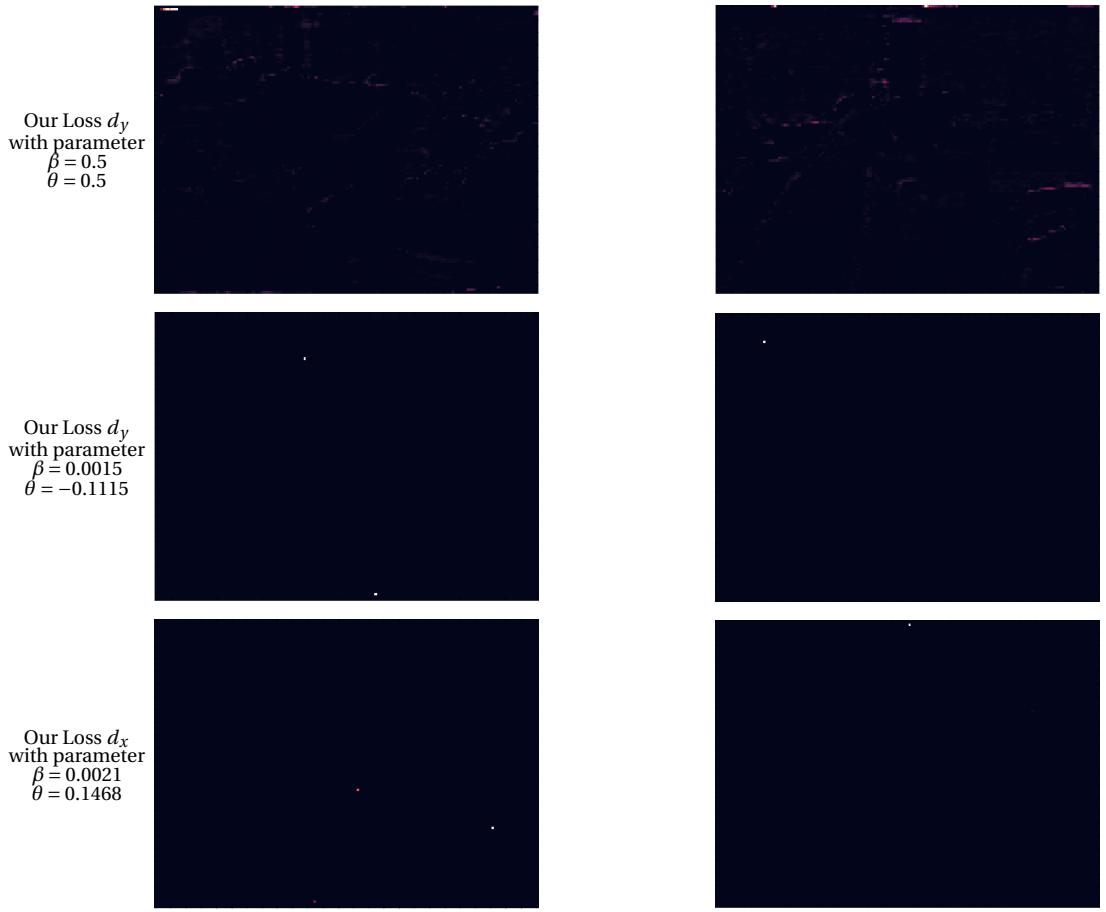


Figure 3.9: Visualization of Our Loss Function Part 2.

Table 3.1: Selected variable pair for β and θ parameters.

$\beta\ d_x$	$\theta\ d_x$	$\beta\ d_y$	$\theta\ d_y$
0.0007	- 0.183	0.0015	- 0.1115
0.0019	0.1481	0.0021	0.1468

Figures 3.6,3.7,3.8 and 3.9 show a visualization of our loss function parts. The first row of part 1 figures shows our loss function without fine-tune β and θ parameters. Without fine-tuning, our loss function tends to penalize some of the edges of the scene. To prevent this, we choose different values for β and θ parameters where loss function mean value is near to zero.

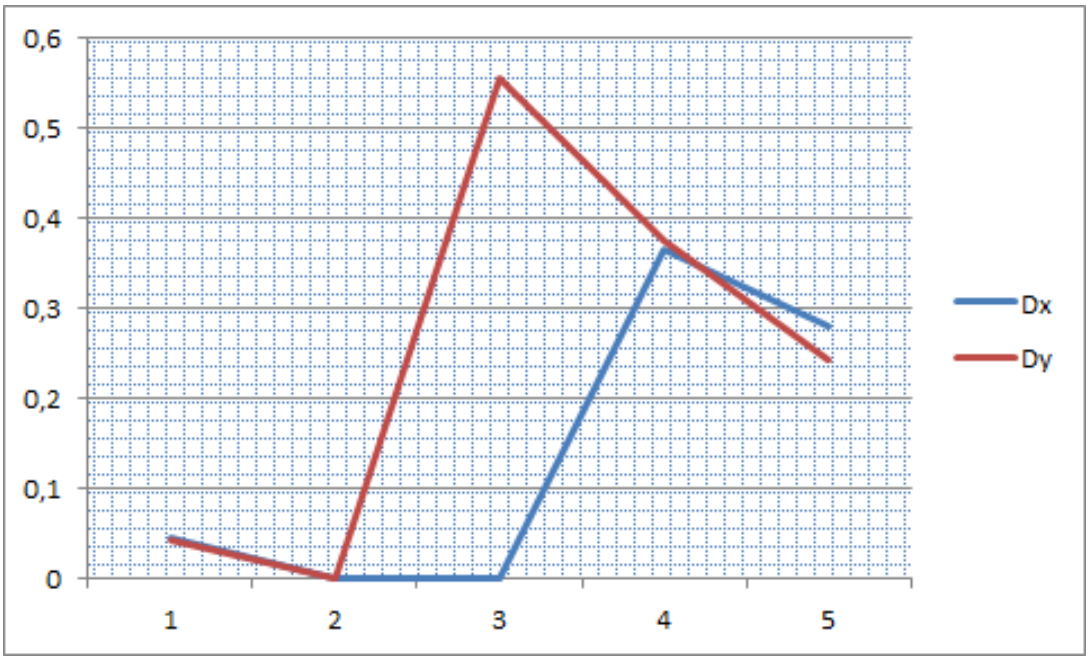


Figure 3.10: Loss variable graph vs epoches

Together with proposed the novel loss function, bound loss, used loss functions are as follows:

$$l_{\text{depth}} = \frac{1}{n}\sum_{i=1}^{n} F(e_i) \tag{36}$$

$$e_i = \left\| d_i - g_i \right\|_1 \tag{37}$$

where d is estimated depth map, g is ground-truth depth map.

$$F(x) = \ln(x + \alpha) \tag{38}$$

$$l_{\text{normal}} = \frac{1}{n} \sum_{i=1}^{n} \left(1 - \frac{\langle n_i^d, n_i^g \rangle}{\sqrt{\langle n_i^d, n_i^d \rangle} \sqrt{\langle n_i^g, n_i^g \rangle}} \right), \tag{39}$$

where h<·,·> denotes the inner product of vectors.

$$l_{\text{grad}} = \frac{1}{n} \sum_{i=1}^{n} \left(F\left(\nabla_x\left(e_i\right)\right) + F\left(\nabla_y\left(e_i\right)\right) \right), \tag{310}$$

where ∇_x is the derivative of e_i computed at the i^{th} pixel.

