Deep Learning Based RGB-D Vision Tasks

Yuanzhouhan Cao

A thesis submitted for the degree of
DOCTOR OF PHILOSOPHY
The University of Adelaide

February 2018

Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I give consent to this copy of my thesis, when deposited in the University Library, being made available for loan and photocopying, subject to the provisions of the Copyright Act 1968.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

Acknowledgments

First of all, I would like to extend my utmost gratitude to my principal supervisor, Prof. Chunhua Shen, for his in-depth guidance on my research. During the course of my Ph.D. study, he has provided me countless insightful discussions and suggestions. His intelligence, enthusiasm and dedication to cutting-edge research encourages me a lot. I would have never been able to finish this thesis without his supervision. It has been an honor to be one of his students. He has developed me into a research scientist. I am sure that the experience of working with him would have a positive impact on my future academic career.

It has also been fantastic to work with a number of generous and supportive collaborators. I would like to thank my co-supervisor Dr. Lingqiao Liu for all his expertise, assistance and patience. He has always been available whenever I need help. I would also like to thank Dr. Guosheng Lin who taught me about structured learning.

I thank my supportive friends and colleagues at the University of Adelaide. I would like to thank Yao Li, Ruizhi Qiao, Qichang Hu, Bohan Zhuang and Hui Li. They have made my experience as a Ph.D. candidate enjoyable. I would like to give special thanks to Tong Shen, who kindly assisted me with the MXNet implementations of my algorithms.

Last but not at least, I would like to express my greatest appreciation to my parents, for all their love and support.

Publications

This thesis is based on the content of the following peer-reviewed journal publications:

- **Yuanzhouhan Cao**, Chunhua Shen, Heng Tao Shen; "Exploiting Depth From Single Monocular Images for Object Detection and Semantic Segmentation", IEEE Transactions on Image Processing (TIP), 2016, DOI: 10.1109/TIP.2016.2621673. (presented in Chapter 3).
- **Yuanzhouhan Cao**, Zifeng Wu, Chunhua Shen; "Estimating Depth from Monocular Images as Classification Using Deep Fully Convolutional Residual Networks", IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), 2017, DOI: 10.1109/TCSVT.2017.2740321. (presented in Chapter 4).
- **Yuanzhouhan Cao**, Tianqi Zhao, Ke Xian, Chunhua Shen, Zifeng Wu, Zhiguo Cao; "Monocular Depth Estimation with Augmented Ordinal Depth Relationships", IEEE Transactions on Image Processing (TIP). (in peer review, presented in Chapter 5).

In addition, I have co-authored the following journal publication:

- Peng Wang*, **Yuanzhouhan Cao***, Chunhua Shen, Lingqiao Liu, Heng Tao Shen; "Temporal Pyramid Pooling Based Convolutional Neural Network for Action Recognition", IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), 2016, DOI: 10.1109/TCSVT.2016.2576761. ($*$ indicates equal contribution).

Abstract

Depth is an important source of information in computer vision. However, depth is usually discarded in most vision tasks. In this thesis, we study the tasks of estimating depth from single monocular images, and incorporating depth for object detection and semantic segmentation. Recently, a significant number of breakthroughs have been introduced to the vision community by deep convolutional neural networks (CNNs). All of our algorithms in this thesis are built upon deep CNNs.

The first part of this thesis addresses the task of incorporating depth for object detection and semantic segmentation. The aim is to improve the performance of vision tasks that are only based on RGB data. Two approaches for object detection and two approaches for semantic segmentation are presented. These approaches are based on existing depth estimation, object detection and semantic segmentation algorithms.

The second part of this thesis addresses the task of depth estimation. Depth estimation is often formulated as a regression task due to the continuous property of depths. Deep CNNs for depth estimation are trained by iteratively minimizing regression errors between predicted and ground-truth depths. A drawback of regression is that it predicts depths without confidence. In this thesis, we propose to formulate depth estimation as a classification task which naturally predicts depths with confidence. The confidence can be used during training and post-processing. We also propose to exploit ordinal depth relationships from stereo videos to improve the performance of metric depth estimation. By doing so we propose a Relative Depth in Stereo (RDIS) dataset that is densely annotated with relative depths.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The goal of computer vision is to enable machines to perceive the real world based on captured vision information as we human beings do. Most existing computer vision tasks are based on RGB only. We are living in a three-dimensional world. The captured RGB images are two-dimensional mappings of the three-dimensional world with depth information inevitably discarded. Undoubtedly, the performance of computer vision tasks such as object detection, semantic segmentation, scene understanding, etc., can be improved with successful incorporation of depth information. To this end, the acquisition of depth information from color images and the incorporation of depth information are fundamental and challenging research topics.

The acquisition of depth information is at the heart of many RGB-D vision tasks. The most straightforward way is to capture depths by commercial depth sensors such as the Microsoft Kinect and the Velodyne LiDAR. However, such depth sensors are not widely applied in vision community. Most training datasets are still RGB only. The captured depth maps are also limited in the diversity of scenes due to the limitation of depth sensors. For example, the Microsoft Kinect can not capture depths over 10 meters, and the Velodyne LiDAR laser scanner can be easily affected by strong sunlight.

Another option is to exploit depths from RGB images. Depth estimation is a notoriously ill-posed problem because one captured image scene may correspond to numerous real world scenarios. Prior works estimate depths from stereo images. These works manage to match key points of stereo images and estimate a disparity map. Given the parameters of stereo cameras, the depths can be obtained through simple geometry. With the monocular case arises in practice, most recent works focus on estimating depth from a single image. Previous works of monocular depth estimation are based on geometric assumptions. For example, the box models are applied to infer the spatial layout of a room Hedau et al. [2010]; Gupta et al. [2010b]. These methods are limited to geometric assumptions and can not be applied for general scenes. Other efforts explore non-parametric models Karsch et al. [2014] which consist of candidate images retrieval, scene alignment and then depth inference using smoothness constraints.

In recent years, the vision community has witnessed a series of breakthroughs introduced by the deep convolutional neural networks (CNNs) Krizhevsky et al. [2012]. Many monocular depth estimation algorithms employ deep CNNs and achieve outstanding performance. These algorithms train deep CNNs on large-scale annotated

datasets in an end-to-end fashion. Depth estimation is normally formulated as a regression task due to the continuous property of depths.

The acquisition of depth information is only one component of RGB-D vision tasks. The other component is the incorporation of acquired depths. One approach manages to recover the real 3D structures of objects in 3D point clouds. Earlier efforts focus on designing feature descriptors in 3D point clouds. These feature descriptors are hand-crafted for specific tasks. Recently some works apply 3D convolutional neural networks on voxelized shapes. However, volumetric representation is constrained by its resolution due to data sparsity and computation cost of 3D convolution. Another approach is to treat depths as one-channel images and aggregate to RGB images to formulate 2.5D data. Compared with the operations in 3D point clouds, this approach is easier to implement.

1.1 Problem formulation

In this thesis, we are interested in exploiting depth from single monocular images, and incorporating depth with RGB data for object detection and semantic segmentation. All of the tasks are based on deep convolutional neural networks (CNNs).

1.1.1 Depth estimation from single monocular images

Depth estimation is a dense prediction problem, the aim is to predict the depth value of each pixel in a given image. Due to the limitation of computational capacity, previous methods predict depths in terms of over-segmented superpixels. This is based on the assumption that scenes with semantically similar appearances should have similar depth distributions. The predicted depth maps usually contain strong inconsistencies. Benefiting from the recent success of deep CNNs, recent methods are able to directly predict the depth of each pixel. These methods directly regress on the depth due to the continuous property of depth. However, different points have different distributions of possible depth values. Depth estimation at some locations is easy while others are not. Typical regression models only output the mean values of possible depth values without the variances, (i.e., the confidence of a prediction is missing). In order to obtain the confidence of a depth prediction, we propose to formulate depth estimation as a classification task. The obtained confidence can be applied during both training and post-processing.

The keys to the success of deep CNNs are improved computational capacity and large-scale training datasets. Deep CNNs for monocular depth estimation are trained on a significant number of images densely labelled with metric depths. The acquisition of ground-truth metric depths requires depth sensors. Moreover, the ground-truth metric depths are limited in the size as well as the diversity of scenes. Another source of depth acquisition is the vast stereo videos. However in practice, the acquisition of metric depths from stereo videos is sometimes impracticable due to the absence of camera parameters. We proposed to acquire large amount of relative depths from stereo videos using stereo matching, and we show that the performance of monocular metric depth estimation can be improved by augmenting our collected relative depths.

1.1.2 Depth incorporation for object detection and semantic segmentation

Object detection and semantic segmentation are two important tasks in computer vision. The aim of object detection is to locate target objects in an image. Previous works on object detection exploit robust hand-crafted features such as SIFT Lowe [2004] and HOG Dalal and Triggs [2005]; Felzenszwalb et al. [2010] and generate image representations through Bag-of-Visual-Words (BoVW) Csurka et al. [2004]; Lazebnik et al. [2006]; Zhang et al. [2007] or Fisher Vector (FV) Perronnin et al. [2010]; Simonyan et al. [2013] encoding shcemes. Then discriminative models such as support vector machines (SVMs) Yu and Joachims [2009]; Joachims [1999] are learned as detectors. Recently, object detection benefits largely from deep convolutional neural networks. The R-CNN model proposed by Girshick et al. [2014] is one of the successful deep learning based object detectors. It first learns deep features of each region proposal, then trains a set of class-specific linear SVMs for object detection. In the later work of Fast R-CNN, Girshick [2015] proposed an end-to-end training scheme for deep learning based object detection. Compared with the R-CNN model, the Fast R-CNN model embeds a regions of interest (RoI) pooling layer that maps an RoI into a fixed-size feature map. And the linear SVMs are replaced with a classification loss at the end of the network. Based on the R-CNN and Fast R-CNN models, Ren et al. [2015a] proposed a Faster R-CNN model. It introduces a region proposal network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals.

The aim of semantic segmentation is to assign a unique label (or category) to every single pixel in the image. It is also a dense prediction problem. Traditional methods exploit contextual information. For example, the early work "TAS" Heitz and Koller [2008] models different types of spatial context between *Things* and *Stuff* using a generative probabilistic graphical model. The recent successful semantic segmentation methods are based on deep CNNs. Specifically, the fully convolutional neural networks (FCNNs) Long et al. [2015] have become a popular choice for semantic segmentation due to their their effective feature generation and end-to-end training. FCNNs have also been applied to other dense prediction tasks such as depth estimation, image restoration and image super-resolution.

The aforementioned algorithms for object detection and semantic segmentation are based on RGB only. We proposed to improve the performance of existing successful algorithms by incorporating exploited depths.

1.2 Main contributions

The main contributions of this thesis include a set of deep learning algorithms for RGB-D vision tasks (i.e., RGB-D object detection, RGB-D semantic segmentation and monocular depth estimation). All of our algorithms are based on state-of-the-art deep convolutional neural networks. More specifically, the main contributions are listed as follows.

- Two RGB-D object detection algorithms based on the recent success of monocular depth estimation and object detection. Specifically, we first employ the deep

convolutional neural field (DCNF) model Liu et al. [2015b] to acquire the depth map from an RGB image. Then we learn the depth and RGB features from the depth map and RGB image respectively. We combine the depth and RGB features for object detection. Our object detection algorithms are based upon the R-CNN Girshick et al. [2014] and Fast R-CNN Girshick [2015] models.

- Two RGB-D semantic segmentation algorithms. Similar to our RGB-D object detection algorithms, we first acquire the depth map from an RGB image with the DCNF model. The first algorithm exploits the depth feature from the depth map and combine it with the RGB feature for semantic segmentation. The depth and RGB information are fused at a later stage. The second algorithm uses the depth map for an additional loss calculation. The depth and RGB information are fused at an earlier stage.

- An algorithm for estimating depth from monocular images as classification. Most existing algorithms formulate depth estimation as a regression task due to the continuous property of depths. We formulate depth estimation as a classification task. Specifically, we first discretize the continuous ground-truth depths into several bins and label the bins according to their depth ranges. Then we solve the depth estimation problem as classification by training a fully convolutional deep residual network. Compared with estimating the exact depth of a single point, it is easier to estimate its depth range. More importantly, by performing depth classification instead of regression, we can easily obtain the confidence of a depth prediction in the form of probability distribution.

- A monocular depth estimation algorithm that employs ordinal depth relationships. We propose to improve the performance of metric depth estimation with relative depths collected from stereo videos using existing stereo matching algorithm. By doing so we introduce a new “Relative Depth in Stereo" (RDIS) dataset that densely labelled with relative depths. We first pretrain a deep residual network on our RDIS dataset. Then we finetune the network on RGB-D datasets labelled with metric ground-truth depths. We test our proposed method on both indoor and outdoor benchmark RGB-D datasets and achieve state-of-the-art performance.

1.3 Thesis overview

The rest of the thesis is organized as follows.

In Chapter 2, we first introduce some background information on convolutional neural networks (CNNs). Then we introduce some typical RGB-D vision tasks including the depth estimation, 2D-to-3D conversion and 3D vision.

In Chapter 3, we elaborate our proposed RGB-D object detection and semantic segmentation algorithms. In particular, we first apply the DCNF model to estimate depths from RGB images. We then derive two approaches for object detection and two approaches for semantic segmentation based on the estimated depths.

In Chapter 4, we elaborate our depth estimation by classification. We discretize the continuous depths into several bins and formulate depth estimation as a classi-

fication task. The classification naturally predicts depths with confidence. We apply the confidence of a depth prediction during training and post-processing.

In Chapter 5, we elaborate our monocular depth estimation algorithm which employs ordinal depth relationships. We first introduce how to extract relative depths from stereo videos to formulate our Relative Depth in Stereo (RDIS) dataset. Then we introduce how to improve the performance metric depth estimation with our RDIS dataset.

In Chapter 6, we conclude this thesis and list a few future directions.

Chapter 2

Literature Review

2.1 Convolutional neural networks

2.1.1 Introduction

The concept of neural networks (NNs) is built upon the back-propagation (BP) method proposed by Rumelhart et al. Rumelhart et al. [1986]. Three years after the BP method was proposed, LeCun et al. LeCun et al. [1989] proposed the earliest form of neural networks and applied to the task of handwritten zip code recognition. He showed that stochastic gradient descent via backpropagation was effective for training NNs. A conventional NN is a non-linear function that maps real-valued data to real-valued labels. It takes a single vector as input and each element of the vector forms one neuron in the input layer. These input data are then propagated to all neurons in the succeeding layer. Each hidden layer consists of a set of neurons and each neuron in the hidden layer connects to all neurons in the previous layer. The output of a neuron is generated by a weighted summation of the inputs from the previous layer followed by an activation function. The last fully-connected layer in a NN is referred to the output layer.

In 1998, LeCun et al. extended their work and proposed the first convolutional neural network (CNN): the LeNet-5 model for document recognition Lecun et al. [1998]. Compared to the conventional NNs, CNNs propose to implement the principle of weight sharing which remarkably reduces the number of parameters and thus increase the generalization capacity. Moreover, CNNs employ the pooling techniques to improve the robustness of CNN features to small translations and distortions in the input image.

CNNs saw heavy use in the 1990s, but then fell out of fashion with the rise of the support vector machines (SVMs) Yu and Joachims [2009]; Suykens and Vandewalle [1999]; Joachims [1999]. One important reason is that the training of CNNs is very time consuming. Some works are proposed to improve the training efficiency. Hinton et al. Hinton and Salakhutdinov [2006] proposed to reduce the dimensionality of data by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. In 2012, the interest in CNNs has been rekindled by Krizhevsky et al. Krizhevsky et al. [2012] thanks to the GPU parallel computing. They showed substantially higher image classification accuracy on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) Deng et al. [2009];

Russakovsky et al. [2015]. Their success resulted from training a large CNN on 1.2 million labelled images, together with a few twists on LeCun's LeNet-5 model. The study of CNNs and deep learning algorithms remains popular till today.

2.1.2 Basic building blocks

In this section, we briefly introduce some basic building blocks in CNNs including convolutional layer, pooling layer, batch normalization layer, fully-connected layer, ReLU layer and loss function.

- *Convolutional layer.* The convolutional layer is the essential building block of CNNs. Each convolutional layer consists of different numbers of muti-dimensional trainable filters that compute the convolution of the input volume. The number of filters matches the depth of input volume. In traditional image processing algorithms, the convolution filter works as a template that operates on an image and outputs a new image to manifest a certain type of feature. The traditional convolution filters are designed for different tasks. In CNNs, the weights of convolutional layers are learned from large labelled datasets and thus can be used for various tasks. During the forward pass of CNNs, each filter is convolved with local regions in a single input map at different spatial locations with a fixed stride and outputs a feature map. The size of the local region is determined by the height and width of the filter. The feature maps generated by all filters are concatenated in the channel dimension, which become the output of this convolutional layer and also serve as the input volume of the next layer. The CNNs are composed of different numbers of connected convolutional layers. Convolutional layers at different places can recognize different levels of features. The convolution operation in CNNs has two prominent properties: local connectivity and parameter sharing, which significantly reduce the number of parameters. Specifically, local connectivity refers to the fact that the convolution is applied to local regions rather than the entire input volume. And parameter sharing means that all the neurons on the same feature map share the convolution filter weights.