$$l_{bound_x} = \frac{1}{n} \sum_{i=1}^{n} |\nabla_x(e_i)| \times \sigma(-\frac{\sqrt{\nabla_x(r_i)^2 + \nabla_x(b_i)^2 + \nabla_x(g_i)^2})}{\beta} - \theta) \tag{311}$$

$$l_{bound_y} = \frac{1}{n} \sum_{i=1}^{n} |\nabla_y(e_i)| \times \sigma(-\frac{\sqrt{\nabla_y(r_i)^2 + \nabla_y(b_i)^2 + \nabla_y(g_i)^2})}{\beta} - \theta) \tag{312}$$

$$L = l_{\text{depth}} + \lambda l_{\text{grad}} + \mu l_{\text{normal}} + \alpha(l_{bound_x} + l_{bound_y}) \tag{313}$$

We train DEN with a dataset composed of indoor and outdoor scenes. The NYU-D2 dataset [42] is used as the indoor scenes and Make3D-2 [39] and KITTI datasets as the outdoor scenes.

As the backbone, we use the Sequence and Excitation Network (SeNet-154) [20]. The network is trained using Adam optimizer for 20 epochs with a learning rate of 0.0001.

In addition to this depth estimation network, we also tried the learning-based depth estimation module. The main purpose of this depth module is replacing all auxiliary loss function with a learning-based approach. The module run as follows:

- Divide RGB and depth images into 10×10 patches.
- Calculate the gradient of both RGB and depth images.
- Each gradient is the input of the two fully connected layers contains 120 and 60 neurons respectively, ReLU and Batch Normalization.
- Concatenate output of each stream.
- The final layer is passed through one fully connected layer which has only one neuron for binary classification and the sigmoid function.

For this module, we created the same number of positive and negative examples. Positive examples mean 10×10 matched patches on the depth and RGB images gradient. Negative examples mean 10×10 non-matched patches on the depth and RGB images gradient. For creating negative examples, the 10×10 patch is randomly selected on the RGB image gradient, then any patches that don't have the same coordinates with the patch on RGB images gradient and have pixel-wise difference between the match case patch greater than a chosen threshold can be selected. However, module loss couldn't converge and it didn't learn match and non-matched cases together. Figure 3.11 shows this module's pipeline.

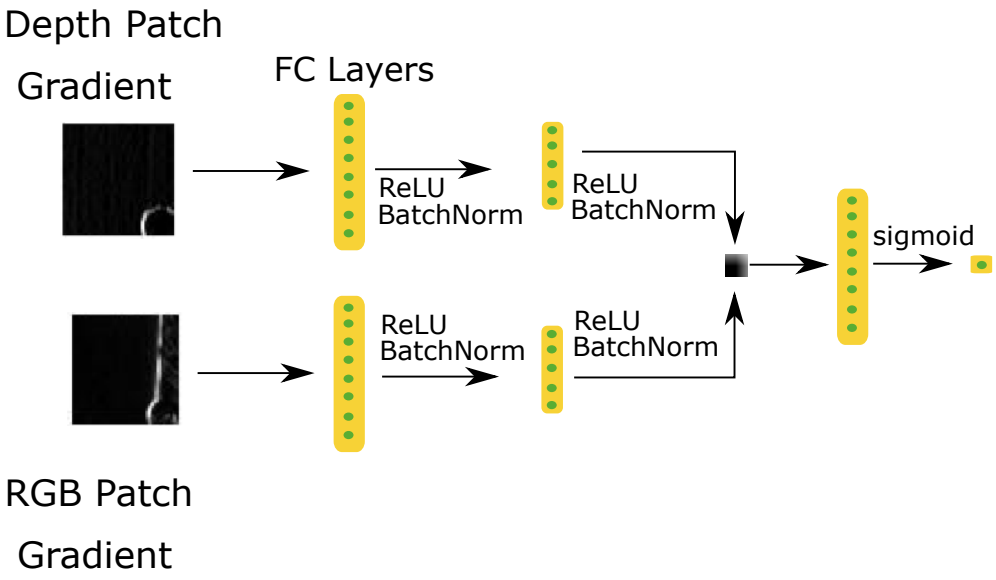


Figure 3.11: Patch Depth module pipeline.

CHAPTER 4

EXPERIMENTS

In this section, we present our experimental results organized in the following five parts:

a-) **Depth Estimation Loss Function Experiments:** We tried different coefficient for combining our loss function with existing loss functions mentioned in 3.3. We proposed result of each tested coefficient with different depth estimation metrics.

b-) **Architecture Experiments:** In this set of experiments, we explored several different ways of integrating the depth map into the object detection pipeline. Specifically, we tried adding the raw depth information in an early or late stage; and processing the depth information before integrating it into the pipeline.

c-) **Depth Estimator Network Experiments:** We tried different state-of-the-art depth estimator models to explore best-estimated depth map and HHA encoding for object detection and observe how these networks would affect the results.

d-) **Depth Estimator Training Set:** We trained chosen depth estimator network with different combinations of different indoor and outdoor depth datasets and observed how training set affects object detection results.

e-) **Datasets Experiments for Object Detection:** Based on the best settings we obtained in the first three parts (a,b and c), we evaluated object detection results for different datasets.

4.1 Depth Estimation Loss Function Experiments

In this section, we tried different values for the α to find its optimal value.

Table 4.1: DEN depth metrics results with different coefficients. Result in the first row is taken from [21]. Our training results of [21] is in the second row. Other rows show results of the network trained with the proposed novel loss function.

Network	δ_1	δ_2	δ_3	ABS_{REL}	MAE	RMSE	MSE	LOG_{10}	P	R	F1
$[21]_1$	**0.866**	**0.975**	0.993	**0.115**	-	0.530	0.281	**0.050**	**0.644**	**0.508**	0.562
$[21]_2$	**0.866**	0.973	0.992	0.116	0.312	0.530	0.281	**0.050**	0.656	0.502	0.562
$\alpha = 20000$	0.371	0.665	0.839	0.377	0.995	142.9	204.4	0.163	1	0.114	0.198
$\alpha = 200$	0.848	0.971	0.992	0.121	0.332	0.559	0.313	0.053	0.661	0.462	0.538
$\alpha = 100$	0.860	0.973	0.992	0.119	0.322	0.545	0.297	0.051	0.660	0.488	0.554
$\alpha = 10$	0.8654	0.973	0.992	**0.115**	0.314	0.538	0.289	**0.050**	0.661	0.492	0.558
$\alpha = 8.5$	**0.866**	0.972	0.991	**0.115**	**0.310**	0.530	0.281	0.05	0.657	0.502	**0.563**
$\alpha = 7.5$	0.865	0.973	**0.993**	0.117	0.312	**0.528**	**0.278**	**0.050**	0.654	0.500	0.562
$\alpha = 6.5$	0.8633	0.972	0.992	0.116	0.315	0.539	0.291	**0.050**	0.659	0.495	0.559
$\alpha = 5$	0.865	0.972	0.991	0.116	0.312	0.538	0.289	**0.050**	0.659	0.499	0.562
$\alpha = 2$	0.864	0.973	0.992	0.117	0.315	0.536	0.287	**0.050**	0.659	0.495	0.560
$\alpha = 1$	0.865	0.972	0.992	**0.115**	**0.310**	0.532	0.283	0.498	0.658	0.500	0.562

α is the proposed loss function coefficient of integration to others loss functions. Therefore larger and smaller values harm the network and prevent learning meaningful depth information. However, when it is chosen proper values such as 8.5 or 7.5, it improves some depth estimation metrics such as absolute relative error, mean average error, root mean square error.