- *Pooling layer.* Pooling is one of the simplest operations in CNNs. It is a form of non-linearly down-sampling that operates on local regions in the input volume with a fixed stride. The values within the local region are aggregated into a single value. The most commonly applied poolings are max pooling and average pooling. The max pooling outputs the max value within the local region, while the average pooling outputs the average value. There are no weights to be learned in pooling layers. By performing the pooling operation, the size of feature maps is down-scaled and thus the computation of CNNs is reduced. More importantly, the pooling layer provides a form of translation invariance.

- *Batch normalization layer.* One of the difficulties of training CNNs is that the distribution of each layer's input changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower

learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities. The batch normalization layer Ioffe and Szegedy [2015] can alleviate this by performing normalization that fixes the means and variances of layer inputs for each training mini-batch. Batch normalization also has a beneficial effect on the gradient flow through the network, by reducing the dependence of gradients on the scale of the parameters or of their initial values. This allows us to use much higher learning rates without the risk of divergence. Furthermore, batch normalization regularizes the model and reduces the need for dropout Srivastava et al. [2014]. Finally, batch normalization makes it possible to use saturating nonlinearities by preventing the network from getting stuck in the saturated modes.

- *Fully-connected layer.* Fully-connected layers in CNNs are the same as those in conventional NNs, in which neurons have full connections to all activations in the previous layer. The activations in a fully-connected layer can be computed by a matrix multiplication followed by a bias offset. Specifically, $Y = XW^T + b$ where X is the input data of the fully-connected layer and W^T and b are the weights that need to be learned.

- *ReLU layer.* The rectified linear unit (ReLU) layer applies the non-saturating activation function $f(x) = max(0, x)$ to the input feature maps. It aims to increase the nonlinearity of the feature representation. Compared to other activation functions, the ReLU addresses the gradient exploding or vanishing problem during the training phase. Thus, it accelerates the CNN training phase without degrading the performance.

- *Loss function.* The final layers of CNNs are loss functions. The loss functions calculate different types of deviations between the the predicted values and the ground-truth labels. These deviations are backpropagated through the CNNs during training to update the network weights. Different loss functions are designed for different tasks. For classification tasks such as object detection and semantic segmentation, the softmax and sigmoid cross-entropy are two commonly applied loss functions. Softmax loss is used for predicting a single class of K mutually exclusive classes. Sigmoid cross-entropy loss is used for predicting K independent probability values in $[0, 1]$. For regression tasks such as depth estimation, the L_2 loss is commonly used. It calculates the squared mean difference between the predicted and ground-truth values.

2.1.3 Famous CNN architectures

In this section, we briefly introduce some renowned CNN architectures that are widely applied in the past years including AlexNet, VGGNet, ResNet and DenseNet.

- *AlexNet.* The ground-breaking AlexNet proposed by Krizhevsky et al. Krizhevsky et al. [2012] is the start of recent fast development of deep learning. It is the first CNN model that applies the ReLU and dropout layers. More importantly, it used a very efficient GPU implementation for the convolution operation. It

has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. The AlexNet is the winner of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) Deng et al. [2009]; Russakovsky et al. [2015] in 2012, which significantly outperformed the second runner-up.

- *VGGNet.* The VGGNet proposed by Simonyan et al. Simonyan and Zisserman [2014] obtained excellent performance in the ILSVRC 2014. Compared to the AlexNet, the VGGNet employs a relatively small (3×3) convolutional filters. The depth of the VGGNet can be increased up to 19 by stacking a sequence of these small filters. It shows the importance of the depths of CNNs for achieving better performance in many recognition and detection tasks.

- *ResNet.* Stacking more layers to CNN architectures does not necessarily improve performance as the training can become very difficult due to the problem of vanishing gradients. The residual network (ResNet) proposed by He et al. He et al. [2016a] manages to learn the residual mapping of a few stacked layers to avoid the vanishing gradients problem. Instead of directly learning the underlying mapping of a few stacked layers, the deep residual network learns the residual mapping. Then the original mapping can be realized by feedforward neural networks with "shortcut connections". Shortcut connections are those skipping one or more layers. With the residual learning scheme the ResNet can be increased up to 152 layers.

- *DenseNet.* The DenseNet proposed by Huang et al. Huang et al. [2017] connects all layers (with matching feature-map sizes) directly with each other. To preserve the feed-forward nature, each layer obtains additional inputs from all preceding layers and passes on its own feature maps to all subsequent layers. Compared to traditional CNNs, DenseNets have several compelling advantages: they alleviate the vanishing gradients problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters.

2.2 RGB-D vision tasks

Most existing computer vision tasks are mainly based on RGB data. The acquisition of RGB data is a mapping from real three dimensional world to two dimensional space with depth data inevitably discarded. The acquisition of depth data is one of fundamental tasks in computer vision field, and the performance of most vision tasks can be improved with incorporated depth information. In this section, we introduce some typical RGB-D vision tasks including depth estimation, 2D-to-3D conversion and 3D vision.

2.2.1 Depth estimation

Depth estimation is the fundamental topic of this thesis. Typical methods for depth estimation can be divided into two categories: single image depth estimation which is referred to as monocular depth estimation, and stereo image depth estimation which is referred to as binocular depth estimation.

- *Monocular depth estimation.* Depth estimation from single monocular images is an ill-posed problem as one captured image scene may corresponds to numerous real world scenarios. Earlier efforts on this task mainly exploit geometric assumptions such as box models to infer the spatial layout of a room Hedau et al. [2010]; Gupta et al. [2010b] or outdoor scenes Gupta et al. [2010a]. These models come with innate restrictions, which are limited to model only particular scene structures and therefore are not applicable for general scene depth estimations. Other methods formulate depth estimation as a markov random field (MRF) or conditional random field (CRF) learning methods. The depth estimation problem then is formulated as a maximum a posteriori (MAP) inference problem. For example, Saxena et al. Saxena et al. [2005] used a discriminatively trained MRF that incorporates multiscale local and global image features, and modeled depths at individual points and the relation between depths at different points. Liu et al. Liu et al. [2010] first performed semantic segmentation and applied the semantic labels as geometric constrains in CRFs.

  The aforementioned algorithms are based on simple models. They are easy to train but the performances are also limited. In recent years, depth estimation has benefited largely from the CNNs. Eigen et al. Eigen et al. [2014] are the first that apply deep CNNs to depth estimation. They addressesed this task by employing two deep network stacks: one that makes a coarse global prediction based on the entire image, and another that refines this prediction locally. In their later work of Eigen and Fergus [2015], they proposed to estimate depth, surface normal and semantic label in a single multiscale CNN architecture. Some recent works combine the the deep CNNs with CRFs for depth estimation. Liu et al. Liu et al. [2015b] proposed a deep convolutional neural field (DCNF) model which learned the unary and pairwise potentials of continuous CRFs in a unified deep network. Similarly, Li et al. Li et al. [2015a] and Wang et al. Wang et al. [2015] also combined the CNNs with CRFs, and formulated depth estimation in a two-layer hierarchical CRF to enforce synergy between global and local predictions.

- *Binocular depth estimation.* Binocular depth estimation manages to estimate depth from images captured by stereo cameras. Stereo cameras consists of a left camera and right camera mounted with a fixed distance B as illustrated in Fig. 2.1. The two views captured by stereo cameras are shifted with a distance d which is known as disparity. In traditional binocular depth estimation methods, given the left and right views, a disparity map is first computed by stereo

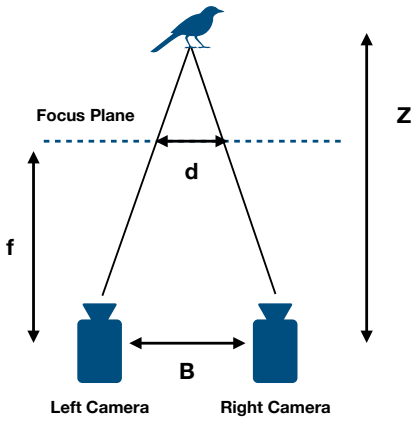

Figure 2.1: Illustration of binocular depth estimation. The objects are captured by two cameras mounted with a fixed distance. Binocular depth estimation first estimate a disparity map from the two views, then given the camera parameters, the depths can be calculated by geometry relations.

matching algorithms, then the depth Z can be computed with

$$Z = \frac{Bf}{B - d}. \tag{2.1}$$

Recently the performance of binocular depth estimation has been improved by CNNs. These methods use one view of the stereo pair as input and manage to reconstruct the other view and update network parameters by an image reconstruction loss. To name a few, Garg et al. Garg and Reid [2016] proposed to reconstruct the other view by Taylor approximation. Clément et al. Godard et al. [2017] embedded a spatial transform layer Jaderberg et al. [2015] in the network and applied the bilinear interpolation to reconstruct the other view. Since the labelled metric depths are absent during training, the performance of binocular depth estimation is not as good as monocular depth estimation. Moreover, in order to calculate depths from disparity maps, the camera parameter such as focus length is required.

2.2.2 2D-to-3D conversion

3D movies are getting popular in the past decade and the demand for 3D movies is still increasing. Moreover, in 2016, the virtual reality (VR) market has been ignited by modern vision techniques and shows great potential. The research for acquiring 3D content has becoming a hot topic. 3D videos are stored in stereoscopic format. For each frame, the format includes two views of the same scene, one of which is exposed to the viewer's left eye and the other to the viewer's right eye, thus giving the viewer the experience of watching the scene in three dimensions.

There are two approaches of producing 3D movies. The first is to shoot natively using stereo cameras. This type of approach costs too much to produce and may

cause special effects such as forced perspective. Forced perspective is an optical illusion technique that makes objects appear larger or smaller than they really are. It breaks down when viewed from another angle, which prevents stereo filming. The second approach is to shoot in 2D and convert to 3D, which is referred to as 2D-to-3D conversion. Traditional 2D-to-3D conversion methods contain two stages. Firstly, manually create a depth map for each 2D image Harman [2000]. Secondly, apply standard depth image based rendering (DIBR) algorithms Fehn [2004]; Nguyen et al. [2009] that combine the 2D image with the depth map to generate a stereo image pair. The first stage is processed by experienced workers from movie production companies, so this type of 2D-to-3D conversion methods is expensive and requires intensive human effort.

Recent research focus on automatic 2D-to-3D conversion. These works manage to learn disparity maps from 2D images in a supervised manner Zhang et al. [2011]. To name a few, Konrad et al. Konrad et al. [2013] developed two types of methods. The first is based on learning a point mapping from local image/video attributes, such as color, spatial position and, in the case of video, motion at each pixel, to scene-depth at that pixel using a regression type idea. The second method is based on globally estimating the entire depth map of a query image directly from a repository of 3D images (RGB-D pairs or stereo pairs) using a nearest-neighbor regression type idea. Appia et al. Appia and Batur [2014a] created a scene model for each image based on certain low-level features like texture, gradient and pixel location and estimated a pseudo depth map. In order to overcome imperfections and generate an enhanced depth map for improved viewing experience, certain high-level image features are also exploited.

The task of 2D-to-3D conversion is similar to binocular depth estimation in the sense that the disparity map needs to be calculated to generate a second view. However, 2D-to-3D conversion is different in two significant ways. Firstly, the disparity map does not need to be highly accurate in terms of metric measure, only need to be accurate to formulate a 3D view. Secondly, the calculation of disparity map is a middle stage and is not necessary. Most recent works embedded the depth rendering in CNNs and update the weights in an end-to-end way. Xie et al. Xie et al. [2016] proposed Deep3D, a fully automatic 2D-to-3D conversion algorithm. Specifically, they proposed a deep neural network that takes as input the left view, internally estimates a soft (probabilistic) disparity map, and then renders a novel right view. The internal disparity-like map produced by the network is computed only in service of creating a good right view. Similarly, Bae et al. Sung-Ho et al. [2017] proposed a fully convolutional depth rendering network which generates probabilistic disparity maps and renders a new view by multiplying the disparity maps with output color images. It takes various-sized images as input and outputs correspondingly-sized images.

2.2.3 3D Vision

The aforementioned tasks treat depths as one-channel images and simply aggregate to the RGB data. These tasks are known as 2.5D vision tasks. 3D vision tasks manage information in real 3D space, i.e., 3D point clouds. The 3D point clouds is a set of points in a three-dimensional coordinate system Eckart et al. [2016]; Elbaz et al.

[2017]; Ameesh et al. [2006]. These points are located on the external surfaces of visible objects. They are intended to represent 3D spatial information such as shape, surface, contour, etc., of the real world. Typical 3D vision tasks include 3D object detection, 3D pose estimation and 3D reconstruction.

- *3D object detection.* Earlier efforts on 3D object detection focus on exploiting geometric descriptors such as spin images Johnson and Hebert [1999], geometry histograms Frome et al. [2004], signatures of histograms Tombari et al. [2010], feature histograms Rusu et al. [2008] and cloud of oriented gradients Ren and Sudderth [2016]. These descriptors are combined with trainable models such as support vector machines Song and Xiao [2014] for object detection. Since they are manually designed for specific applications or 3D data types, it is often difficult for them to generalize to new data modalities. In recent years, the hand-crafted feature representations has been improved by deep CNNs that can learn powerful 3D and color features from the data. Wu et al. [2015] proposed 3D ShapeNets to represent a geometric 3D shape as a probabilistic distribution of binary variables on a 3D voxel grid. They applied convolutional deep belief network to learn the complex joint distribution of all 3D voxels in a data-driven manner. Gupta et al. [2015] proposed to represent objects in an RGB-D scene with corresponding 3D models. Zeng et al. [2017] presented a data-driven model named 3DMatch that learns a local volumetric patch descriptor for establishing correspondences between partial 3D data. Song and Xiao [2016] introduced Deep Sliding Shapes, a complete 3D formulation that takes a 3D volumetric scene from a RGB-D image as input and outputs 3D object bounding boxes. They proposed the first 3D region proposal network (RPN) that takes a 3D volumetric scene as input and outputs 3D object proposals, and the first joint object recognition network (ORN) to extract geometric features in 3D and color features in 2D.

- *3D pose estimation.* 3D pose estimation is a comprehensive task which contains object pose estimation Michel et al. [2016]; Brachmann et al. [2016]; Krull et al. [2015], human pose estimation Zhou et al. [2016]; Chu et al. [2017], hand pose estimation Ge et al. [2016]; Sinha et al. [2016], gaze estimation Funes-Mora and Odobez [2016], etc.. 3D pose estimation benefits largely from the deep CNNs. To name a few, for 3D human pose estimation, Pavlakos et al. [2017] proposed a fine discretization of the 3D space around the subject and train a CNN to predict per voxel likelihoods for each joint. They also employed a coarse-to-fine prediction scheme to improve upon initial estimates. Moreno-Noguer [2017] proposed a standard two-step pipeline for 3D human pose estimation. They first detected the 2D position of body joints by training a CNN-based detector, and then used these observations to infer the 3D pose. Chu et al. [2017] proposed to incorporate CNNs with a multi-context attention mechanism into an end-to-end framework for human pose estimation. The conditional random field (CRF) is also utilized to model the correlations among neighboring regions in the attention map. As for hand pose estimation, Wan et al. [2017] proposed a dual generative model that captures the latent spaces of hand poses and corresponding depth images for estimating 3D hand pose. The variational auto

encoder (VAE) and the generative adversarial network (GAN) are applied to model the generation process of hand poses and depth maps respectively. Ge et al. [2017] proposed a 3D CNN based hand pose estimation approach that can capture the 3D spatial structure of the input and accurately regress full 3D hand pose in a single pass.

- *3D reconstruction.* 3D reconstruction is the process of capturing the shape of objects in the real world. It is an important research topic in both computer vision and computer graphics. Generally, the 3D reconstruction methods can be divided into two categories: multi-view 3D reconstruction and single-view 3D reconstruction. The goal of multi-view 3D reconstruction is to infer geometrical structure of a scene captured by a collection of images. To name a few, Kolev et al. [2012] proposed a probabilistic model that computes the most probable surface that gives rise to the given observations. Qian et al. [2017] proposed a stereo-based 3D reconstruction approach for dynamic fluid surfaces. They also proposed a global optimization based approach that can recover both depths and normals of all 3D points simultaneously. The single-view 3D reconstruction is considered more difficult than multi-view 3D reconstruction due to the limited information. Most existing methods manage to exploit additional information. Toeppe et al. [2013] exploited the relative volume constraints for 3D reconstruction. The key idea is to formulate a variational reconstruction approach with shape priors in the form of relative depth profiles or volume ratios. Kong et al. [2017] used the locally corresponding CAD models for dense 3D reconstruction. They first employed an orthogonal matching method to rapidly choose the most corresponding CAD model, then employed a graph embedding based on local dense correspondence to allow for sparse linear combinations of CAD models.