From these experiments, we have chosen alpha as 7.5 and this network was used for next object detection experiments in section 4.5.

4.2 Architecture Experiments

Here we tried to find out the best of the way of using the depth information to improve the end result (i.e. object detection). Also, we wanted to see how depth map with and without extracted features affect the performance of Faster R-CNN. For this purpose, we identified potential concatenation points, shown in Figure 3.4 as red circles. These points are respectively, the output of the backbone network, the output of ROI Align Module and output of backbone network classifier. Then, we tried to combine pooled RGB image feature, size of N x 4096 where N is a number of predicted objects, with ground truth depth maps which are resized to 64 x 64 and flatten to 1 x 4096, without any feature extraction step. Each predicted object shares the same resized and flattened depth vector, so the final feature vector size is N x 8192. This case is shown in figure 3.4 case a. For this case, mAP of Faster R-CNN is equal to 0, shown in the second row of Table 4.2. Therefore, we concluded that concatenation of depth map without feature extraction harms the performance of Faster R-CNN dramatically.
Then, we tried to extract depth maps' features with a small network and these extracted features are concatenated at the same point as in case a. This small network consists of three Conv2D - Relu - Batch Normalization blocks. Conv2D layers' the number of filters is set to 1 for comparing the effect of feature extraction. Like in case a, extracted features are interpolated to size 64 x 64, flattened to 1 x 4096 and concatenated with the pooled feature of RGB images. The final concatenated vector size is N x 8192. This case is shown in figure 3.4 case b. For this case, depth map doesn't improve the performance of Faster R-CNN and Faster R-CNN per-

forms nearly the same as when input is consist of an only RGB image, shown in the third row of Table 4.2

Then, we changed the feature extracted network used in case a and b and concatenation point. We increased the Conv2D layers' number of filters to 128,256 and 512, respectively and adding the max-pooling layer. The output of this network is interpolated to 7 x 7 for matching with the size of pooled RGB image features. Also, we tried to concatenate features at the output of Roi Align Module. Concatenated features have 1024 x 7 x 7 and they reduced to size 512 x 7 x 7 for matching with size of pre-trained backbone network classifier. This case is shown in figure 3.4 case c. In this case, mAP of Faster R-CNN dropped by nearly 9%, shown in the fourth row of Table 4.2

By considering these experiments and previous works [3, 17, 19], we concluded that (i) the depth information should be integrated early into the network, and (ii) it should be processed like RGB images. Therefore, we tried to extract depth features using the VGG-16 network. We have two separate VGG-16 networks as a backbone and features obtained from these two networks are concatenated. We chose early concatenation because we wanted to minimize the number of parameters. Then, we add the Conv2D layer for matching the pre-trained backbone network classifier size. This case is shown in figure 3.4 case d. In this case, the depth map improves Faster R-CNN performance, shown in the last row of Table 4.2.

4.3 Depth Estimator Network Experiments

Widely used Pascal VOC and MS-COCO provided do not provide ground-truth depth. Therefore, if the depth is to be used, it should be estimated from single RGB images. For this purpose, we tested the performance of different state-of-the-art depth estimation networks for depth map estimation and creating HHA encoding from the estimated depth map. Tested methods are respectively [30], [15], [21], [51]. Figure 4 shows the input image, ground truth depth map and estimated depth map of each network. For all networks, official pre-trained models are used. For [15], case d, we tested indoor the pre-trained model.

Figure 4.1: Depth estimation network outputs for images example from Pascal VOC 2007.

Also, we tried to generate HHA images from the gray-scale depth map by training "Image-to-Image Translation with Conditional Adversarial Networks", [24]. We used NYUD2 RGB-D dataset as the training set and tested results with Pascal Voc 2007 images. We observed that generated HHA images have great noises and artifacts. Although we enlarged training sets, these noises and artifacts did not disappear.

Reference [21] produces the best results as seen in Figure 4, case e. It has loss function related to object boundaries and this is crucial for generating HHA encoding. When generating HHA encoding from all other networks, outputs are visually senseless.

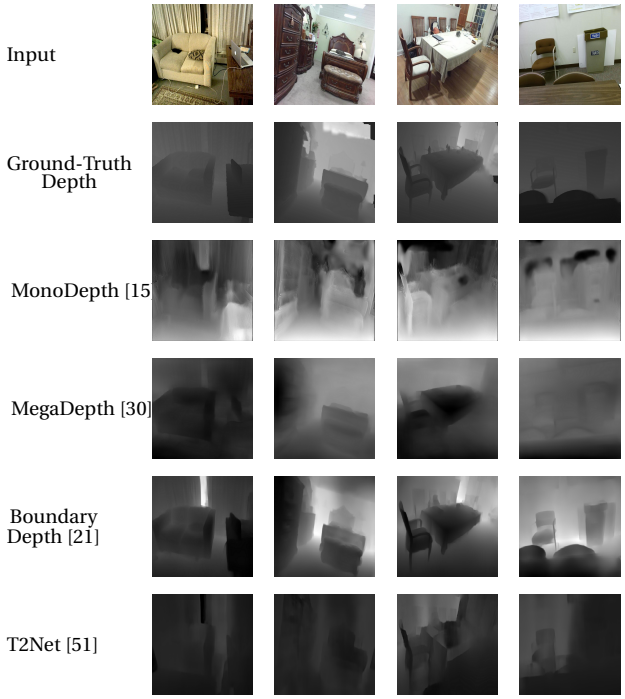


Figure 4.2: Depth estimation network outputs for images example from SUN RGB-D test set.

4.4 Depth Estimator Training Set Experiments

After choosing the best depth estimator network, we observed the estimated depth map effect on Faster R-CNN's performance. Even if they contribute Faster R-CNN's mAP on SUN RGB-D, NYU-D2 and Pascal VOC 2007 Indoor Categories, shown in table 4.3, 4.4, 4.5, they drop mAP of Faster R-CNN on whole Pascal VOC 2007 dataset, shown in table 4.6. this could be related to the training set of chosen depth estimator network, because of the official pre-trained model trained with Nyu-d2 datasets which consist of an only indoor scene. Therefore, we trained it with different indoor and outdoor datasets to see how training datasets affect Pascal VOC 2007 object detection result. Used datasets are respectively, NYUD2 for indoor scenes, Make3D [39],[40] and KITTI for outdoor scenes. By using these datasets, we trained 4 different combinations and they are as follows:

1. NYUD2 official training split:
    - It includes 249 different scenes and 45k RGB-D indoor images from

NYUD2 dataset

2. NYUD2 official training split + Make3D-2 training and test split

    - It includes 249 different scenes and 498 RGB-D indoor images from NYUD2 dataset and 374 different scenes and outdoor images from Make3D-2 datasets. To maintain a balance of indoor-outdoor data distribution, we chose two random images for each NYUD2 scene.

3. KITTI raw dataset

    - It includes 24k RGB-D outdoor images from KITTI raw dataset.

4. KITTI raw dataset + NYUD2 official training split

    - It includes 24k RGB-D outdoor images from KITTI raw dataset and 45k indoor images from NYUD2 datasets.

Visual results are shown in Figure 4.3 and object detection results are shown in Table 4.7. Even if training set of depth estimator network enlarged with different outdoor datasets, estimated depth maps couldn't improve Pascal VOC 2007 object detection result.

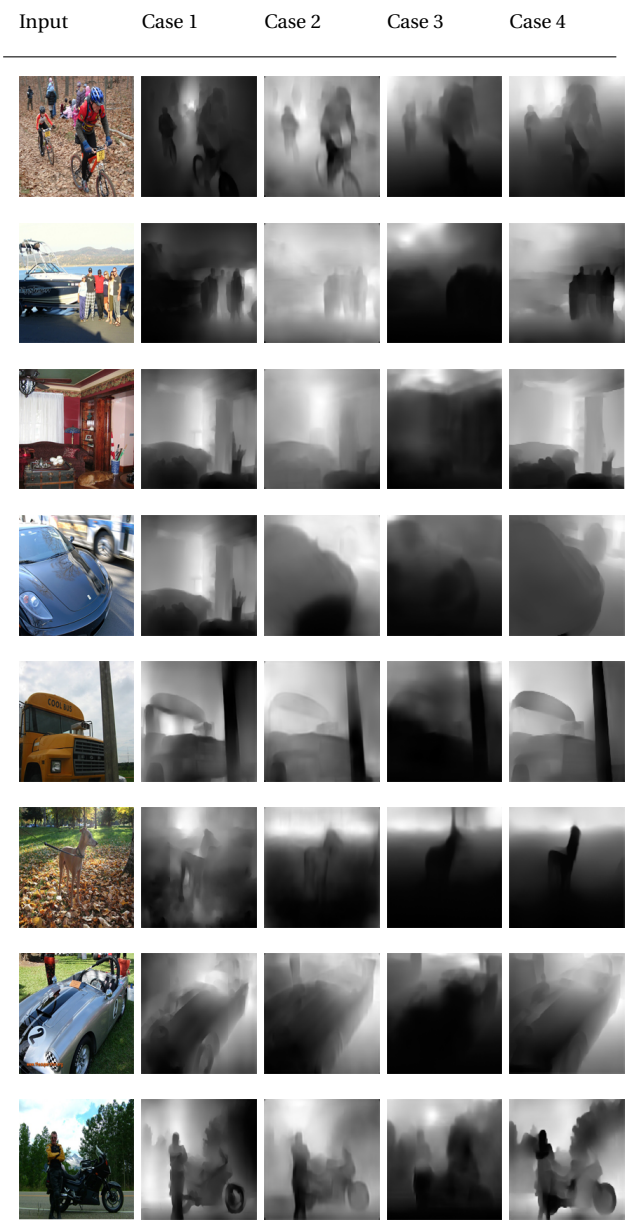


Figure 4.3: Depth estimation network trained with different datasets output results. Images are taken from Pascal VOC 2007.

4.5 Datasets Experiments for Object Detection

After choosing the best architecture and depth estimator network, different datasets with different ground-truth and estimated input types, gray-scale depth map, depth map in the jet color space, HHA encoding, are used for observing the performance of RGB-D object detection. Figure 3.3 shows these input types. We choose VGG-16 [43] and ResNet-101 [[18] as the backbone, and faster-R CNN as the base object detector. NYUD2, SUN RGB-D, and PASCAL VOC 2007 datasets are used.

Table 4.2: Object detection results in NYUD2 Dataset for Architecture Experiments. The first coloumn shows only RGB input results of Faster R-CNN. The next four coloumn show ground truth depth map as additional input. NYUD2 datasets are taken from [17]. Training and test set are same with official splits.

	Base Network	Case a	Case b	Case c	Case d
bathtub	**30.0**	0.0	27.2	18.5	28.8
bed	60.0	0.0	58.9	56.1	**65.6**
b.shelf	**41.8**	0.0	41.0	37.1	37.1
box	2.7	0.0	2.4	1.8	**3.1**
chair	41.1	0.0	43.9	38.3	**44.6**
counter	40.0	0.0	37.9	30.9	**42.3**
desk	11.6	0.0	13.0	11.6	**15.7**
door	**23.3**	0.0	22.3	20.4	22.4
dresser	36.1	0.0	38.4	36.2	**41.8**
g.bin	**27.8**	0.0	23.7	23.5	24.9
lamp	29.7	0.0	31.3	28.2	**33.9**
monitor	**53.6**	0.0	50.6	51.7	49.8
n.stand	27.0	0.0	32.4	30.8	**35.9**
pillow	28.8	0.0	**30.0**	28.1	29.2
sink	30.3	0.0	28.3	28.1	**33.7**
sofa	48.0	0.0	49.6	41.0	**51.3**
table	25.1	0.0	27.8	22.0	**31.5**
t.vision	**42.9**	0.0	39.5	33.5	39.2
toilet	56.0	0.0	**56.4**	55.3	55.5
mean	34.5	0.0	34.4	31.2	**36.1**

Table 4.3: Object detection results in NYUD2 Dataset. (*) means ground-truth inputs. The First row shows only RGB input results of Faster R-CNN. The next three rows show additional ground truth input types result. The next six rows show additional estimated input types result. (‘) means that proposed bound loss is used for generating estimated input types. GDep means the depth map in grayscale and JDep means the depth map in the jet color map. Training and test set are same with the official split.