Chapter 3

Exploiting Depth for Object Detection and Semantic Segmentation

3.1 Introduction

Since the formulation of color images are two-dimensional projections of the three-dimensional world with depth information inevitably discarded, depth information and color information are complementary in restoring the real three-dimensional scenes. It is not difficult to conjecture that the performance of vision tasks such as object detection and semantic segmentation is likely to be improved if we can take advantage of the depth information.

Depth has been proven to be a very informative cue for image interpretation in a significant amount of work Gupta et al. [2014]; Song and Xiao [2014]; Bo et al. [2011]. In order to exploit depth information, most of these algorithms require depth data captured by depth sensors, as well as camera parameters, to relate point clouds to pixels. However, despite recent development of a range of depth sensors, most computer vision datasets such as the ImageNet and the PASCAL VOC are still RGB only. Moreover, images within these datasets are captured by different cameras with no camera parameters available, rendering it unclear how to generate accurate point clouds.

In this chapter, we take advantage of deep convolutional neural networks (CNN) based depth estimation methods and show that the performance of object detection and semantic segmentation can be improved by incorporating an explicit depth estimation process. This may appear to be somewhat counter-intuitive, because the depth values estimated by CNN models are correlated to RGB images. The valuable depth information is acquired by depth sensors and lies in the RGB-D training data on which the depth estimation model is trained.

We here propose two methods to exploit depth information from color images for object detection and semantic segmentation. The two methods make use of estimated depth in different forms. The first method manages to learn deep depth features from the estimated depth. The learned depth features are combined with RGB features for both object detection and semantic segmentation. The depth information and color

information are fused at a later stage. For semantic segmentation, the estimated depth is used to compute an additional task loss. During training, the two losses jointly update the layers in an unified network in an end-to-end style. Thus, the depth information and color information are fused at an earlier stage.

Depth estimation from single monocular images is the first step of our proposed methods. Recently, CNNs have been applied to depth estimation and shown great success. For example, Eigen *et al.* Eigen et al. [2014] proposed to estimate depth using multi-scale CNNs. It directly regresses on the depth using CNN with two components: one estimates the global structure of the scene while the other one refines the structure using local information. Here, we follow the most recent work of Liu *et al.* Liu et al. [2015b] which trains a deep convolutional neural fields (DCNF) model for depth estimation. It jointly learns the unary and pairwise potentials of conditional random fields (CRF) in a single CNN framework, and achieved state-of-the-art performance.

To summarize, we highlight the main contributions of this chapter as follows.

1. For the tasks of object detection and semantic segmentation, we propose to incorporate estimated depth information with RGB data and improve the performance.

2. We propose two methods of exploiting depth from single monocular images for object detection and semantic segmentation.

3. We show that it is possible to improve the performance of object detection and semantic segmentation by exploiting relevant data which do not share the same set of labels.

The rest of the chapter is organized as follows. In section 3.2, we briefly review some related work. In section 3.3, we introduce the DCNF model which we use for depth estimation. In section 3.4 and section 3.5, we describe our RGB-D object detection and semantic segmentation frameworks respectively. Section 3.6 shows the experimental results and section 3.7 concludes this chapter.

3.2 Background

Our work is inspired by the recent progresses of depth estimation, vision with RGB-D data, object detection and semantic segmentation. In this section, we briefly review some most related work.

3.2.1 Depth estimation

Depth estimation from a single RGB image is the first step of our method. As part of the 3D structure understanding, traditional depth estimation methods are mainly based on geometric models. For example, the works in Hedau et al. [2010]; Gupta et al. [2010b]; Schwing and Urtasun [2012] rely on box-shaped models and fit the box edges to those observed in the image. These methods rely heavily on geometric

assumptions and fail to provide a detailed 3D description of the scene. Other methods attempt to exploit additional information. In particular, the authors of Russell and Torralba [2009] estimate depth through user annotations. The works of Liu et al. [2010]; Ladicky et al. [2014] make use of semantic class labels. Given the fact that the extra source of information is not always available, most of recent works formulate depth estimation as a Markov Random Fields (MRF) Saxena et al. [2005, 2009, 2007] or Conditional Random Fields (CRF) Liu et al. [2014] learning problem. These methods learn the parameters of MRF/CRF from a training set of monocular images and their ground-truth depth maps in a supervised fashion. The depth estimation problem is then formulated as maximum a posteriori (MAP) inference within the CRF model.

Most of the aforementioned algorithms use hand-crafted features such as texton, GIST, SIFT, PHOG, etc. With the fast development of DCNN recently, some works attempt to solve the depth estimation problem in a deep network and have achieved very impressive performance Eigen et al. [2014]; Liu et al. [2015b]; Li et al. [2015b]. In this article, we follow the deep conditional neural fields (DCNF) model introduced in Liu et al. [2015b] for depth estimation.

3.2.2 Incorporating depth

These methods can be roughly divided into two categories. The first one attempts to recover the real 3D shape of the scenes and explores 3D feature descriptors. To name a few, Song *et al.* Song and Xiao [2014] extended the deformable part based model (DPM) from 2D to 3D by proposing four point-cloud based shape features: point density feature, 3D shape feature, 3D normal feature and Truncated Signed Distance Function (TSDF) feature. Bo *et al.* Bo et al. [2011] developed a set of kernel features on depth images that model the size, 3D shape, and depth edges for object recognition. In Pepik et al. [2012], 3D information of object parts is treated as constrains for object detection. Sun *et al.* Sun et al. [2010] also estimate 3D shape of objects to improve object detection accuracy. Other algorithms make use of the context information such as inter-object relations or object-background relations Gould et al. [2008]; Stefan et al. [2010]; Lin et al. [2013]. These methods are able to provide a multi-view understanding of objects but also need large amounts of 3D shape training data.

The second category encodes the depth map as a 2D image and combines with the RGB image to formulate the 2.5D data. For example, Gupta *et al.* Gupta et al. [2014] proposed a depth map embedding scheme that encodes the height above ground, angle with gravity and horizontal disparity (HHA) as an additional input to RGB for object detection and semantic segmentation. Janoch *et al.* Janoch et al. [2011] extracted HOG features from depth images and trained a DPM model on these depth features for object detection. Schwarz *et al.* Schwarz et al. [2015] proposed an object-centered colorization scheme which is tailored for object recognition and pose estimation. All these methods require direct measurements of depths.

3.2.3 Object detection

Object detection has been an active research topic for decades. Conventional algorithms exploit hand-crafted feature descriptors such as SIFT Lowe [2004], HOG Dalal and Triggs [2005]; Felzenszwalb et al. [2010], and generate mid-level image representations through Bag-of-Visual-Words (BoVW) Csurka et al. [2004]; Lazebnik et al. [2006]; Sivic and Zisserman [2003]; Zhang et al. [2007] or Fisher Vector (FV) Perronnin et al. [2010]; Simonyan et al. [2013] encoding schemes. In recent years, CNN based methods have been demonstrated to outperform all hand-crafted features based algorithms. In Girshick et al. [2014], Girshick *et al.* combined regional proposals with CNN features and have achieved outstanding detection results. In their sequel work Girshick [2015], the authors proposed an region of interest (RoI) pooling layer and further improved the processing speed. In Zhang et al. [2015], Zhang *et al.* improved the CNN based object detection with Bayesian optimization and structured prediction. In Ouyang et al. [2015], Ouyang *et al.* proposed a deformable CNN for object detection. It effectively integrates feature representation learning, part deformable learning, context modeling, model averaging and bounding box location refinement into the detection system.

3.2.4 Semantic segmentation

Convolutional neural networks have also shown great potential in semantic segmentation Farabet et al. [2013]; Ning et al. [2005]; Pinheiro and Collobert [2014]. Recently, Long *et al.* Long et al. [2015] proposed a fully convolutional network (FCN) for semantic segmentation. It is the first work to train FCN end-to-end for pixelwise prediction. Many recent works combined DCNNs and CRFs for semantic segmentation and have achieved state-of-the-art performance. Schwing *et al.* Schwing and Urtasun [2015] jointly learned the dense CRFs and CNNs. The pairwise potential functions only enforce smoothness and are not CNN-based. Lin *et al.* Lin et al. [2016] jointly trained FCNs and CRFs and learned CNN-based general pairwise potential functions.

Unlike the aforementioned methods which are purely based on RGB information, our methods make use of depth information to improve objected detection and semantic segmentation performance.

3.3 Depth estimation model

We use the DCNF model introduced by Liu *et al.* Liu et al. [2015b] for depth estimation. It jointly learns the unary term and pairwise term of continuous CRF in an unified network. In this section, we first introduce continuous CRF model, then we introduce the DNCF model with fully convolutional networks and superpixel pooling.

3.3.1 Continuous CRF

Similar to Liu et al. [2010]; Saxena et al. [2009]; Liu et al. [2014], the images are represented as sets of small homogeneous regions (superpixels). The depth estimation is based on the assumption that pixels within a same superpixel have the same depth values. Each superpixel is represented by the depth value of its centroid. Let $\mathbf{x}$ be an image and $\mathbf{y} = [y_1, ..., y_n]^\top \in \mathbb{R}^n$ be a vector of depth values for all the superpixels, the conditional probability distribution of the data can be modelled as the following density function:

$$\Pr(\mathbf{y}|\mathbf{x}) = \frac{1}{\mathrm{Z}(\mathbf{x})} \exp(-\mathrm{E}(\mathbf{y}, \mathbf{x})), \tag{3.1}$$

where E is the energy function and Z is the partition function:

$$\mathrm{Z}(\mathbf{x}) = \int_{\mathbf{y}} \exp\{-\mathrm{E}(\mathbf{y}, \mathbf{x})\} \mathrm{d}_{\mathbf{y}}. \tag{3.2}$$

Since the depth value $\mathbf{y}$ is continuous, it is possible to calculate the partition function exactly. The depth value of a new image could be predicted through the following MAP inference:

$$\mathbf{y}^* = \underset{\mathbf{y}}{\mathrm{argmax}} \Pr(\mathbf{y}|\mathbf{x}). \tag{3.3}$$

The energy function is formulated as a typical combination of unary potentials U and pairwise potentials V over the nodes (superpixels) $\mathcal{N}$ and edges $\mathcal{S}$ of the image $\mathbf{x}$:

$$\mathrm{E}(\mathbf{y}, \mathbf{x}) = \sum_{p \in \mathcal{N}} \mathrm{U}(y_p, \mathbf{x}) + \sum_{(p,q) \in \mathcal{S}} \mathrm{V}(y_p, y_q, \mathbf{x}). \tag{3.4}$$

3.3.1.1 Unary potential function

The unary term U regresses the depth value from a single superpixel. It is formulated by the following least square loss:

$$\mathrm{U}(y_p, \mathbf{x}; \boldsymbol{\theta}) = (y_p - z_p(\boldsymbol{\theta}))^2, \forall p = 1, \ldots, n; \tag{3.5}$$

where z_p is the regressed depth value of superpixel p parametrized by parameters $\boldsymbol{\theta}$.

3.3.1.2 Pairwise potential function

The pairwise term V encourages neighbouring superpixels with similar appearances to have similar depth values. It is constructed by 3 types of similarities: color, color histogram and texture disparity in terms of local binary patterns (LBP), which are represented as follows:

$$\mathrm{S}_{pq}^{(k)} = \exp\{-\gamma \| s_p^{(k)} - s_q^{(k)} \|\}, k = 1, 2, 3; \tag{3.6}$$

where $s_p^{(k)}$ and $s_q^{(k)}$ are the observation values of superpixel p and q from color, color histogram and LBP respectively. $\| \cdot \|$ denotes the ℓ_2 norm of a vector and γ is a

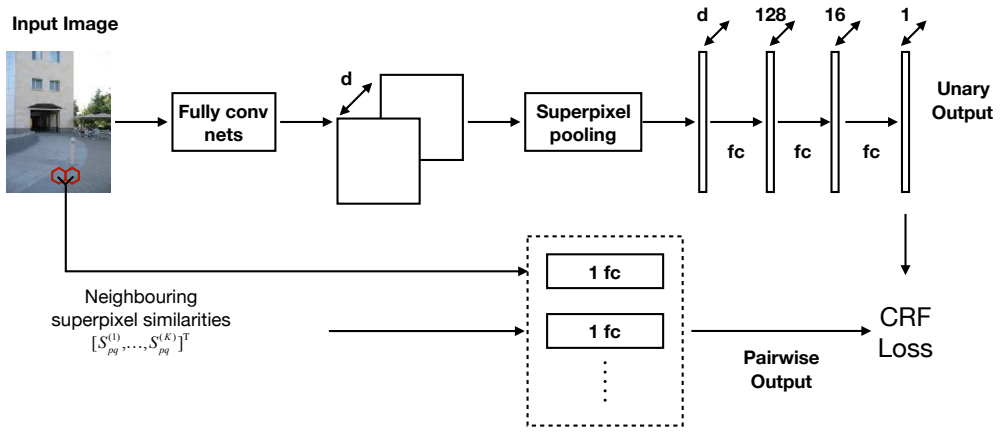

Figure 3.1: Overview of DCNF model. It is composed of a unary part and a pairwise part. The output of the unary part and the pairwise part are fed to the CRF structured loss layer, which minimizes the negative log-likelihood.

constant.

The similarity observation values are fed into a fully-connected layer:

$$\mathrm{R}_{pq} = \boldsymbol{\beta}^{\top}[\mathrm{S}_{pq}^{(1)},\ldots,\mathrm{S}_{pq}^{(k)}]^{\top} = \sum_{k=1}^{K}\beta_k \mathrm{S}_{pq}^{(k)}, \tag{3.7}$$

where $\boldsymbol{\beta} = [\beta_1,\ldots,\beta_k]^{\top}$ are trainable parameters. Finally, the pairwise potential function is formulated as:

$$\mathrm{V}(y_p, y_q, \mathbf{x}; \boldsymbol{\beta}) = \frac{1}{2}\mathrm{R}_{pq}(y_p - y_q)^2. \tag{3.8}$$

3.3.2 Deep convolution neural field model

The DCNF network is composed of a unary part, a pairwise part and a CRF loss layer, as is illustrated in Fig. 3.1. During training, an input image is first over-segmented into n superpixels. Then the entire image is fed into the unary part and outputs an n-dimensional vector containing regressed depth values of the n superpixels. Specifically, the unary part is composed of a fully convolution part and a superpixel pooling part. After fully convolution, an input image is convolved into a set of convolutional feature maps. The superpixel pooling takes the convolutional feature maps as the input and outputs n superpixel feature vectors. The n superpixel feature vectors are then fed into 3 fully-connected layers to produce the unary output.

The pairwise part takes similarity vectors (each with 3 components) of all neighbouring superpixel pairs as input and feeds each of them into a fully-connected layer (parameters are shared among different pairs), then outputs a vector containing all the 1-dimensional similarities for each of the neighbouring superpixel pairs. The continuous CRF loss layer takes the outputs from the unary and the pairwise terms to minimize the negative log-likelihood.

3.4 RGB-D object detection

We now describe how we exploit depth information from RGB images for object detection. Specifically, we build our models upon the R-CNN Girshick et al. [2014] and Fast R-CNN Girshick [2015] detection models and propose to learn depth features for each object proposal. We use the extracted depth features as extra information for object detection.

3.4.1 System overview

We design our RGB-D detection system based on two observations. Firstly, pixels within a single object have similar depth values. Secondly, the camera and the depth sensor have the same viewing angles in generating RGB-D datasets such as the NYUD2 and Make3D. As a result, an object in a depth image exhibits the same 2D shapes with its RGB counterpart, only with RGB values replaced by depth values (similar to intensity). Depth images can aggregate inter-object variations and eliminate intra-object variations. This is important for scene understanding, e.g., a chair appears in a painting or TV monitor should not be detected.

An overall structure of our RGB-D detection system is illustrated in Fig. 3.2. As we can see from the figure, it comprises 3 parts: depth estimation and encoding, RGB and depth feature extraction and object detection. During testing, we first estimate depth values of the input RGB image using the DCNF model trained on RGB-D dataset. With the estimated depth, we encode the estimated depth values into a 3-channel image. Similar to R-CNN and Fast R-CNN, we also follow the "recognition using regions" paradigm and generate object proposals on RGB images. We extract RGB and depth features of each object proposal through two separate streams. The RGB features capture intra-object information such as color and texture while the depth features aggregate 2D shapes of objects. Finally, we concatenate the two features for object detection.

3.4.2 RGB-D R-CNN object detector

The training of the R-CNN detector is a multi-stage pipeline. It first fine-tunes a CNN network pre-trained on a large classification dataset such as ImageNet. Then SVMs are trained on features extracted from the fine-tuned network in the first stage. These SVMs act as detectors.