	b.bone	bathtub	bed	b.shelf	box	chair	counter	desk	door	dresser	g.bin	lamp	monitor	n.stand	pillow	sink	sofa	table	t.vision	toilet	mean
RGB	VGG-16	30.0	60.0	41.8	2.7	41.1	40.0	11.6	23.3	36.1	27.8	29.7	53.6	27.0	28.8	30.3	48.0	25.1	**42.9**	56.0	34.5
RGB+HHA*	VGG-16	32.5	71.7	41.4	2.7	47.2	43.9	15.8	23.7	40.1	37.0	35.7	48.6	38.5	30.0	41.1	51.0	27.5	37.5	60.6	38.2
RGB+GDep*	VGG-16	28.8	65.6	37.1	3.1	44.6	42.3	15.7	22.4	41.8	24.9	33.9	49.8	35.9	29.2	33.7	51.3	**31.5**	39.2	55.5	36.1
RGB+JDep*	VGG-16	36.3	71.3	45.7	4.4	46.0	38.3	10.6	23.9	43.1	28.9	31.7	52.5	38.7	27.9	33.2	49.4	28.9	28.3	54.0	36.5
RGB+HHA	VGG-16	37.5	65.7	42.3	3.2	45.6	42.9	11.3	26.5	49.5	31.4	33.7	47.6	41.2	29.7	35.0	51.3	27.3	33.4	63.8	37.8
RGB+HHA ‘	VGG-16	37.3	65.4	42.7	2.7	45.3	42.8	11.0	26.2	49.3	31.1	33.3	47.8	40.9	30.0	34.5	50.9	26.8	33.1	63.3	37.6
RGB+GDep	VGG-16	29.6	71.8	44.2	3.2	46.5	43.8	19.7	24.8	49.7	30.9	31.8	48.8	32.1	29.9	33.0	45.6	26.9	36.4	62.0	37.4
RGB+GDep ‘	VGG-16	28.3	70.2	46.1	2.4	39.7	41.1	18.3	25.3	47.6	28.7	32.3	51.2	29.0	27.1	30.3	42.5	22.4	32.9	58.3	35.5
RGB+JDep	VGG-16	35.4	71.2	35.7	3.0	44.3	43.7	12.6	23.9	45.4	28.5	33.4	50.0	37.0	27.3	31.4	49.0	29.7	31.2	53.0	36.1
RGB+JDep ‘	VGG-16	24.4	63.5	40.6	2.8	40.0	44.5	13.0	23.7	44.9	31.8	32.3	53.9	39.0	28.6	35.3	46.4	27.1	32.1	58.7	35.9
RGB	Res-101	30.2	68.2	44.0	4.4	48.2	44.7	16.0	26.5	46.2	38.8	38.4	**60.9**	**53.0**	29.4	41.1	53.9	28.4	40.3	58.1	40.6
RGB+HHA*	Res-101	**43.6**	75.6	45.4	5.4	**54.1**	50.4	18.2	29.9	51.2	39.0	39.9	48.0	49.1	**41.1**	**48.**	62.1	29.9	37.4	**63.9**	**43.8**
RGB+GDep*	Res-101	26.4	72.6	45.0	4.8	49.4	50.7	16.6	31.2	**53.8**	41.0	35.9	52.1	49.0	34.3	46.3	60.0	27.1	28.3	60.5	41.3
RGB +JDep*	Res-101	28.0	**76.3**	46.7	**6.0**	52.8	50.3	**22.5**	**32.0**	50.6	40.7	**41.6**	52.1	51.1	35.2	45.7	61.0	30.6	40.9	61.3	43.4
RGB + HHA	Res-101	26.8	68.8	47.5	5.0	50.6	**53.7**	18.3	28.9	46.2	41.4	38.2	38.7	48.0	33.8	38.0	60.0	30.3	31.8	58.8	40.2
RGB + HHA ‘	Res-101	26.1	67.7	**48.8**	4.9	47.4	49.8	14.0	28.2	44	29.9	38.5	52.4	48.3	34.5	43.0	58.0	27.7	40.5	57.2	40.1
RGB + GDep	Res-101	24.4	73.7	44.08	4.1	49.1	47.6	19.7	30.6	52.5	**41.9**	37.3	54.4	51.5	34.8	40.6	**62.8**	27.9	31.2	58.6	41.4
RGB + GDep ‘	Res-101	25.0	70.2	48.0	4.8	46.4	49.2	20.6	27.5	47.0	38.5	35.7	53.4	42.5	34.8	42.2	62.7	23.0	34.7	57.0	40.2
RGB + JDep	Res-101	24.5	74.2	41.8	5.2	51.4	46.4	17.4	27.7	50.2	33.1	36.5	49.4	47.1	34.8	42.3	60.9	27.2	32.8	61.5	40.2
RGB + JDep ‘	Res-101	36.2	73.0	45.4	4.9	47.1	49.8	14.3	28.7	44.6	37.4	38.2	47.3	48.5	33.4	37.8	56.2	24.2	29.7	67.2	40.2

Table 4.4: Object detection results in SUN RGB-D Dataset. (*) means ground-truth inputs. The First row shows only RGB input results of Faster R-CNN. The next three rows show additional ground truth input types result. The next six rows show additional estimated input types result. (‘) means that proposed bound loss is used for generating estimated input types. GDep means the depth map in grayscale and JDep means the depth map in the jet color map.

	B.bone	bathtub	bed	b.shelf	box	chair	counter	desk	door	dresser	g.bin	lamp	monitor	n.stand	pillow	sink	sofa	table	t.vision	toilet	mean
RGB	VGG-16	55.2	72.6	45.6	14.3	59.8	48.3	26.7	52.8	40.9	56.4	58.6	47.4	56.6	48.8	62.0	52.9	42.9	39.0	84.8	50.8
RGB+HHA*	VGG-16	64.6	82.0	46.7	16.3	63.3	50.7	30.0	53.5	43.0	60.3	59.1	49.8	60.3	56.7	65.2	53.1	47.1	42.4	86.9	54.3
RGB+GDep*	VGG-16	64.3	80.3	49.8	17.1	62.9	49.5	29.7	53.7	43.6	59.9	61.1	45.8	59.5	55.5	65.4	57.6	47.6	45.0	88.8	54.5
RGB +JDep*	VGG-16	69.8	81.8	45.5	15.1	63.4	50.7	29.0	50.0	42.9	60.1	61.4	48.9	**61.5**	54.6	63.9	55.1	46.8	40.8	87.5	54.2
RGB + HHA	VGG-16	55.9	78.4	45.9	14.6	60.5	50.4	28.4	53.4	42.2	57.1	59.3	48.6	59.7	48.9	62.2	55.8	43.5	39.7	**89.6**	52.3
RGB + HHA ‘	VGG-16	55.0	75.1	47.6	14.0	59.7	48.5	27.1	50.2	39.5	56.8	58.9	44.2	56.4	49.3	61.5	51.8	45.6	34.5	84.0	50.5
RGB + GDep	VGG-16	56.0	78.9	47.4	14.0	61.2	48.3	26.6	52.8	42.8	55.7	59.0	47.0	57.6	48.6	61.4	53.0	45.4	37.5	86.5	51.6
RGB + GDep ‘	VGG-16	53.1	72.5	46.6	13.1	59.1	45.6	25.4	50.7	37.8	54.2	56.3	45.5	56.5	48.4	60.0	50.1	42.6	36.2	86.6	49.5
RGB + JDep	VGG-16	59.25	78.7	49.5	15.1	61.0	49.5	26.6	52.8	42.2	57.6	59.2	45.0	60.1	49.7	63.6	55.6	43.9	39.7	87.4	52.4
RGB + JDep ‘	VGG-16	53.8	75.5	44.0	14.6	59.1	47.0	25.4	52.0	38.6	56.2	57.2	44.7	57.6	47.9	60.7	52.6	43.7	33.3	86.0	50.0
RGB	Res-101	65.5	80.0	50.8	16.4	63.5	52.4	30.4	56.7	43.7	60.3	61.1	50.2	57.0	54.3	72.9	58.8	47.0	45.7	88.9	55.5
RGB+HHA*	Res-101	**72.7**	**84.2**	51.4	17.8	66.3	51.0	32.3	56.3	43.7	60.1	**64.3**	49.2	60.9	60.1	**75.1**	60.3	**47.8**	45.1	88.6	**57.2**
RGB+GDep*	Res-101	65.4	84.1	49.9	**19.7**	**66.8**	**54.7**	**32.6**	57.7	**46.3**	**61.4**	63.8	50.1	60.0	**60.7**	74.2	**61.7**	47.0	**49.9**	59.2	57.6
RGB +JDep*	Res-101	71.1	**84.2**	**52.8**	18.0	65.8	51.0	32.2	57.2	44.3	61.2	62.7	46.9	58.5	56.9	73.4	60.6	47.3	42.1	88.5	56.7
RGB + HHA	Res-101	62.2	80.8	54.6	15.7	63.7	52.0	29.8	56.5	44.7	57.3	62.6	47.7	59.0	54.7	70.4	57.9	45.2	40.5	88.9	55.0
RGB + HHA ‘	Res-101	63.9	81.0	49.3	16.9	63.7	52.9	30.1	55.2	42.2	57.2	61.1	45.5	55.1	53.7	73.0	59.2	47.4	40.6	90.5	54.6
RGB + GDep	Res-101	55.8	80.5	51.8	16.3	64.1	51.6	29.6	57.5	45.3	59.7	61.2	**52.4**	60.6	56.1	70.5	60.0	43.5	43.1	88.9	55.2
RGB + GDep ‘	Res-101	58.7	80.8	51.0	14.5	64.6	51.3	31.8	56.6	43.2	59.3	62.5	47.7	59.4	55.1	70.1	59.2	47.1	40.3	88.8	53.8
RGB + JDep	Res-101	55.8	81.7	51.6	15.8	64.7	50.5	30.9	**57.7**	43.5	58.9	60.0	47.0	56.1	55.1	69.3	57.2	46.0	41.5	89.9	54.4
RGB + JDep ‘	Res-101	53.8	79.7	50.8	14.5	64.2	51.1	29.2	55.1	42.8	59.1	58.3	46.7	58.2	55.8	68.9	58.8	46.1	34.7	88.3	53.4

Table 4.5: Pascal VOC 2007 Indoor Categories Results. The training set is VOC 2007 official trainval and test set is VOC 2007 official test set. The first row shows RGB results for Faster R-CNN. Other rows show additional estimated input types result. JDep means the depth image in the jet color space. GDep means the gray-scale depth map.

	B.Bone	Bottle	Chair	D.Table	P.Plant	Sofa	Tv/Mon.	Mean
RGB	VGG-16	39.05	54.96	54.26	42.12	74.03	75.09	56.58
RGB+HHA	VGG-16	41.41	51.76	61.42	40.07	76.91	76.49	58.01
RGB+JDep	VGG-16	44.03	52.92	64.38	39.84	73.08	74.53	58.13
RGB+GDep	VGG-16	46.17	51.38	56.44	41.88	77.26	75.19	58.05
RGB	Res-101	**49.3**	57.3	63.4	**46.4**	77.5	78.6	62.1
RGB+HHA	Res-101	47.8	58.1	**68.5**	45.8	**80.0**	78.9	**63.1**
RGB+JDep	Res-101	47.8	**58.3**	62.1	45.8	77.9	**79.1**	61.8
RGB+GDep	Res-101	46.4	57.1	60.0	44.0	77.6	78.1	60.5

Table 4.6: Pascal VOC 2007 Results. The training set is VOC 2007 official trainval and test set is VOC 2007 official test set. The first row shows RGB results for Faster R-CNN. Other rows show additional estimated input types result. JDep means the depth image in the jet color space. GDep means the gray-scale depth map. (‘) means that proposed bound loss is used for generating estimated input types

	B.Bone	A.plane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	D.table	Dog	Horse	M.bike	Person	P.plant	Sheep	Sofa	Train	Tv/mon.	Mean
RGB	VGG-16	70.5	77.6	66.3	54.5	52.2	80.8	84.5	84.1	50.2	78.2	65.3	77.5	82.9	72.6	77.2	42.2	71.8	65.5	75.2	72.0	70.1
RGB-HHA	VGG-16	73.7	76.4	67.1	52.7	51.3	78.1	82.8	79.5	50.5	78.5	62.9	78.5	81.9	75.6	77.4	42.7	70.0	66.4	75.6	70.5	69.6
RGB-HHA ‘	VGG-16	72.2	76.2	67.9	55.7	51.3	78.1	85.5	81.6	49.3	74.7	63.0	78.7	82.1	73.2	77.1	44.8	67.6	64.8	73.0	71.2	69.4
RGB-JDep	VGG-16	72.7	78.0	67.7	53.1	51.1	79.1	84.2	81.2	**58.4**	77.8	64.1	79.2	80.3	75.8	76.8	42.9	72.4	66.3	73.4	71.4	69.8
RGB-JDep ‘	VGG-16	70.8	78.8	67.0	54.8	50.7	78.7	83.3	80.5	47.7	77.1	61.5	78.7	84.5	75.7	77.2	42.9	69.8	65.4	72.7	70.3	69.4
RGB-GDep	VGG-16	69.8	79.3	66.1	54.24	51.8	74.2	83.2	82.1	46.9	76.9	56.9	77.8	82.5	74.1	76.6	42.5	67.3	66.8	73.3	71.2	68.7
RGB-GDep ‘	VGG-16	68.3	77.4	64.0	55.8	53.1	74.6	83.0	80.5	48.1	77.4	58.2	76.6	81.7	74.4	77.1	41.2	68.2	61.9	72.2	70.1	68.2
RGB	Res-101	**79.1**	**81.5**	76.7	**67.6**	59.9	81.5	86.3	86.6	55.4	**83.6**	66.2	**86.7**	**85.5**	78.3	79.6	**50.0**	**75.6**	**75.3**	79.0	73.4	**75.4**
RGB+HHA	Res-101	77.7	80.0	**78.6**	66.3	62.5	**83.**	86.1	86.8	56.7	81.4	67.8	85.7	83.1	**79.7**	78.6	48.9	74.2	74.5	**79.1**	74.1	75.3
RGB+HHA ‘	Res-101	77.6	79.5	76.6	64.7	59.3	**83.**	86.5	86.2	54.0	80.2	**68.0**	85.0	82.3	76.1	78.7	43.9	74.6	75.3	80.6	72.1	74.2
RGB+JDep	Res-101	79.0	80.8	76.5	60.9	59.8	81.2	**86.8**	**87.7**	53.6	81.1	65.8	84.9	84.3	78.6	79.5	47.0	73.4	74.7	76.5	75.4	74.4
RGB+JDep ‘	Res-101	77.3	80.9	76.2	64.1	60.4	83.1	86.1	86.0	54.0	77.7	68.4	85.0	83.9	76.3	78.7	46.0	75.0	74.7	78.1	74.9	74.3
RGB+GDep	Res-101	78.3	80.2	75.9	62.3	60.1	80.5	87.1	87.4	52.7	80.8	66.0	83.9	84.8	77.7	**79.8**	47.6	72.6	73.9	77.3	**76.2**	74.3
RGB+GDep ‘	Res-101	76.7	78.6	77.6	65.0	**61.2**	83.0	85.3	85.7	55.8	80.5	66.2	84.9	85.5	77.8	78.7	45.5	74.8	75.9	76.0	73.5	74.4

Table 4.7: Pascal VOC 2007 Results for depth map output of Depth estimation network training with different datasets. The backbone is VGG-16 and the additional input type is the gray-scale depth map. The training set is VOC 2007 the official trainval and test set is VOC 2007 official test set.