3.4.2.1 Depth encoding and feature learning

The output of the DCNF model is a single-channel depth map. In order to extract features from the estimated depth map, we log-normalize these depth values to the range of $[0, 255]$ and duplicate into three channels. Here we do not apply any geometric contour cues. This is mainly because the calculation of geometric cues (normals, normal gradients, depth gradients, etc.) requires point clouds as well as accurate depth values. Since we use estimated depth values, outliers could affect the accuracy of geometric cues. Moreover, we do not access camera parameters for most of the color images which makes it impractical to recover the point clouds.

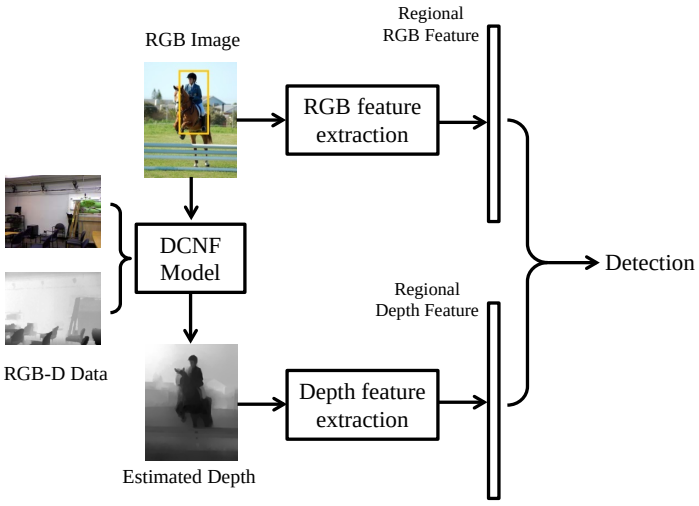

Figure 3.2: An overview of our RGB-D detection system. A DCNF model is first learned from RGB-D datasets. The input RGB image and its estimated depth image are fed into two feature extraction networks. The RGB feature and depth features of each object proposal are concatenated for object detection.

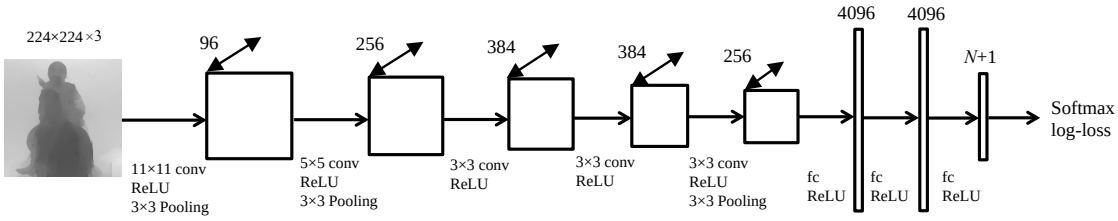

Figure 3.3: Detailed structure of the depth feature learning network of our RGB-D R-CNN detector. It consists of five convolutional layers and three fully-connected layers. During training, the input image patch is first resized to $224 \times 224 \times 3$ and fed into the network. The last layer calculates softmax log-loss and back-propagates through the network for parameter updating. We use the output of the first or second fully-connected layer which is a 4096-dimensional vector as the additional depth feature for object detection.

The CNN pre-trained on a large classification dataset can be used as a generic extractor of mid-level features Oquab et al. [2014]. Since RGB and depth images also share similar structures, we fine-tune a CNN pretrained on RGB images using depth images for depth feature learning. The structure of our regional depth feature learning network is illustrated in Fig. 3.3. It consists of 5 convolutional layers and 3 fully-connected layers interlaced with ReLU and max pooling layers. The last layer of the network is a classification layer with $N + 1$ channels, where N is the number of classes and the extra one for background. We initialize the parameters of our network with AlexNet Krizhevsky et al. [2012]. During fine-tuning, each object proposal is resized to $224 \times 224 \times 3$ and then fed into the network. For each class, we use object proposals that have $> 50\%$ intersection over union with the ground-truth boxes as positive training samples and the rest as negative training samples. During testing, we use the output of the fully-connected layers as depth feature.

3.4.2.2 Detector learning

The learnt depth features are 4096-dimensional feature vectors. We fine-tune another network using RGB images for RGB feature learning. The depth and RGB feature learning networks have the same structure but do not share parameters. For each of the object proposals, we concatenate the depth feature and RGB feature, which results in a 8192-dimensional feature vector. Since the RGB and depth images have the same scales and the two feature learning networks have the same structures, we do not normalize the features after concatenation.

After feature concatenation we train multiple one-vs.-all binary SVMs as object detectors. Training is done using liblinear Fan et al. [2008] with hyper-parameters $C = 0.001, B = 10, w_1 = 2$ where C is the SVM trade-off parameter. B is bias term and w_1 is the cost factor on hinge loss for positive examples. We use the same SVM hyper-parameters as in Girshick et al. [2014] and find that the final detection performance is not sensitive to these parameters. Following Girshick et al. [2014], the positive examples are from the ground truth boxes for the target class and the negative examples are defined as regions having $< 30\%$ intersection over union with the ground truth boxes. During training, we adopt standard hard negative mining. Hard negative mining converges quickly and in practice detection accuracy stops increasing after only a single pass over all images.

3.4.3 RGB-D Fast R-CNN object detector

As mentioned above, the training of our RGB-D R-CNN detector is a multi-stage pipeline. For the SVM training, both depth and RGB features are extracted from each object proposal in each image. As a result, training of R-CNN detector is expensive. The Fast R-CNN detector alleviated these disadvantages. It is based on fully convolutional networks which take as inputs arbitrarily sized images, and output convolutional spatial maps. Hence we propose the second RGB-D detection network based on Fast R-CNN.

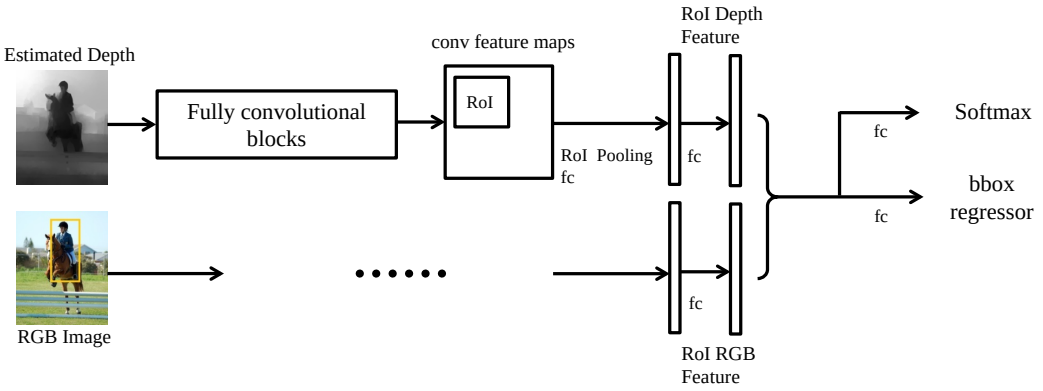

Figure 3.4: Detailed structure of our RGB-D Fast R-CNN network. It inputs the entire RGB and depth images and outputs two convolutional feature maps. After RoI pooling and fully-connected layers, we concatenate the RGB and depth features of each RoI and calculate a multi-task loss. The RGB and depth feature extraction networks have the same structure before concatenation. We omit the RGB feature extraction stream in the figure.

Block 1:	Block 2:	Block 3:	Block 4:	Block 5:
3×3 conv 64	3×3 conv 128	3×3 conv 256	3×3 conv 512	3×3 conv 512
3×3 conv 64	3×3 conv 128	3×3 conv 256	3×3 conv 512	3×3 conv 512
2×2 pooling	2×2 pooling	2×2 pooling	2×2 pooling	3×3 conv 512

Figure 3.5: Detailed structure of the fully convolutional blocks in Figs. 3.4, 3.7, 3.6. The parameters are transferred from VGG16.

3.4.3.1 Architecture

We show the architecture of our RGB-D Fast R-CNN network in Fig. 3.4. During training, the network takes as the input the entire RGB and depth images and a set of object proposals to produce two convolutional feature maps, one for RGB and one for depth. Then for each object proposal a region of interest (RoI) pooling layer extracts a fixed-length feature vector from the feature map. The RGB and depth feature vectors of each RoI are concatenated and fed into a sequence of fully connected layers that finally branch into two output layers: one that produces softmax probability estimates over $N+1$ classes (extra one for background) and the other one that outputs four real-valued numbers for each of the N object classes. The detailed structure of the fully convolutional blocks is illustrated in Fig. 3.5. We initialize our network with the VGG-16 net Simonyan and Zisserman [2014]. Before the feature concatenation, The RGB and depth streams in our RGB-D Fast R-CNN networks have the same structure but do not share parameters.

3.4.3.2 RoI Pooling and multi-task loss

An RoI is a rectangular window on a convolutional feature map. Each RoI is defined by a four-tuple (r, c, h, w) that specifies its top-left corner (r, c) and its height and width (h, w). The RoI pooling layer uses max pooling to convert the features inside any valid region of interest into a small feature map with a fixed spatial extent of $H \times W$, where H and W are layer hyper-parameters that are independent of any particular RoI.

During training, each RoI is labelled with a ground-truth class u and a ground-truth bounding-box regression target v. We use a multi-task loss L on each labelled RoI for joint training for classification and bounding box regression:

$$\mathrm{L}(p, u, t^u, v) = \mathrm{L}_{cls}(p, u) + \lambda[u \geq 1]\mathrm{L}_{loc}(t^u, v), \tag{3.9}$$

where $\mathrm{L}_{cls}(p, u) = -\log p_u$ is log loss for true class u. $[u \geq 1]$ evaluates to 1 when $u \geq 1$ and 0 otherwise. L_{loc} is defined over a tuple of true bounding-box regression targets for class u, $v = (v_x, v_y, v_w, v_h)$, and a predicted tuple $t^u = (t^u_x, t^u_y, t^u_w, t^u_h)$, again for class u. For bounding-box regression, the loss is:

$$\mathrm{L}_{loc}(t^u, v) = \sum_{i \in \{x,y,w,h\}} \mathrm{smooth}_{\mathrm{L}_1}(t^u_i - v_i), \tag{3.10}$$

in which

$$\mathrm{smooth}_{\mathrm{L}_1}(x) = \begin{cases} 0.5x^2 & \text{if} |x| < 1; \\ |x| - 0.5 & \text{otherwise.} \end{cases} \tag{3.11}$$

3.5 RGB-D semantic segmentation

In this section, we describe how we exploit depth for semantic segmentation. We first elaborate our RGB-D semantic segmentation with feature concatenation. Then we show the RGB-D semantic segmentation method which utilizes multi-task joint training.

3.5.1 RGB-D semantic segmentation by feature concatenation

Similar to our RGB-D detection methods, we extract RGB and depth features from RGB and depth images respectively and concatenate the features for semantic segmentation. Specifically, we follow the fully convolutional networks Long et al. [2015] which can take as inputs arbitrarily sized images. We show the network architecture in Fig. 3.6. After depth estimation, the RGB and estimated depth images are fed into two separate fully convolutional processing streams and output two convolutional feature maps. The two feature maps are concatenated and fed into 5 convolutional layers with channels $512, 512, 256, 128$ and $N + 1$ respectively where N is the number of classes and added 1 for background. Finally, a softmax layer is added during training to backpropagate the classification loss. We initialize our RGB-D segmentation network with the VGG-16 net Simonyan and Zisserman [2014] and the last 5

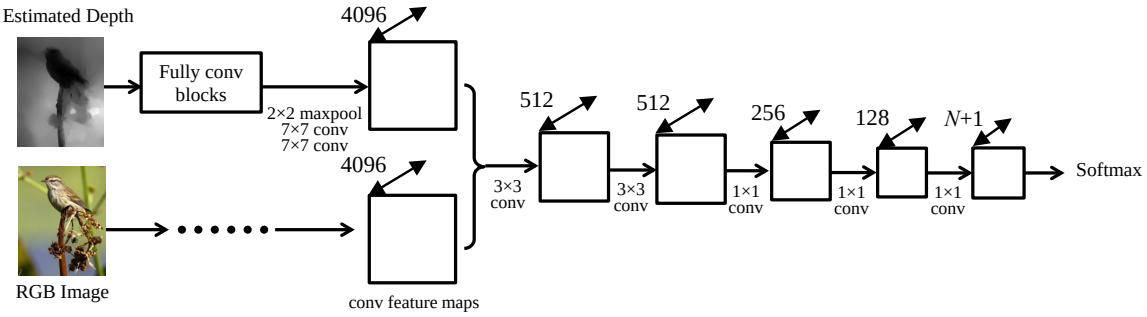

Figure 3.6: Network structure of our RGB-D segmentation with feature concatenation. The network takes the entire RGB and depth images as input and outputs two feature maps. We concatenate the RGB and depth feature maps and calculate softmax log-loss after several convolutional layers.

convolution layers are added. Before feature map concatenation, the RGB and depth processing streams have the same network structure but do not share parameters.

3.5.2 RGB-D semantic segmentation by multi-task training

The aforementioned method for RGB-D semantic segmentation has two separate networks for RGB and depth feature extraction. The two networks do not share parameters and thus are inefficient to train. Inspired by the Fast R-CNN detector which applies a multi-task training scheme, we propose another RGB-D semantic segmentation method. Since an RGB image has two labels after depth estimation: one semantic label and one depth label, we apply a multi-task loss which can jointly update a unified network in an end-to-end style.

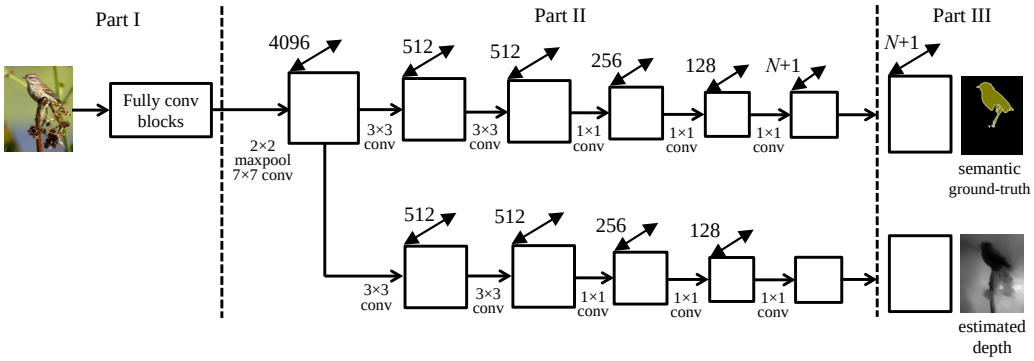

Figure 3.7: An overview of our RGB-D segmentation by multi-task training. During training, an entire RGB image is fed into the network and two loss functions are computed at the end of the network: softmax log-loss and regression loss. The two losses jointly update the network parameters in an end-to-end style. The detail of fully convolutional blocks is illustrated in Fig. 3.5.

3.5.2.1 Network architecture

The architecture of our multi-task RGB-D segmentation network is illustrated in Fig. 3.7. As we can see from the figure, the network can be broadly divided into 3 parts. The first part takes as input the entire RGB image and outputs convolutional feature maps. We initialize this part with the layers up to "fc6" in the VGG-16 net. During backpropagation, the semantic and depth loss share parameters here.

The second part has two sibling processing streams: one for color information and one for depth information. The color information processing stream contains 5 convolutional layers with channels $512, 512, 256, 128$ and $N+1$ respectively. The depth information processing stream also contains 5 convolutional layers. The only difference is that the last convolutional layer only has 1 channel due to different loss functions. The outputs of the second part are two feature maps of $N+1$ channels and 1 channel respectively. The multi-task loss part takes these two feature maps as input and calculates two losses: semantic label prediction loss and depth value regression loss. The two processing streams in the second part backpropagate the two losses through the entire network.

3.5.2.2 Multi-task loss

The last part of our segmentation network calculates a multi-task loss: softmax log-loss for semantic label prediction, and least squared loss for depth regression. The two losses jointly update a unified network in an end-to-end style.

Let $\mathbf{M}$ be the output feature map of color information processing stream in second part of our segmentation network. We first upscale $\mathbf{M}$ to be the same size of original input image using nearest neighbour interpolation. Assuming that size of original input image is $h \times w$, and d is the number of channels of the output feature map. The softmax log-loss computes

$$\mathrm{L}_{\mathrm{color}} = -\sum_{i,j}(\mathrm{M}_{ijc} - \log \sum_{k=1}^{\mathrm{d}} \mathrm{e}^{\mathrm{M}_{ijk}}), \tag{3.12}$$

where $i \in [1, h]$, $j \in [1, w]$, $k \in [1, d]$ and c is the ground-truth label.

The estimated depth values are used to compute the depth regression loss. Let $\mathbf{F}$ be the output feature map of depth processing stream in the second part of our segmentation network. Similar to the softmax log-loss, we upscale $\mathbf{F}$ to be the same size of original input image $h \times w$. The least squared loss computes:

$$\mathrm{L}_{\mathrm{depth}} = \sum_{i,j}(\mathrm{F}_{ij} - \mathrm{z}_{ij})^2, \tag{3.13}$$

where $i \in [1, h]$, $j \in [1, w]$ and z is the estimated depth values by DCNF model. Notably, different from DCNF model training, here we do not compute the least squared loss based on superpixels.

During backpropagation, the two separate losses are combined:

$$\mathrm{L} = \mathrm{L}_{\mathrm{color}} + \lambda \mathrm{L}_{\mathrm{depth}}, \tag{3.14}$$

where λ is a hyper-parameter to control the balance between depth information and color information in network updation.

3.6 Experiments

In this section, we show the performance improvement of object detection and semantic segmentation obtained by estimated depth. Our experiments are organized in two parts: RGB-D object detection and RGB-D semantic segmentation. We test the RGB-D object detection on 4 datasets: NYUD2 Silberman et al. [2012], B3DO Janoch et al. [2011], PASCAL VOC 2007 and 2012 and report the mean average precision (mAP). We test our RGB-D semantic segmentation on the VOC2012 dataset. The performance is measured by the intersection-over-union (IoU) score.

For depth prediction of indoor datasets such as the NYUD2 and B3DO, we train the DCNF model on the NYUD2 dataset. For the VOC2007 and VOC2012 datasets which contain both indoor and outdoor scenes, we train the DCNF model on images from both the NYUD2 and Make3D Saxena et al. [2007] datasets. The depths of the NYUD2 and B3DO are captured by the Microsoft Kinect, and the depths of the Make3D are captured by laser scanner.

3.6.1 RGB-D object detection

In this section we show the results of our RGB-D object detection. We first show the results of our RGB-D R-CNN object detector, then we show the results of our RGB-D Fast R-CNN object detector. Finally we analyse some key components in our experiments.

3.6.1.1 RGB-D R-CNN detection results

We first show the test results on the NYUD2 dataset, which we also use to train the DCNF model. The NYUD2 dataset has 1449 RGB-D image pairs captured by a Microsoft Kinect. We use the same object proposals in Gupta et al. [2014]. The RGB feature learning network is also from Gupta et al. [2014] which is fine-tuned on additional synthetic images. We use the training set to fine-tune our depth feature learning network. The detection results are shown in Table 3.1. Detectors' hyper-parameters are determined on the training and validation data set. We choose the output of different layers (denoted as pool5, fc6, fc7 which is same with AlexNet) as depth or RGB features. As we can see from the table, with the depth information being added, the best mAP is 25.2%, which is 5.5% higher than the result in Gupta et al. [2014].

We then use the same DCNF model to estimate depths of the B3DO dataset. The B3DO dataset is a relatively smaller RGB-D dataset than the NYUD2. It consists of 849 RGB-D image pairs captured by the Microsoft Kinect. The scene types of the B3DO dataset are mainly domestic and office which are very similar to the NYUD2. We generate around 2000 object proposals using selective search Uijlings et al. [2013] in each image. All the detectors are trained on the training set and tested on the validation set. We follow the detection baseline in Janoch et al. [2011] and report

Table 3.1: Detection results on the NYUD2 dataset. The first three columns are detection results of RGB features only. Columns 4-6 are detection results of depth features only. Columns 7-10 are detection results of combined RGB and depth features. The last two columns are detection results using features extracted from ground-truth depth. "pool5", "fc6", and "fc7" denote different layers for feature extraction, "scratch" denotes the depth feature learning network is trained from scratch.

	RGB (pool5)	RGB (fc6)	Gupta et al. [2014] RGB (fc7)	Depth (pool5)	Depth (fc6)	Depth (fc7)
bathtub	16.2	5.5	**18.3**	0.7	3.9	4.6
bed	38.4	**52.6**	40.7	35.9	**40.1**	39.3
bookshelf	22.8	19.5	**27.4**	**4.4**	4.1	2.0
box	0.8	**1.0**	0.8	0.1	0.1	0.1
chair	23.3	24.6	**27.1**	11.7	14.6	13.9
counter	27.8	20.3	**34.2**	15.4	19.9	20.0
desk	5.2	6.7	**8.8**	3.2	**4.3**	3.4
door	13.3	14.1	**15.0**	1.7	**1.9**	1.9
dresser	10.3	16.2	**17.0**	4.3	7.0	**7.3**
garbage bin	15.0	**17.8**	16.4	2.6	**4.3**	4.1
lamp	24.0	12.0	**25.9**	10.2	10.2	**10.3**
monitor	28.8	32.6	**37.4**	5.3	**7.6**	6.8
nightstand	11.8	**18.1**	12.4	0.9	0.9	**1.1**
pillow	13.0	10.7	**14.5**	3.7	5.5	**7.0**
sink	21.2	6.8	**25.7**	3.6	7.6	**9.4**
sofa	**29.0**	21.6	28.6	11.5	**13.9**	12.3
table	9.7	10.0	**11.2**	7.8	**10.3**	9.6
television	23.9	**31.6**	26.0	**4.4**	3.9	4.3
toilet	40.0	**52.0**	44.8	**29.5**	27.6	26.9
mAP	19.7	19.7	**22.7**	8.3	**9.9**	9.7

	RGB-D (pool5)	RGB-D (fc6)	RGB-D (fc7)	RGB-D (fc6,scratch)	GT Depth	RGB-GTD (fc6)
bathtub	15.4	19.3	**19.8**	18.5	23.9	**39.1**
bed	51.4	**52.5**	49.1	45.6	58.8	**65.4**
bookshelf	26.0	**30.5**	29.0	28.1	27.7	**34.2**
box	0.8	0.7	0.8	0.6	0.4	**0.8**
chair	25.3	**28.8**	28.6	28.2	31.7	**39.6**
counter	32.4	36.2	**37.2**	34.9	34.5	**45.2**
desk	8.6	**11.2**	10.4	8.2	3.8	**11.8**
door	13.5	14.1	15.4	**15.5**	3.6	**18.3**
dresser	16.1	**25.0**	20.1	17.1	14.0	**25.4**
garbage bin	17.8	**20.9**	17.6	16.5	13.2	**28.3**
lamp	22.3	27.4	**27.8**	26.9	27.1	**34.1**
monitor	31.5	36.5	**38.4**	37.1	7.0	**38.4**
nightstand	**14.0**	10.8	13.2	12.2	23.3	**31.4**
pillow	16.3	**18.3**	17.5	16.8	15.2	**27.1**
sink	20.0	26.1	26.3	**28.1**	20.0	**36.2**
sofa	30.6	**31.8**	29.3	27.7	28.7	**38.2**
table	14.5	**15.0**	14.4	11.7	21.1	**21.4**
television	27.6	29.3	28.4	**29.5**	7.8	**26.4**
toilet	42.8	**44.8**	44.2	40.2	42.2	**48.2**
mAP	22.5	**25.2**	24.6	23.3	21.3	**32.1**

the test results on 8 objects. But different from Janoch et al. [2011] which uses 6 different splits for evaluation and reported the averaged AP values, we only use the first split (with 306 images in the training set and 237 images in the validation set). We directly use the AlexNet without fine-tuning to extract RGB features. The results are illustrated in Table 3.2, from which we can see that the added depth features improve the detection mAP by 2.0%.

Table 3.2: Detection results of our RGB-D R-CNN detector on the B3DO dataset. The first row shows the results of RGB features only; the second row shows the results of estimated depth features only; the third row shows the results of combined RGB features and depth features.

	RGB	Estimated Depth	RGB-D (Estimated)
chair	37.0	35.9	**43.3**
monitor	79.0	59.9	**78.6**
cup	33.0	21.8	**33.8**
bottle	19.5	6.1	**20.2**
bowl	29.6	14.7	**34.3**
keyboard	56.5	15.2	**57.0**
mouse	26.9	9.0	**24.2**
phone	37.9	8.4	**44.0**
mAP	39.9	21.4	**41.9**

In order to further show the the effectiveness of our deeply learned depth features, we test object detection on the PASCAL VOC 2007 and 2012 datasets which have no measured depths. We use the same regional proposals in Girshick et al. [2014] which are generated by selective search. We first report the results on 6 indoor objects in the VOC2012 dataset in Table 3.3. We select all the images containing these 6 objects and use the training set (1508 indoor images) to train the depth feature learning network and detector, and use the validation set (1523 indoor images) for testing. We directly use the AlexNet without finetuning for RGB feature extraction and use the output of the “fc6" layer as feature vectors. As we can see from Table 3.3, our learned depth features improve the detection mAP by 3.6%.

Table 3.3: Detection results of our RGB-D R-CNN detector on indoor objects of the VOC2012 dataset. The first row shows the results of RGB features only, the second row shows the results of estimated depth features only, the third row shows the results of combining RGB features and depth features.

	bottle	chair	table	plant	sofa	tv	mAP
RGB	20.0	16.4	23.2	21.6	24.8	55.1	26.9
Estimated Depth	11.4	11.0	16.0	4.7	20.2	32.7	16.0
RGB-D (Estimated)	**21.5**	**20.8**	**29.6**	**24.1**	**31.7**	**55.1**	**30.5**

Table 3.4 shows the detection results on the VOC2007 dataset. Since we use all images in the VOC2007 dataset, we train the DCNF model using a combination of

Figure 3.8: Some detection examples. The red boxes are instances detected by the RGB detector, and the yellow boxes are instances obtained by our RGB-D detector.

the NYUD2 and Make3D datasets. We first directly use AlexNet without fine-tuning for the RGB feature extraction. As we can see from the first 3 rows, the additional depth features improve the mAP by 3.2%. We then use fine-tuned AlexNet for the RGB feature extraction and report the results in the last two rows. As we can see, even though the RGB fine-tuning increases correlativity between color and depth information, the additional depth features still improve the detection mAP by 0.9%. Similar to the experiment on the VOC2012 dataset, we use the output of the "fc6" layer as the RGB and depth features. Some examples of our RGB-D detection are shown in Fig. 3.8.

3.6.1.2 RGB-D Fast R-CNN detection results

We test our RGB-D Fast R-CNN detection on the VOC2007 dataset and show the results on Table 3.5. We use the object proposals generated by the region proposal network (RPN) network in Ren et al. [2015b]. Similar to Ren et al. [2015b], we use 2000 RPN proposals in each image to train and test on 300 RPN proposals in each image. In order to fit the GPU memory, we use single-scale training $s = 300$ pixels and cap the longest image side at 500 pixels (600 and 1000 respectively in Girshick [2015]). We also use a mini-batch size of 1 and only sample 32 RoIs from one image. The experiments in Girshick [2015] use a mini-batch size of 2 and sample 128 RoIs from 2 images. As we can see from Table 3.5, that our RGB-D detection mAP outperforms RGB only detection mAP by 1.0%. Notably, our RGB only detection results is lower than Ren et al. [2015b]. This is mainly because we shrink the input image size and especially the mini-batch size as the RoIs in one image are correlated.

Additionally, we can also learn a region proposal network (RPN) network with our estimated depth incorporated. Specifically, we can apply a two steam structure with RGB and depth images as two separate inputs. After two processing streams that do not share parameters, the RGB and depth feature maps are concatenated to generate object proposals. With the estimated depth being incorporated in RPN network learning, our RGB-D detection can also be applied on the latest Faster-RCNN.

Table 3.4: Detection results on the VOC 2007 dataset of our RGB-D R-CNN detector. The first column shows the results of RGB features only, the second column shows the results of depth features only, the third column shows the results of combined RGB and depth features. The first three columns are results of RGB features directly extracted from AlexNet and the last two columns are the results of RGB features extracted from fine-tuned AlexNet.

	RGB Girshick et al. [2014]	Depth	RGB-D	RGB Girshick et al. [2014] (finetune)	RGB-D (finetune)
aero	59.3	39.1	**61.2**	63.5	**67.7**
bike	61.8	33.7	**63.4**	66.0	**66.4**
bird	43.1	21.2	**44.4**	**47.9**	47.8
boat	34.0	15.6	**35.9**	**37.7**	36.0
bottle	25.1	10.9	**28.2**	29.9	**31.9**
bus	53.1	36.1	**55.6**	**62.5**	61.5
car	60.6	43.2	**63.7**	70.2	**70.6**
cat	52.8	30.7	**56.5**	**60.2**	59.0
chair	21.7	12.0	**26.9**	32.0	**32.6**
cow	47.8	21.3	**49.5**	**57.9**	52.3
table	42.7	27.7	**47.1**	47.0	**51.4**
dog	47.8	14.6	**49.7**	53.5	**55.9**
horse	52.5	29.5	**56.2**	60.1	**61.1**
mbike	58.5	28.4	**59.9**	**64.2**	63.8
person	44.6	29.1	**46.3**	52.2	**52.8**
plant	25.6	10.3	**28.3**	31.3	**32.6**
sheep	48.3	13.0	**50.4**	55.0	**57.7**
sofa	34.0	25.4	**46.8**	50.0	**50.5**
train	53.1	31.9	**55.4**	57.7	**58.1**
tv	58.0	30.4	**62.7**	63.0	**64.8**
mAP	46.2	25.2	**49.4**	53.1	**54.0**

3.6.1.3 Ceiling performance

Since we use estimated depth for depth feature extraction, we need to find out whether there is any space for further improvement of detection by improving depth estimation accuracy. We test our RGB-D R-CNN object detection using ground-truth depth on the NYUD2 dataset and show the results in the last two columns in Table 3.1. As we can see from Table 3.1, the detection mAP of ground-truth depth only is 21.3%, which is even better than the RGB only mAP. The mAP by ground truth RGB-D outperforms the mAP by estimated RGB-D by 6.9%. From this experiment we can see that depth estimation is a key component of our RGB-D detection methods, the detection performance can be further improved by improving depth estimation accuracy.

3.6.1.4 Network initialization

In the aforementioned RGB-D R-CNN experiments, we initialize our depth feature learning network with AlexNet and fine-tune it using depth images based on the

Table 3.5: Detection results on the VOC 2007 dataset of our RGB-D Fast R-CNN detector. The first row shows the results of RGB features only; the second row shows the results of depth features only, and the last two rows show the result of RGB-D features.

	RGB	Depth	RGB-D (fc6)	RGB-D (fc7)
aero	65.3	47.8	60.9	**65.3**
bike	74.3	46.7	**76.7**	75.8
bird	60.3	26.8	**60.7**	59.6
boat	48.7	27.7	46.8	**48.3**
bottle	36.2	14.3	35.5	**36.8**
bus	74.0	54.7	73.6	**74.3**
car	75.7	56.6	**76.4**	75.8
cat	77.9	49.5	**79.4**	78.7
chair	40.5	21.5	**41.5**	41.4
cow	66.1	27.2	69.8	**70.5**
table	54.6	40.7	57.3	**57.5**
dog	73.8	35.1	75.5	**75.9**
horse	77.7	53.7	78.2	**79.1**
mbike	73.1	45.2	68.7	**68.8**
person	67.5	47.8	70.6	**71.7**
plant	35.4	14.7	**34.4**	33.5
sheep	57.6	17.5	**60.6**	58.6
sofa	62.7	45.3	**64.8**	63.4
train	74.3	51.2	76.1	**76.6**
tv	57.1	38.8	**62.6**	59.7
mAP	62.6	38.1	63.5	**63.6**

similarities among RGB and depth images. We also duplicate depth map to 3 channels to fit the input of the pre-trained AlexNet. In order to show the effectiveness of this step, we train a depth feature learning network from scratch on a larger RGB-D dataset and test object detection on the NYUD2 dataset. Specifically, we use SUN-RGBD Song et al. [2015] which contains a total number of 10335 images. Since we report detection results on the NYUD2 dataset, we use all the images exclude the 654 NYUD2 test images. We use selective search to generate around 2000 object proposals in each image. We show the results in Table 3.1 denoted as RGB-D (fc6,scratch). As we can see, AlexNet initialization works better than training depth learning network from scratch. This experiment confirms that there are certain similarities between RGB and depth images and we can take advantage of networks pre-trained on RGB datasets to exploit depth information.

In addition, we also replace the depth features with RGB features learned from another network. Specifically, we initialize another RGB feature learning network with Place-AlexNet Zhou et al. [2014] which is trained on the Place dataset with 2.5 million images. We fine-tune this network also with RGB images to extract new RGB features (output of fc6). Then the two streams in are RGB features. We test detection with the two RGB features and get an mAP of 21.9%. This experiment

further confirms that the information exploited from depth images is useful for RGB only tasks.

3.6.2 RGB-D semantic segmentation

In this section, we first show the results of our RGB-D segmentation by feature concatenation, then we show the result of our RGB-D segmentation by multi-task training. All the segmentation results are reported on the validation of the VOC2012 dataset which contains 1449 images. The standard segmentation training set of VOC2012 has 1464 images. Following the conventional setting in Hariharan et al. [2014], we also report the segmentation results using the 10582 training images augmented in Hariharan et al. [2011].