Training Dataset Number of Training Data	A.plane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	D.table	Dog	Horse	M.bike	Person	P.plant	Sheep	Sofa	Train	Tv/mon.	Mean
NYUD2 45k	**72.7**	78.0	**67.7**	53.1	51.1	79.1	84.2	81.2	**58.4**	77.8	64.1	79.2	80.3	75.8	76.8	**42.9**	**72.4**	66.3	73.4	71.4	69.8
NYUD2 + Make3D 1k	71.9	**78.2**	66.6	55.7	49.9	80.2	**84.9**	80.8	46.8	76.8	**65.4**	**81.1**	80.4	**76.4**	**77.2**	42.5	70.1	**67.1**	**73.8**	**72.2**	**69.9**
KITTI 24k	69.3	78.0	65.7	55.6	50.5	**80.4**	84.4	**82.2**	47.1	76.2	63.0	78.8	81.5	74.1	77.0	42.5	69.6	66.4	69.5	70.7	69.1
KITTI+ NYUD2 70K	69.1	77.6	63.9	**56.4**	**51.8**	78.5	84.7	81.8	48.8	**78.7**	63.3	80.3	**82.5**	75.1	76.7	41.8	69.9	65.2	72.6	71.5	69.5

In previous works [3, 17], used base networks are respectively, R-CNN [14] and Fast R-CNN [13], but the idea behind using additional input is the same with each other. Additional input and RGB images pass through separate networks until reaching the bounding box regression layer. Before passing through this layer, features obtained from these two separate networks are concatenated. Bounding box regression and classification tasks are performed on these fused features. We implemented this idea on Faster R-CNN. Figure 4.4, case a shows this architecture in detail. Tables 4.8, 4.9 and 4.10 show comparisons in terms of mAP and number of parameters. Our method outperforms the idea of late concatenation used in [3, 17] for all test cases that use HHA encoding as additional input, except the only case of ground-truth HHA, NYU-D2 dataset for VGG16 backbone, as shown in table 4.9. Also, it again outperforms them for all test cases that use depth as additional input, as shown in table 4.10.

Also, we tried to combine our architecture with an idea that separately processes each channel of HHA encoding, proposed in [19]. Figure 4.4, case b shows this combination pipelines and tables 4.8, 4.9 show comparisons in terms of mAP and number of parameters. Although there is a huge difference between our method and [19] in terms of the number of parameters, our method performs better than [19] for NYU-D2 datasets. However, it outperforms our work for SUN RGB-D datasets.

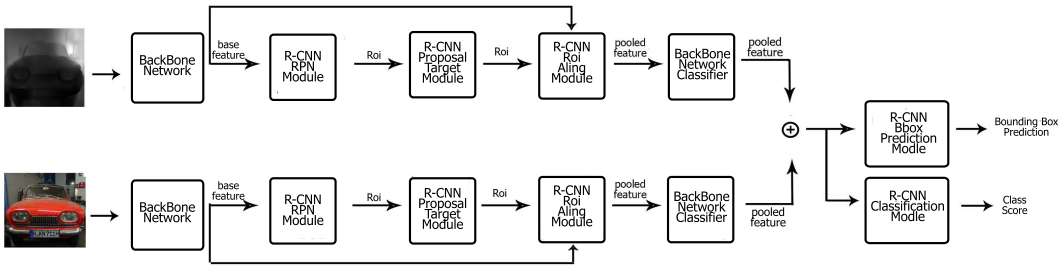


Figure 4.4: Previous works' model architecture on Faster R-CNN. Case a represents, late on concatenation on Faster R-CNN. Case b represents a combination of our architecture and proposed HHA encoding processing method in [19] on Faster R-CNN.

Table 4.8: Number of parameter comparison between Faster R-CNN, Proposed Architecture, and Previous Works' Architecture.

	VGG-16				Res-101			
Models	Faster R-CNN	Ours	[3, 17]	[19]	Faster R-CNN	Ours	[3, 17]	[19]
Trainable Parameters	137M	152M	273M	181M	47M	76M	94M	133M
Fixed Parameters	260K	520K	520K	1M	328K	632K	655K	1M
Total Parameters	137M	152M	274M	182M	47M	77M	95M	135M

Table 4.9: Mean Average Precision Comparison between Faster R-CNN, Proposed Architecture and Previous Works' architecture for the case of using HHA representation as additional input. * means ground-truth input type.

	NYUD2		SUN RGB-D		Pascal VOC 2007	
	VGG-16	Res-101	VGG-16	Res-101	VGG-16	Res-101
Faster R-CNN	34,5	40.6	50.8	55.5	**70.1**	**75.4**
*Ours	38.2	**43.8**	54.3	57.2	-	-
*Previous' [3, 17]	**39.3**	42.8	51.0	54.7	-	-
*Previous' [19]	38.2	43.5	**55.0**	**57.9**	-	-
Ours	37.8	40.2	52.3	55.0	69.6	75.3
Previous' [3, 17]	35.2	38.2	49.0	52.0	66.6	73.0
Previous' [19]	35.9	40.5	51.6	54.9	69.8	73.8

Table 4.10: Mean Average Precision Comparison between Faster R-CNN. Proposed Architecture and Previous Works' architecture for the case of using the gray-scale depth map as additional input. * means ground-truth input type.

	NYUD2		SUN RGB-D		Pascal VOC 2007	
	VGG-16	Res-101	VGG-16	Res-101	VGG-16	Res-101
Faster R-CNN	34.5	40.6	50.8	55.5	**70.1**	**75.4**
*Ours	36.1	41.3	**54.5**	**57.6**	-	-
*[3, 17]	35.9	40.9	51.4	55.1	-	-
Ours	**37.4**	**41.4**	51.6	55.2	69.8	74.4
[3, 17]	37.0	40.0	49.5	52.5	66.6	73.0

Figure 4.5: Results visualization examples for improved cases. Images are taken from SUN RGB-D test split.

Figure 4.6: Results visualization examples for failure cases. Images are taken from SUN-RGBD test split.

Figures 4.5 and 4.6 show examples of the visual results for improved and failure cases.

4.6 Implementation Details

During experiments, faster R-CNN in [48] was used. For backbone VGG-16 and Resnet-101, pre-trained models are used. For VGG-16, before the conv3 layers are fixed. For Resnet-101, the first block and batch normalization layers are fixed. For RGB and other selected input type, there are two separate backbone networks. All input images' shortest side is scaled 600 in the training and the testing stages. During training, the flip operation is used as data augmentation. After applying non-maximum suppression (NMS), top 2000 scoring boxes are selected in training and top 300 scoring boxes are selected in the testing stage. For all experiment batch size equals to 1. Stochastic gradient descent with momentum is applied. The initial learning rate is 0.001 and momentum is 0.9. For every 5 epochs, the learning rate is decreased by factor 0.1. For Pascal Voc 2007 and SUN-RGB datasets, the RGB model is trained with 6 epochs, other models are trained with 7 epochs. For

NYUD2 dataset, all models are trained with 15 epoch. Tesla v100 is used in experiments.

CHAPTER 5

SUMMARY & DISCUSSION

In this thesis, we investigate depth maps effects on object detection problem from different angles. Also, we propose a new model that combines RGB images and depth maps features at the early layer and new loss functions that try to handle consistency between RGB images and the depth map discontinuities.

Compared to existing RGB-D object detection works, our model has an advantage of the fewer number of parameters, see table 4.8. Another advantage is that our method outperforms previous works for most cases, see tables 4.9 and 4.10. Besides our model, we explore new loss functions which are hand-crafted and learning-based. Although our hand-crafted loss function improves mean average and absolute errors for depth estimation, alignment problems between ground-truth depth maps and RGB images decrease the effectiveness of it. The learning-based loss function which is proposed to solve these alignment problems also fails because the neural network architecture that we used was not as successful as we expected in discriminating matched and unmatched patches.