Table 3.6: Segmentation results of our RGB-D segmentation by feature concatenation on the VOC2012 dataset. We use the standard 1464 images for training and test on the validation set. The first row shows the results using RGB features only, and the last 3 row shows the results with additional depth features.

	RGB	RGB-D (pool5)	RGB-D (fc6)	RGB-D (fc7)
aero	52.3	57.7	**61.8**	61.5
bike	23.2	15.5	**25.7**	24.0
bird	41.8	50.5	**58.5**	57.1
boat	34.5	38.8	**47.5**	44.8
bottle	36.9	41.6	**49.7**	47.9
bus	60.4	60.8	**67.3**	66.9
car	56.3	58.1	65.7	**66.1**
cat	53.4	59.1	**65.1**	63.9
chair	11.9	13.9	15.6	16.0
cow	14.4	33.6	42.5	**43.1**
table	32.5	29.9	36.5	**36.8**
dog	44.5	48.7	**55.9**	54.8
horse	33.4	33.3	**44.9**	44.0
mbike	50.3	52.0	**58.1**	56.7
person	63.1	65.0	**67.0**	66.8
plant	23.4	27.9	**34.2**	33.0
sheep	41.4	45.4	54.0	**54.3**
sofa	14.7	26.4	**31.2**	29.7
train	48.2	52.6	**60.8**	60.2
tv	45.0	43.3	**48.5**	47.6
mean	41.4	44.8	**51.4**	50.6

3.6.2.1 RGB-D segmentation by feature concatenation

We first train our RGB-D segmentation network using the standard 1464 images and show the results in Table 3.6. Similar to our RGB-D detection experiments, we also concatenate the RGB and depth features at different places (denoted as pool5, fc6,fc7 which is same with VGG16 net) and report the results. As we can see from the

Table 3.7: Segmentation results on VOC2012 dataset. We use the augmented data for training and test on validation set. The first row shows the results using RGB images only The second row shows the result of our RGB-D segmentation by multi-task training. The last 2 rows show the results of our RGB-D segmentation by feature concatenation.

	RGB	RGB-D (multi-task)	RGB-D(fc6)	RGB-D(fc7)
aero	69.2	**69.5**	**70.9**	69.4
bike	**31.0**	30.2	**29.9**	29.9
bird	64.5	**68.4**	**68.5**	68.3
boat	48.6	**53.3**	54.9	**55.7**
bottle	49.9	**54.1**	**55.8**	53.5
bus	**73.2**	72.8	**75.0**	73.2
car	68.3	**70.3**	**72.0**	71.4
cat	72.2	**72.2**	**72.6**	72.4
chair	**24.2**	22.4	21.2	**22.6**
cow	50.9	**54.6**	**53.5**	53.1
table	34.2	**36.5**	**37.5**	37.0
dog	61.2	**63.5**	63.4	**63.9**
horse	52.9	**55.9**	**56.2**	55.3
mbike	**64.4**	63.7	63.4	**63.7**
person	**72.0**	71.5	**72.6**	71.7
plant	**41.7**	40.7	**43.7**	42.7
sheep	59.1	**60.2**	**61.6**	61.6
sofa	27.5	**31.5**	32.6	**33.2**
train	67.9	**69.7**	**71.2**	70.4
tv	57.7	**59.5**	**60.0**	57.7
mean	56.2	**57.6**	**58.4**	57.9

table, all the results with depth features outperform the RGB only result. And the maximum mean IoU score improvement is 10% when we choose the output of fc6 layer to be the depth feature.

Next we train our RGB-D segmentation network using augmented images and report the result in Table 3.7. From the table we can see that the maximum mean intersection over union (IoU) score improvement is 2.2%, which is much less comparing to the improvement by standard training data. This is caused by the high correlativity between the estimated depth maps and the RGB images. When the RGB training images are sufficient, there is less additional information the depth maps can provide.

3.6.2.2 RGB-D segmentation by multi-task training

In our RGB-D segmentation network by feature concatenation, the separate feature extraction networks minimize the correlativity between the estimated depths and the RGB images. While in our RGB-D segmentation network by multi-task training, since the depth and color information share parameters, it is a question how much

the estimated depth can help in updating the network.

During back-propagation, there are two factors controlling the extent of the participation of depth information in updating the network parameters. The first one is a trade-off hyper-parameter λ multiplied by the lease squared loss. When λ is 0, the depth information contributes nothing to the parameter updating. The second one is the number of convolution layers n in the depth information processing stream. Fig. 3.7 shows an example of $n = 5$. We also test $n = 2$ and $n = 3$. When $n = 2$, the depth information processing stream contains 2 convolution layers with 128 and 1 channels respectively. When $n = 3$, the depth information processing stream contains 5 convolution layers with channels 256, 128 and 1 respectively. This factor controls the number of common layers that shared by color and depth information in the network.

We first use the standard 1464 images for training and report the test results in Table 3.8. As we can see from the table, the maximum improvement introduced by depth information is 9%. We also notice that smaller participation of depth information in updating network parameters (larger value of n and smaller value of λ) lead to better performance. This is because the smaller participation of depth information, the less correlativity between the depths and RGB images.

Combining losses is often hard to tease apart from adjusting the learning rate or other regularizations, since adding a loss effectively changes the gradient size. In the aforementioned experiments, we initialize the learning rate (lr) to be 10^{-7} and decrease by a factor of 0.4 every 5 epochs. The weight decay (wd) is set to 0.0005. In order to make sure the performance improvement is introduced by the exploited depth information, we also test on 5 other different learning rate and weight decay schemes: (a) $wd = 0.001$, initial $lr = 10^{-7}$ and decrease 0.4 every 5 epochs, (b) $wd = 0.0001$, initial $lr = 10^{-7}$ and decrease 0.4 every 5 epochs,(c) $wd = 0.0005$, initial $lr = 10^{-7}$ and decrease 0.1 every 5 epochs, (d) $wd = 0.0005$, initial $lr = 10^{-7}$ and decrease 0.4 every 7 epochs, (e) $wd = 0.0005$, initial $lr = 10^{-8}$ and decrease 0.4 every 5 epochs. We get the mean IoU score of 50.2%, 50.3%, 50.4%, 50.1% and 48.8% respectively. These results confirm that the performance improvement is introduced by exploited depth information.

Table 3.8: The VOC2012 segmentation IoU scores of our method on the *validation* set using the standard *training* set for training. The coefficient λ is a trade-off multiplied by the depth error in the back-propagation. n is the number of fully-connected layers in the depth processing stream.

	$\lambda = 0$	$\lambda = 0.1$	$\lambda = 1$	$\lambda = 10$	$\lambda = 50$
$n = 2$	41.4	50.2	49.9	49.1	46.8
$n = 3$	-	50.0	50.2	49.4	49.8
$n = 5$	-	50.0	**50.4**	49.9	48.3

Next we we show the results using augmented training images in Table 3.7. We set $\lambda = 10$ and $n = 3$. As we can see, the mean IoU score is only improved by 1.4% when incorporating depth information. Similar to the result of our RGB-D

segmentation by feature concatenation, the improvement is much less when data augmentation is applied due to the limited additional information provided by the estimated depth.

3.7 Conclusion

We have combined the task of depth estimation with object detection and semantic segmentation and proposed two ways of exploiting depth information from estimated depth to improve the performance of object detection and semantic segmentation. Experiment results show the effectiveness of our proposed methods.

We have attempted to answer the question of whether separately estimating depths from RGB images and incorporating them as a cue for detection and segmentation improves performance, and shown that it does. This holds despite the fact that the depth estimation process only has access to the same test data as the main (detection or segmentation) algorithm. This is particularly interesting because it shows that it is possible to improve performance by exploiting related data which does not share the same set of labels.

Chapter 4

Estimating Depth as Classification Using Deep Fully Convolutional Residual Networks

4.1 Introduction

Depth estimation is one of the most fundamental tasks in computer vision. Many other computer vision tasks such as object detection, semantic segmentation, scene understanding, can benefit considerably from accurate estimation of depth information. Most existing methods Eigen and Fergus [2015]; Li et al. [2015a]; Wang et al. [2015]; Liu et al. [2015b] formulate depth estimation as a structured regression task due to the fact of depth values being continuous. These regression models for depth estimation are trained by iteratively minimizing the L_2 norm between the predicted depths and the ground-truth depths, and aim to output depths as close to the actual depths as possible during evaluation. However, it is difficult to regress the depth value of input data to be exactly the ground-truth value. For human beings, we may find it difficult to tell the exact distance of a specific point in a natural scene, but we can easily give a rough distance range of that point. Motivated by this, we formulate depth estimation as a pixel-wise classification task by discretizing the continuous depth values into several discrete bins. Instead of training a model to predict the depth value of a point, we train a model to predict the depth range. We show that this simple re-formulation scheme performs surprisingly well.

Another important reason for us to choose classification over regression for depth estimation is that it naturally predicts a confidence in the form of probability distribution over the output space. Different points have different distributions of possible depth values. The depth estimation of some points are easy while others are not. Typical regression models only output the mean values of possible depth values without the variances, (i.e., the confidence of a prediction is missing). Some efforts have been made to obtain this confidence such as the constrained structured regression Pathak et al. [2016], or the Monte-Carlo dropout Kendall et al. [2015]; Gal and Ghahramani [2016]. Compared to these methods which either require specific constraints or multiple forward passes during evaluation, our proposed approach is simple to implement.

The obtained probability distribution can be an important cue during both train-

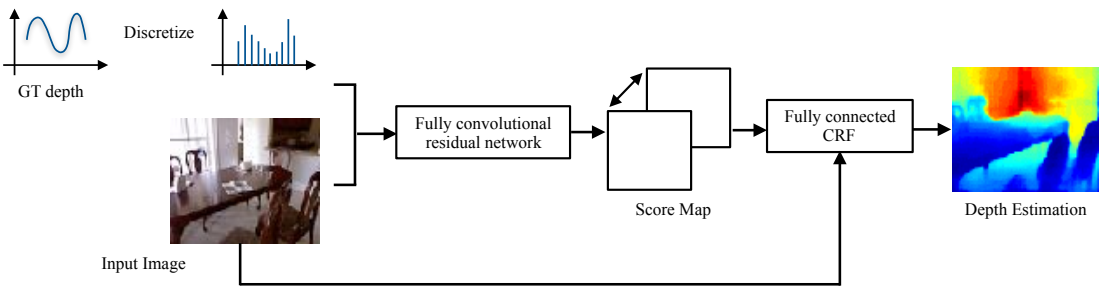

Figure 4.1: An overview of our depth estimation model. It takes as input an image and output dense score maps. Fully-connected CRFs are then applied to obtain the final depth estimation.

ing and post-processing. Although we formulate depth estimation as a classification task by discretization, the depth labels are different from the labels of typical classification tasks such as semantic segmentation. During training, the predicted depth labels that are close to ground-truth and with high confidence can also be used to update model parameters. This is achieved by an information gain loss. As for the post-processing, we apply the fully-connected conditional random fields (CRF) Krähenbühl and Koltun [2011] which have been frequently applied in semantic segmentation Lin et al. [2016]; Chen et al. [2015a]. With the fully connected CRFs, pixel depth estimation with low confidence can be improved by other pixels that are connected to it.

Traditional depth estimation methods enforce geometric assumptions and rely on hand-crafted features such as SIFT, PHOG, GIST, texton, etc. Recently, computer vision has witnessed a series of breakthrough results introduced by deep convolutional neural networks (CNN) Krizhevsky et al. [2012]; Simonyan and Zisserman [2014]. The success of deep networks can be partially attributed to the rich features captured by the stacked layers. Recent evidence has shown that depth estimation benefits largely from increased number of layers Eigen et al. [2014]; Eigen and Fergus [2015]; Liu et al. [2015b]. However, stacking more layers does not necessarily improve performance as the training can become very difficult due to the problem of vanishing gradients. In this work, we apply the the deep residual learning framework proposed by He et al. He et al. [2016a]. It manages to learn the residual mapping of a few stacked layers to avoid the vanishing gradients problem.

An overview of our proposed depth estimation model is illustrated in Fig. 4.1. It takes as input an arbitrarily sized image and outputs a dense score map. Fully connected CRFs are then applied to obtain the final depth estimation. The remaining content of this chapter is organized as follows. Section 4.2 reviews some relevant work. Then we present the proposed method in Section 4.3. Experiment results are presented in Section 4.4. Finally, Section 4.5 concludes the chapter.

4.2 Background

Previous depth estimation methods are mainly based on geometric models. For example, the works of Hedau et al. [2010]; Gupta et al. [2010b]; Schwing and Urtasun [2012] rely on box-shaped models and try to fit the box edges to those observed in an image. These methods are limited to only model particular scene structures and therefore are not applicable for general-scene depth estimations. More recently, non-parametric methods Karsch et al. [2014] are explored. These methods consist of candidate images retrieval, scene alignment and then depth inference using optimizations with smoothness constraints. These methods are based on the assumption that scenes with semantically similar appearances should have similar depth distributions when densely aligned.

Other methods attempt to exploit additional information. To name a few, the authors of Russell and Torralba [2009] estimated depths through user annotations. The work of Liu et al. [2010] performed semantic label prediction before depth estimation. The works of Ladicky et al. [2014]; Wang et al. [2015] have shown that jointly perform depth estimation and semantic labelling can help each other. Given the fact that the extra source of information is not always available, most of recent works formulated depth estimation as a Markov Random Field (MRF) Saxena et al. [2005, 2009, 2007] or Conditional Random Field (CRF) Liu et al. [2014] learning problem. These methods managed to learn the parameters of MRF/CRF in a supervised fashion from a training set of monocular images and their corresponding ground-truth depth images. The depth estimation problem then is formulated as a maximum a posteriori (MAP) inference problem on the CRF model.

With the popularity of deep convolutional neural networks (CNN) since the work of Krizhevsky et al. [2012], some works attempted to solve the depth estimation problem using deep convolutional networks and achieved outstanding performance. Eigen et al. Eigen and Fergus [2015] proposed a multi-scale architecture for predicting depths, surface normals and semantic labels. The multi-scale architecture is able to capture many image details without any superpixels or low-level segmentation. Liu et al. Liu et al. [2015b] presented a deep convolutional neural field model for depth estimation. It learned the unary and pairwise potentials of continuous CRF in a unified deep network. The model is based on fully convolutional networks (FCN) with a novel superpixel pooling method. Similarly, Li et al. Li et al. [2015a] and Wang et al. Wang et al. [2015] also combined the CNNs with CRFs, they formulated depth estimation in a two-layer hierarchical CRF to enforce synergy between global and local predictions.

Anirban et al. Roy and Todorovic [2016] proposed a neural regression forest (NRF) architecture which combines convolutional neural networks with random forests for predicting depths in the continuous domain via regression. The NRF processes a data sample with an ensemble of binary regression trees and the final depth estimation is made by fusing the individual regression results. It allows for parallelizable training of all shallow CNNs, and efficient enforcing of smoothness in depth estimation results. Laina et al. Laina et al. [2016] applied the deep residual networks for depth estimation. In order to improve the output resolution, they presented a novel way to efficiently learn feature map up-sampling within the network.

They also presented a reverse Huber loss which is driven by the value distributions commonly present in depth maps for the network optimization.

Experiment results in the aforementioned works reveal that depth estimation benefits from: (a) an increased number of layers in deep networks; (b) obtaining fine-level details. In this work, we take advantage of the successful deep residual networks He et al. [2016a] and formulate depth estimation as a dense prediction task. We also apply fully connected CRFs Krähenbühl and Koltun [2011] as post-processing. Although Laina et al. Laina et al. [2016] also applied the deep residual network for depth estimation, our method is different from Laina et al. [2016] in 3 distinct ways: Firstly, we formulate depth estimation as a classification task, while Laina et al. [2016] formulated depth estimation as a regression task. Secondly, we can obtain the confidence of depth predictions which can be used during training and post-processing. Lastly, in order to obtain high resolution predictions, Laina et al. [2016] applied an up-sampling scheme while we simply use bilinear interpolation.

The aforementioned CNN based methods formulate depth estimation as a structured regression task due to the continuous property of depth values. However for different pixels in a single monocular image, the possible depth values have different distributions. Depth values of some pixels are easy to predict while others are not. The output of continuous regression lacks this confidence. In Pathak et al. [2016], Pathak et al. presented a novel structured regression framework for image decomposition. It applied special constraints on the output space to capture the confidence of predictions. In Kendall et al. [2015], Kendall et al. proposed a Bayesian neural network for semantic segmentation. It applied the Monte-Carlo dropout during training and obtained the confidence of predictions by multiple forward passes during evaluation. In this work, we obtain the confidence by simply formulating depth estimation as a classification task.

4.3 Proposed Method

In this section, we describe our depth estimation method in detail. We first introduce the network architecture, followed by the introduction of our loss function. Finally, we introduce the fully connected conditional random field (CRF) which is applied as post-processing.