In addition to the proposed RGB-D object detection model and loss functions for depth estimation, we clarify the following issues after thorough experimentation:

- Effect of the depth map feature extraction on Faster R-CNN performance.
- Effect of the depth map/HHA encoding types which are ground-truth and estimated on Faster R-CNN performance.
- Effect of different types of encoding such as HHA, depth map in grayscale and jet color space on Faster R-CNN performance.

- Effect of depth/HHA encoding features integration point to RGB features on Faster R-CNN performance.

For the first issue, we can say that the depth map should be processed like an RGB image. The depth map features should be extracted by the backbone network and these features should be used in the Faster R-CNN pipeline. Using the depth map in the raw form drops the performance of Faster R-CNN dramatically.

For explaining the second and the third issue, we use different depth/HHA encodings as an additional input to an RGB image. This additional input has two main types. The first one is ground-truth depth images obtained from depth sensors and HHA encoding obtained from these depth images. In this case, our method always improves Faster R-CNN detection results without being affected by the backbone network choice and datasets. The second one is using estimated depth images obtained from the depth estimator network and HHA encoding obtained from these depth images. In this case, the backbone network and used datasets affect proposed method detection performance. While depth/HHA features improve object detection results for the indoor scenes, our RGB-D object detection model harms object detection results for the outdoor scenes, because state-of-the-art depth estimation networks generate indoor objects better than outdoor objects. Also, current depth sensors which are v1,v2 version of Kinect and Real Sense 1 work well in indoor scenes, so most of the RGB-D datasets published with the only indoor scenes. This affects outdoor scene depth generalization negatively. Hence, using estimated depth/HHA encoding provides improvement for only indoor scene object detection results. Object detection datasets contain both indoor and outdoor scenes such as Pascal VOC are affected negatively from our method due to poor quality of outdoor scenes' depth maps.

Like the depth map types, the depth map encoding has also effects on Faster R-CNN performance. According to the results of our experiments, we say that HHA encoding makes the most contribution to Faster R-CNN performance. The reason for this is that it contains richer information than other encodings. While the depth map contains only information about each pixel distance to a camera origin, HHA representations encode a disparity map, height above ground and angle with

gravity direction. Hence, more information about scenes provides a more accurate object detector.

Also, we conducted experiments to determine the best concatenation point for depth/HHA features. There are two possible concatenation points that increase Faster R-CNN performance. The first one is used in our method and Hou et al. work [19]. This is called the early concatenation that joins additional encoding features and RGB image features before the RPN layer. The other one is used at Cao et al. [3] and Gupta et al. [17] works. This is also called the late concatenation that joins additional features to RGB image features just before the bounding box and the classification layers. For estimated depth data, early concatenation should be chosen. Although there is not evident performance dominance between early and late concatenations for ground-truth depth data, early concatenation has the advantage of using fewer parameters than the late concatenation.

Finally, table 2.1 shows used backbone networks, datasets, mAp improvements for our and previous RGB-D works. As can be seen from table 2.1, this thesis provides more detailed research than previous works.

REFERENCES

[1] Shigeo Abe. *Support vector machines for pattern classification*, volume 2. Springer, 2005.

[2] Filippo Aleotti, Fabio Tosi, Matteo Poggi, and Stefano Mattoccia. Generative adversarial networks for unsupervised monocular depth prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[3] Yuanzhouhan Cao, Chunhua Shen, and Heng Tao Shen. Exploiting depth from single monocular images for object detection and semantic segmentation. *IEEE Transactions on Image Processing*, 26(2):836–846, 2017.

[4] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[5] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE Computer Society, 2005.

[6] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015.

[7] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *arXiv preprint arXiv:1406.2283*, 2014.

[8] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, and Wolfram Burgard. Multimodal deep learning for robust rgb-d object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 681–687. IEEE, 2015.

[9] Mark Everingham, L Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge 2007 (voc 2007) results (2007), 2008.

[10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[11] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.

[12] Ravi Garg, BG Vijay Kumar, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. *arXiv preprint arXiv:1603.04992*, 2016.

[13] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[15] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. *arXiv preprint arXiv:1609.03677*, 2016.

[16] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 564–571, 2013.

[17] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014.

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

[19] Saihui Hou, Zilei Wang, and Feng Wu. Object detection via deeply exploiting depth information. *Neurocomputing*, 286:58–66, 2018.

[20] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018.

[21] Junjie Hu, Mete Ozay, Yan Zhang, and Takayuki Okatani. Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. *arXiv preprint arXiv:1803.08673*, 2018.

[22] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[23] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[24] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[25] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. *arXiv preprint arXiv:1703.04309*, 2017.

[26] Christian Kerl, Jürgen Sturm, and Daniel Cremers. Robust odometry estimation for rgb-d cameras. In *2013 IEEE International Conference on Robotics and Automation*, pages 3748–3754. IEEE, 2013.

[27] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel realsense stereoscopic depth cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–10, 2017.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[29] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *2011 IEEE international conference on robotics and automation*, pages 1817–1824. IEEE, 2011.

[30] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2041–2050, 2018.

[31] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017.

[32] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[33] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. *arXiv preprint arXiv:1411.6387*, 2014.

[34] David G LOEW. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 2004.

[35] Ryuzo Okada, Yoshiaki Shirai, and Jun Miura. Object tracking based on optical flow and depth. In *1996 IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems (Cat. No. 96TH8242)*, pages 565–571. IEEE, 1996.

[36] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[37] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[38] Xiaofeng Ren, Liefeng Bo, and Dieter Fox. Rgb-(d) scene labeling: Features and algorithms. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2759–2766. IEEE, 2012.

[39] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. Learning depth from single monocular images. In *Advances in neural information processing systems*, pages 1161–1168, 2006.

[40] Ashutosh Saxena, Min Sun, and Andrew Y Ng. Learning 3-d scene structure from a single still image. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[41] Jamie Shotton, Ross Girshick, Andrew Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, Alex Kipman, et al. Efficient human pose estimation from single depth images. *IEEE transactions on pattern analysis and machine intelligence*, 35(12):2821–2840, 2012.

[42] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[44] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[45] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[46] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5038–5047, 2017.

[47] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[48] Jianwei Yang, Jiasen Lu, Dhruv Batra, and Devi Parikh. A faster pytorch implementation of faster r-cnn. *https://github.com/jwyang/faster-rcnn.pytorch*, 2017.

[49] Hyun Woo Yoo, Woo Hyun Kim, Jeong Woo Park, Won Hyong Lee, and Myung Jin Chung. Real-time plane detection based on depth map from kinect. In *IEEE ISR 2013*, pages 1–4. IEEE, 2013.

[50] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2):4–10, 2012.

[51] Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. T2net: Synthetic-to-realistic translation for solving single-image depth estimation tasks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 767–783, 2018.

[52] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. *arXiv preprint arXiv:1704.07813*, 2017.

[53] Youding Zhu, Behzad Dariush, and Kikuo Fujimura. Controlled human pose estimation from depth image streams. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8. IEEE, 2008.