4.3.1 Network architecture

We formulate our depth estimation as a spatially dense prediction task. When applying CNNs to this type of task, the input image is inevitably down-sampled due to the repeated combination of max-pooling and striding. In order to handle this, we follow the fully convolutional network (FCN) which has been proven to be successful in dense pixel labeling. It replaces the fully connected layers in conventional CNN architectures with convolutional layers. By doing this, it makes the fully convolutional networks capable of taking input of arbitrarily sized images and output a down-sampled prediction map. After applying a simple upsample such as bilinear interpolation, the prediction map is of the same size of the input image.

The depth of CNN architectures is of great importance. Much recent works reveal that the VGG Simonyan and Zisserman [2014] network outperforms the shallower AlexNet Krizhevsky et al. [2012]. However, simply stacking more layers to existing CNN architectures does not necessarily improve performance due to the notorious problem of vanishing gradients, which hampers convergence from the beginning during training. The recent ResNet model solves this problem by adding skip connections. We follow the recent success of deep residual network with up to 152 layers He et al. [2016a], which is about 8× deeper than the VGG network but still having fewer parameters to optimize.

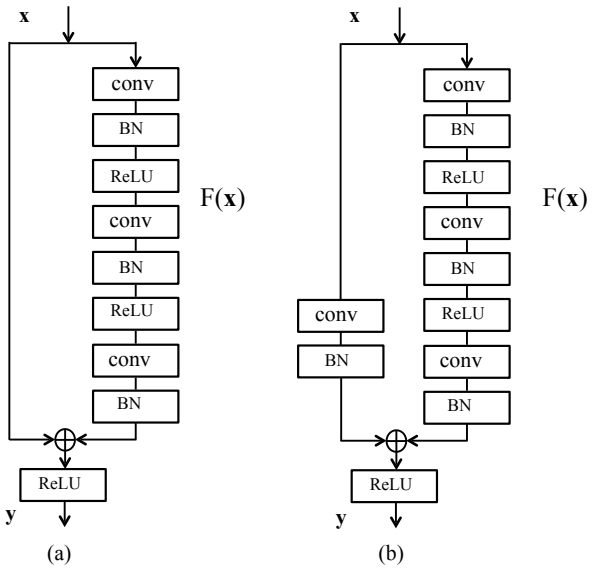

Figure 4.2: Two types of building blocks that can be used in our depth estimation model. (a) building block with identity mapping. (b) building block with linear projection.

Instead of directly learning the underlying mapping of a few stacked layers, the deep residual network learns the residual mapping. Then the original mapping can be realized by feedforward neural networks with "shortcut connections". Shortcut connections are those skipping one or more layers. In our model, we consider two shortcut connections and the building blocks are shown in Fig. 4.2. The building block illustrated in Fig. 4.2(a) is defined as:

$$\mathbf{y} = F(\mathbf{x}, \{W_i\}) + \mathbf{x}, \tag{4.1}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the input and output matrices of stacked layers respectively. The function $F(\mathbf{x}, \{W_i\})$ is the residual mapping that need to be learned. Since the shortcut connection is an element-wise addition, the dimensions of $\mathbf{x}$ and F need to be same.

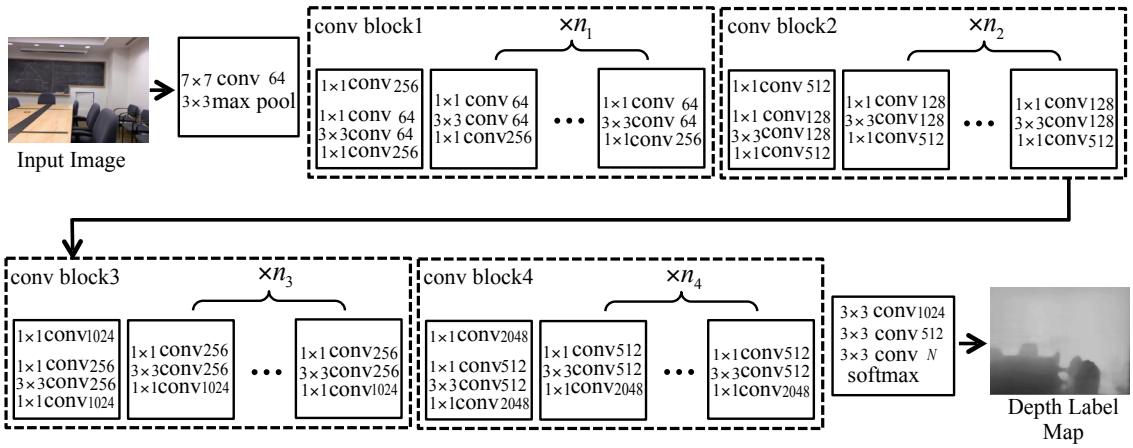

Figure 4.3: Network architecture of our depth estimation model. The input image is fed into a convolutional layer, a max pooling layer and 4 convolution blocks. We consider network architectures with 101 and 152 layers. The value of $[n_1, n_2, n_3, n_4]$ is $[2, 3, 22, 2]$ for the 101-layer network architecture and $[2, 7, 35, 2]$ for the 152-layer network architecture. The last 4 layers are 3 convolutional layers and a softmax layer. The output map is downsampled by a factor of 8 and we preform bilinear interpolation during prediction.

The building block illustrated in Fig. 4.2(b) is defined as:

$$\mathbf{y} = F(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}. \tag{4.2}$$

Compared to the shortcut connection in Eq. (4.1), a linear projection W_s is applied to match the dimensions of $\mathbf{x}$ and F.

The overall network architecture of our depth estimation model is illustrated in Fig. 4.3. The input image is fed into a convolutional layer, a max pooling layer followed by 4 convolution blocks. Each convolution block starts with a building block with linear projection followed by different numbers of building blocks with identity mapping. In this article, we consider two deep residual network architectures with 101 and 152 layers respectively. For the network architecture with 101 layers, the number of building blocks with identity mapping in the four convolution blocks (i.e., n_1, n_2, n_3, n_4 in Fig. 4.3) are 2, 3, 22 and 2 respectively. As for the network architecture with 152 layers, the numbers are 2, 7, 35 and 2. The last four layers are three convolutional layers with channels 1024,512 and N, and a softmax layer, where N is the number of ground-truth labels. Batch normalization and ReLU layers are performed between these convolutional layers. Downsampling is performed by pooling or convolutional layers that have a stride of 2. These include the first 7×7 convolutional layer, the first 3×3 max pooling layer, and the first building block of convolution block 2 in Fig. 4.3. As a result, the output prediction map is downsampled by a factor of 8. During prediction, we perform a bilinear interpolation on this map to make it the same size with the input image.

4.3.2 Loss function

In this work, we use the pixel-wise multinomial logistic loss function as we formulate depth estimation as a classification task. We uniformly discretize the continuous depth values into multiple bins in the log space. Each bin covers a range of depth values and we label the bins according to the range (i.e., the label index of a pixel indicates its distance). The depth labels however are different from the labels of typical classification tasks. For typical classification tasks such as semantic segmentation and object detection, the predictions that are different from ground-truth labels are considered wrong and contribute nothing in updating network parameters. As for depth estimation, the predictions that are close to ground-truth depth labels can also help in updating network parameters. This is achieved by an "information gain" matrix in our loss function.

Specifically, our loss function is defined as:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{D=1}^{B} H(D_i^*, D)\log(P(D|z_i)) \tag{4.3}$$

where $D_i^* \in [1, \ldots, B]$ is the ground-truth depth label of pixel i and B is the total number of discretization bins. $P(D|z_i) = e^{z_{i,D}}/\sum_{d=1}^{B} e^{z_{i,d}}$ is the probability of pixel i labelled with D. $z_{i,d}$ is the output of the last convolutional layer in the network. The "information gain" matrix H is a $B \times B$ symmetric matrix with elements $H(p,q) = \exp[-\alpha(p-q)^2]$ and α is a constant. It encourages the predicted depth labels that are closer to ground-truths have higher contributions in updating network parameters.

During prediction, we set the depth value of each pixel to be the center of its corresponding bin. By formulating depth estimation as classification, we can get the confidence of each prediction in the form of probability distribution. This confidence can also be applied during post-processing via fully connected CRFs.

4.3.3 Fully connected conditional random fields

A deep convolutional network typically does not explicitly take the dependency among local variables into consideration. It does so only implicitly through the field of view. That is why the size of field of view is important in terms of the performance of a CNN. In order to greatly refine the network output, we apply the fully connected CRF proposed in Krähenbühl and Koltun [2011] as post-processing. It connects all pairs of individual pixels in the image. Specifically, the energy function of a fully connected CRF is the sum of unary potential U and pairwise potential V:

$$E(\mathbf{D}) = \sum_i U(D_i) + \sum_{i,j} V(D_i, D_j), \tag{4.4}$$

where $\mathbf{D}$ is the predicted depth labels of pixels and i, j are pixel indices. We use the logistic loss of pixel defined in Eq. (4.3) as the unary potential, which is

$$U(D_i) = L(D_i) = -\log(P(D_i|z_i)).$$

The pairwise potential is defined as

$$\sum_{i,j} V(D_i, D_j) = \Delta(D_i, D_j) \sum_{s=1}^{M} w_s \cdot k^s(\mathbf{f}_i, \mathbf{f}_j),$$

where $\Delta(D_i, D_j)$ is a penalty term on the labelling. Since the label here indicates depth, we enforce a relatively larger penalty for labellings that are far away from ground-truth. For simplicity, we use the absolute difference between two label values to be the penalty: $\Delta(D_i, D_j) = |D_i - D_j|$. There is one pairwise term for each pair of pixels in the image no matter how far they are from each other (i.e., the model's factor graph is fully connected).

Each k^s is the Gaussian kernel depends on features (denoted as $\mathbf{f}$) extracted for pixel i and j and is weighted by parameter w_s. Following Krähenbühl and Koltun [2011], we adopt bilateral positions and color terms, specifically, the kernels are:

$$w_1 \exp(-\frac{\|p_i - p_j\|^2}{2\sigma_\alpha^2} - \frac{\|I_i - I_j\|^2}{2\sigma_\beta^2}) + w_2 \exp(-\frac{\|p_i - p_j\|^2}{2\sigma_\gamma^2}). \tag{4.5}$$

The first kernel is appearance kernel, which depends on both pixel positions (denoted as p) and pixel color intensities (denoted as I). It is inspired by the observation that nearby pixels with similar color are likely to be in the same depth range. The degrees of nearness and similarity are controlled by hyper parameters σ_α and σ_β. The second kernel is smoothness kernel which removes small isolated regions, the scale of smoothness is controlled by σ_γ.

4.4 Experiments

We evaluate our proposed depth estimation approach on 2 benchmark RGB-D datasets: the indoor NYUD2 Silberman et al. [2012] dataset and the outdoor KITTI Geiger et al. [2013] dataset. We organize our experiments into the following three parts:

(1) We show the effectiveness of our depth discretization scheme and compare our discrete depth label classification with continuous depth value regression.

(2) We evaluate the contribution of different components in our proposed approach.

(3) We compare our proposed approach with state-of-the-art methods to show that our approach performs better in both indoor and outdoor scenes. Several measures commonly used in prior works are applied for quantitative evaluations:

- root mean squared error (rms): $\sqrt{\frac{1}{T}\sum_p (d_{gt} - d_p)^2}$
- average relative error (rel): $\frac{1}{T}\sum_p \frac{|d_{gt} - d_p|}{d_{gt}}$
- average $\log_{10}$ error (log10): $\frac{1}{T}\sum_p |\log_{10} d_{gt} - \log_{10} d_p|$
- root mean squared log error (rmslog) $\sqrt{\frac{1}{T}\sum_p (\log d_{gt} - \log d_p)^2}$
- accuracy with threshold *thr*:

percentage (%) of d_p s.t. $\max(\frac{d_{gt}}{d_p}, \frac{d_p}{d_{gt}}) = \delta < thr$ where d_{gt} and d_p are the ground-truth and predicted depths respectively of pixels, and T is the total number of pixels

in all the evaluated images.

4.4.1 Depth label classification vs. depth value regression

Discretizing continuous data would inevitably discard some information. In this part, we first show that the discretization of continuous depth values degrades the depth estimation model by negligible amount. Specifically, we equally discretize the ground-truth depth values of test images in the NYUD2 dataset into different numbers of bins in the linear and log space respectively and calculate three errors as is mentioned above. The results are illustrated in Fig. 4.4.

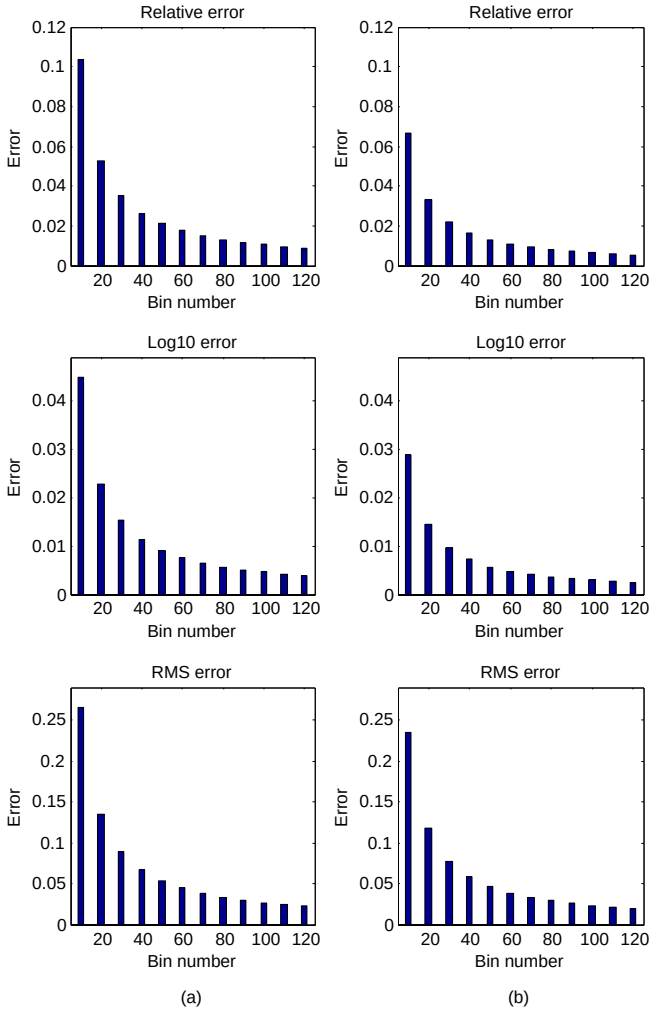

Figure 4.4: Quantitative evaluations of discretized ground-truth depth values of the NYUD2 dataset. (a): errors of ground-truth depth values discretized in linear space. (b): errors of ground-truth depth values discretized in the log space.

We can see from Fig. 4.4 that with the increment of discretization bins, the errors

of discretized ground-truth depths decrease and stop at a negligible amount. And the discretization in the log space leads to lower error than the discretization in the linear space.

As for the accuracies, all the discretized ground-truth depths can reach 100% except for the accuracy with threshold 1.25 when linearly discretizing the ground-truth depths into 10 bins. From this experiment we can see that converting the ground-truth depths from continuous values to discrete labels has negligible effect on the performance. We can reformulate depth estimation from a conventional regression task to a classification task.

We next compare our proposed depth estimation by classification with the conventional depth regression and show the results in Table 4.1. In this experiment, we apply the deep residual network with 101 layers and the parameters are initialized with the ResNet101 model in He et al. [2016a] which is trained on the ImageNet classification dataset. We train our models on standard NYUD2 training set with 795 images and standard KITTI training set with 700 images Eigen et al. [2014] for fast comparison. As for the test sets, we select 650 and 700 images from the raw NYUD2 and KITTI test sets respectively as validation sets. For depth regression, the loss function is standard $L2$ norm which minimizes the squared euclidean norm between predicted and ground-truth depths. The output depth map is upsampled to the same size of the input image through bilinear interpolation. As for our depth estimation by classification, we discretize the continuous depth values into different numbers of bins in the log space. We do not apply CRF post-precessing for both regression and classification. As we can see from Table 4.1 that depth estimation by classification outperforms the conventional depth regression, and the performance of depth classification is not very sensitive to the number of discretization bins.

One important reason for depth estimation by classification outperforms the depth regression is that the regression tends to converge to the mean depth values. This may cause larger errors in areas that are either too far from or too close to the camera. The classification with the information gain may alleviate this problem. In order to testify this, we break down the NYUD2 ground-truth depths into 3 ranges and report the results in Table 4.2. The general setting is the same with the aforementioned experiment. The ground-truth depths are discretized into 100 bins in the log space and the α defined in Eq. (4.3) is set to 0.2.

4.4.2 Component evaluation

In this section, we analyze the contribution of key components including the information gain matrix, fully connected CRFs and network architectures in our proposed approach. We evaluate depth estimation on both the NYUD2 and KITTI datasets. We use the standard training set containing 795 images of the NYUD2 dataset and evaluate on the standard 654 test images. The continuous depth values are discretized into 100 bins in the log space. As for the KITTI dataset, we apply the same split in Eigen et al. [2014] which contains 700 training images and 697 test images. We only use left images and discretize the continuous depth values into 50 bins in the log space. We cap the maximum depth to be 80 meters. During training, we ignore the missing values in ground-truth depths and only evaluate on valid points.

Table 4.1: Depth estimation results by continuous depth value regression and discrete depth label classification for the NYUD2 and KITTI datasets. The first row is the result by regression. The following rows are results of depth label classification with different number of discretization bins.

	Accuracy			Error		
	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	rel	log10	rms
NYUD2						
Regression	65.3%	91.5%	97.4%	0.231	0.095	0.778
10 bins	69.4%	**92.4%**	97.5%	0.213	0.091	0.754
30 bins	70.5%	92.1%	**97.8%**	0.210	**0.090**	0.751
50 bins	68.9%	91.9%	97.0%	0.209	0.092	0.750
80 bins	**70.6%**	92.0%	97.6%	0.211	0.091	**0.747**
100 bins	70.1%	92.1%	97.6%	**0.209**	0.091	0.749
KITTI						
Regression	67.5%	88.6%	90.4%	0.279	0.104	7.916
50 bins	76.3%	**92.1%**	96.3%	0.183	0.077	**6.209**
80 bins	**77.1%**	91.7%	96.6%	**0.180**	**0.072**	6.311
120 bins	76.8%	91.9%	**96.7%**	0.187	0.076	6.263

4.4.2.1 Benefit of information gain matrix

In this part, we evaluate the contribution of the information gain matrix in our loss function. We train the ResNet101 model on both the NYUD2 and KITTI datasets with and without information gain matrices. The α defined in Eq. (4.3) is set to 0.2 and 0.5 for NYUD2 and KITTI respectively. In our experiments, we find that the performance is not sensitive to α. The results are illustrated in Table 4.3. As we can see from this table that the information gain matrix improves the performance of both indoor and outdoor depth estimation.

4.4.2.2 Benefit of fully connected CRFs

In order to evaluate the effect of the fully connected CRFs, we first train the ResNet101 model on both the NYUD2 and KITTI datasets, and then apply the fully connected CRFs as post-processing. We illustrate the results in Table 4.4. As we can see from the table, the fully-connected CRF can improve the depth estimation of both indoor and outdoor scenes.

4.4.2.3 Network Comparisons

In this part, we compare the performance of deep residual networks with the baseline VGG16 net Simonyan and Zisserman [2014] on the NYUD2 dataset. Since we formulate depth estimation as a classification task, we can apply network structures

Table 4.2: Test results on the NYUD2 dataset with different ground-truth ranges. We break down the ground-truth depths into 0m-3m, 3m-7m and 7m-10m.

	Accuracy			Error		
	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	rel	log10	rms
Regression						
0m-3m	65.7%	90.9%	97.4%	0.233	0.087	0.561
3m-7m	70.3%	95.5%	99.5%	0.175	0.075	0.936
7m-10m	45.0%	75.4%	93.5%	0.242	0.129	2.346
Classification						
0m-3m	69.6%	91.2%	97.2%	0.216	0.083	0.561
3m-7m	76.0%	94.9%	98.6%	0.151	0.070	0.857
7m-10m	49.7%	74.9%	93.1%	0.238	0.126	2.199

Table 4.3: Test results on the NYUD2 and KITTI datasets with and without information gain matrices. For each dataset, the first row is the result without information gain matrix, the second row is the result with information gain matrix.

	Accuracy			Error		
	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	rel	log10	rms
NYUD2						
Plain	70.9%	92.1%	98.0%	0.193	0.079	0.716
Infogain	**72.2%**	**92.6%**	**98.0%**	**0.192**	**0.077**	**0.688**
KITTI						
Plain	79.9%	93.7%	97.6%	0.166	0.067	5.443
Infogain	**81.4%**	**93.9%**	**97.6%**	**0.153**	**0.062**	**5.290**

that perform well on semantic segmentation task. Specifically, for the VGG16 net, we apply the structure in Lin et al. [2016]. We keep the layers up to "fc6" in VGG16 net and add 2 convolutional layers with 512 channels, and 2 fully-connected layers with 512 and 100 channels respectively. The results are illustrated in Table 4.5. The performance of residual networks unsurprisingly outperform the VGG16 net, reinforcing the importance of network depth. Note that the performance by the ResNet152 improves little to the ResNet101, this is caused by the overfitting as the training set contains only 795 images. We also compare the number of parameters in the Table 4.5.

4.4.3 State-of-the-art comparisons

In this section, we evaluate our approach on the NYUD2 and KITTI datasets and compare with recent depth estimation methods. We apply the deep residual network

Table 4.4: Test results on the NYUD2 and KITTI datasets with and without the fully connected CRFs as post-processing. For each dataset, the first row is the result without CRFs, the following row is the result with CRFs.

	Accuracy			Error		
	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	rel	log10	rms
NYUD2						
Plain	70.9%	**92.1%**	98.0%	0.193	0.079	0.716
CRF	**71.3%**	92.0%	**98.0%**	**0.190**	**0.079**	**0.696**
KITTI						
Plain	79.9%	93.7%	97.6%	**0.166**	0.067	5.443
CRF	**81.0%**	**94.1%**	**97.9%**	0.167	**0.066**	**5.349**

Table 4.5: Test results on the NYUD2 dataset with different network structures. The first row is the result of the VGG16 net, the following two rows are the results of deep residual networks. We also show the total numbers of parameters of the three networks in the last row.

	Accuracy			Error		
	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	rel	log10	rms
VGG16	62.1%	87.2%	96.0%	0.236	0.097	0.857
ResNet101	70.9%	92.1%	98.0%	0.193	0.079	0.716
ResNet152	**71.2%**	**92.3%**	**98.0%**	**0.187**	**0.071**	**0.681**

	VGG16	ResNet101	ResNet152
Parameters	13.9×10^7	6.7×10^7	8.2×10^7

with 152 layers and the parameters are initialized with the ResNet152 model in He et al. [2016a].

4.4.3.1 NYUD2

We train our model using the entire raw training data specified in the official train/test distribution and test on the standard 654 test images. We discretize the depth values into 100 bins in the log space. We set the parameter α of the information gain matrix to be 0.2. The fully connected CRFs are applied as post-processing. The results are reported in Table 4.6. The first row is the result in Wang et al. [2015] which jointly performs depth estimation and semantic segmentation. The second row is the result of deep convolutional neural fields (DCNF) with fully convolutional network and super-pixel pooling in Liu et al. [2015b]. The third row is the result of nerual regression forest (NRF) in Roy and Todorovic [2016]. The fourth row is the result in Eigen and Fergus [2015] which performs depth estimation in a multi-scale network architecture. The fifth row is the result in Laina et al. [2016] which applies an up-

Table 4.6: Comparison with state-of-the-art on the NYUD2 dataset. The first 4 rows are results by recent depth estimation models. The last row is the result of our approach.

	Accuracy			Error		
	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	rel	log10	rms
Wang et al. Wang et al. [2015]	60.5%	89.0%	97.0%	0.210	0.094	0.745
Liu et al. Liu et al. [2015b]	65.0%	90.6%	97.6%	0.213	0.087	0.759
Anirban et al. Roy and Todorovic [2016]	-	-	-	0.187	0.078	0.744
Eigen et al. Eigen and Fergus [2015]	76.9%	95.0%	98.8%	0.158	-	0.641
Laina et al. Laina et al. [2016]	81.1%	95.3%	98.8%	**0.127**	**0.055**	0.573
Ours	**81.9%**	**96.5%**	**99.2%**	0.141	0.060	**0.540**

sampling scheme. The last row is depth estimation result by our model. As we can see from the table, our deep fully convolutional residual network with depth label classification achieves state-of-the-art performance of 4 evaluation metrics. We also show some qualitative results in Fig. 4.5, from which we can see our method yields better visualizations in general.

4.4.3.2 KITTI

We train our model on the same training set in Godard et al. [2017] which contains 33131 images and test on the same 697 images in Eigen et al. [2014]. But different from the depth estimation method proposed in Godard et al. [2017] which applies both the left and right images in stereo pairs, we only use the left images. The missing values in the ground-truth depth maps are ignored during both training and evaluation. The depth values are discretized into 50 bins in the log space. We set the parameter α of the information gain matrix to be 0.5 and apply fully connected CRFs as post-processing. In order to compare with the recent state-of-the-art results, we cap the maximum depth into both 80 meters and 50 meters and present the results in Table 4.7. We can see from Table 4.7 that our method outperforms the rest methods significantly. Some qualitative results are illustrated in Fig. 4.6. Our approach yields visually better results.

4.4.3.3 Cross-dataset evaluation

In order to show the generalization of our proposed method, we train our model on the raw NYUD2 dataset and test on the SUN RGB-D dataset Song et al. [2015]. The SUN RGB-D is an indoor dataset contains 10335 RGB-D images captured by four different sensors. We only select 500 images randomly from the test set for cross-dataset evaluation. The SUN RGB-D contains 1449 images from the NYUD2 dataset. Our selected testset exlucdes all the images from the NYUD2. We compare our method with Liu et al. Liu et al. [2015b] and Laina et al. Laina et al. [2016]. We use the trained models and evaluation codes released by these authors. The results

Table 4.7: Comparison with state-of-the-art results on the KITTI dataset. We cap the maximum depth to 50 and 80 meters to compare with recent works. For the work in Godard et al. [2017], we also report their results with additional training images in the CityScapes dataset Cordts et al. [2016] and denote as Godard et al. CS.

	Accuracy			Error		
	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	rel	rmslog	rms
			Cap 80 meters			
Liu et al. Liu et al. [2015b]	65.6%	88.1%	95.8%	0.217	-	7.046
Eigen et al. Eigen et al. [2014]	69.2%	89.9%	96.7%	0.190	0.270	7.156
Godard et al. Godard et al. [2017]	81.8%	92.9%	96.6%	0.141	0.242	5.849
Godard et al. CS Godard et al. [2017]	83.6%	93.5%	96.8%	0.136	0.236	5.763
Ours	**88.7%**	**96.3%**	**98.2%**	**0.115**	**0.198**	**4.712**
			Cap 50 meters			
Garg et al. Garg and Reid [2016]	74.0%	90.4%	96.2%	0.169	0.273	5.104
Godard et al. Godard et al. [2017]	84.3%	94.2%	97.2%	0.123	0.221	5.061
Godard et al. CS Godard et al. [2017]	85.8%	94.7%	97.4%	0.118	0.215	4.941
Ours	**89.8%**	**96.6%**	**98.4%**	**0.107**	**0.187**	**3.605**

are illustrated in Table 4.8. We can see that our method can reach satisfactory results on different dataset, and outperforms other methods.

Table 4.8: Test results on the SUN RGB-D dataset for cross-dataset evaluation. The first 2 rows are results by recent depth estimation models. The last row is the result of our approach.

	Accuracy			Error		
	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	rel	log10	rms
Liu Liu et al. [2015b]	35.6%	57.6%	83.1%	0.316	0.161	0.931
Laina Laina et al. [2016]	53.9%	70.3%	**89.0%**	0.279	0.138	0.851
Ours	**56.3%**	**72.7%**	88.2%	**0.256**	**0.127**	**0.839**

4.5 Conclusion

We have presented a deep fully convolutional residual network architecture for depth estimation from single monocular images. We have made use of the recent deep residual networks, discretized continuous depth values into different bins and formulated depth estimation as a discrete classification problem. By this formulation we can easily obtain the confidence of a prediction which can be applied during training via information gain matrices as well as post-processing via fully-connected CRFs.

We have shown that our discretization approach surprisingly performs well.

Note that the proposed network can be further improved by applying the techniques that have been previously explored. For example, it is expected that

- Multi-scale inputs as in Eigen and Fergus [2015] would improve our result.
- Concatenating the mid-layers' outputs may better use the low-, mid-layers information as in Hariharan et al. [2015].
- Upsampling the prediction maps as in Long et al. [2015] would be beneficial too.

We leave these directions in our future work.

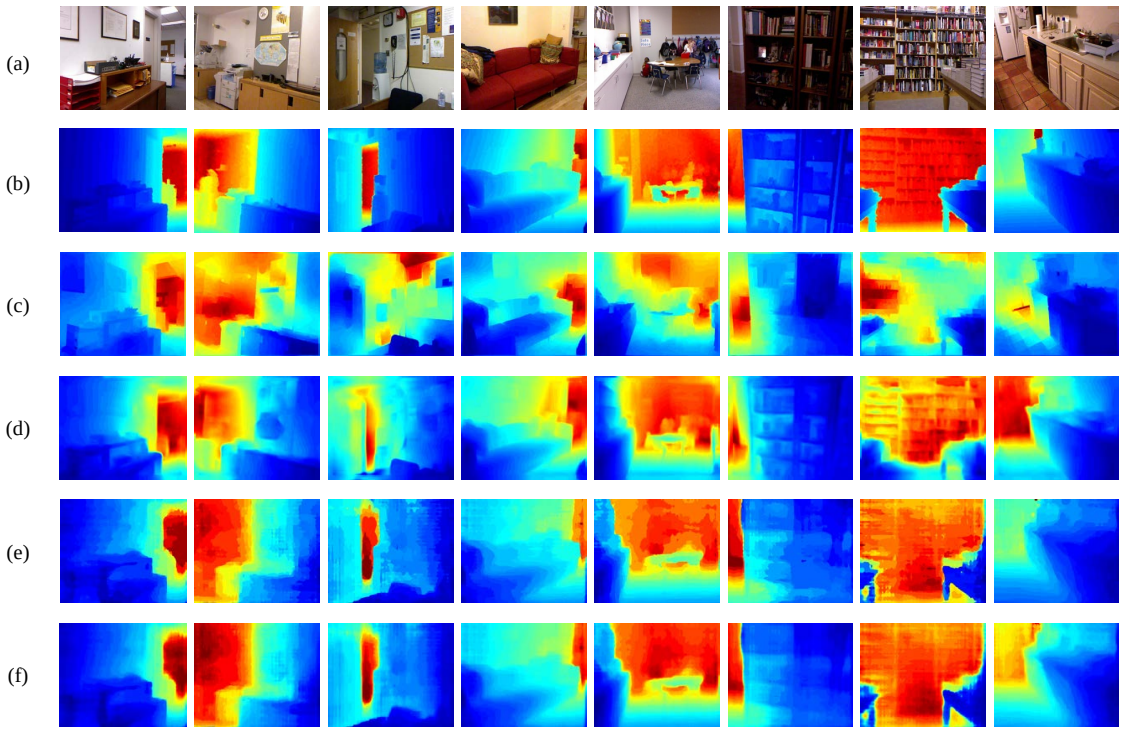

Figure 4.5: Some depth estimation results on the NYUD2 dataset. (a) RGB Input; (b) Ground-truth depth; (c) Results of Liu et al. Liu et al. [2015b]; (d) Results of Eigen et al. Eigen and Fergus [2015]; (e) Results of our model without fully-connected CRFs; (f) Results of our model with fully-connected CRFs.

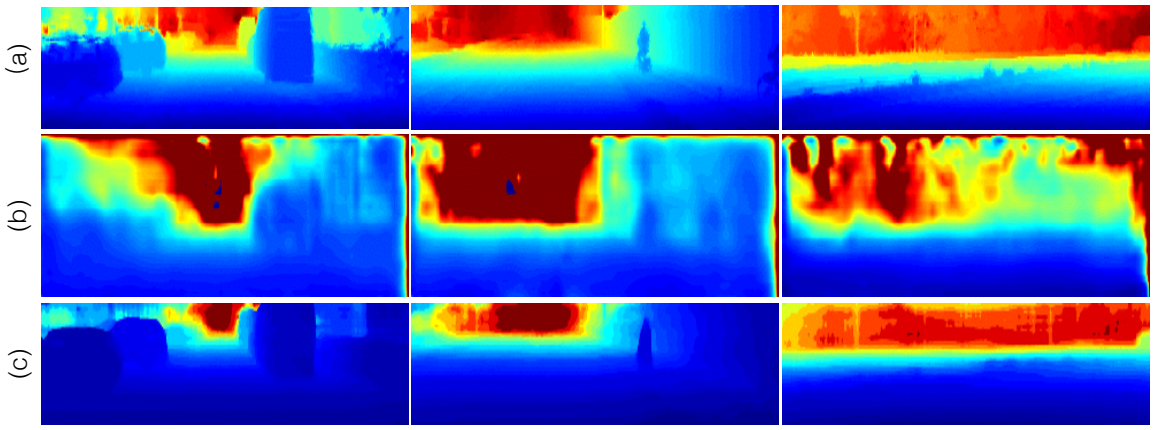

Figure 4.6: Some depth estimation results of the KITTI dataset. The first row are the ground-truth depths, the second row are the results by Garg and Reid [2016], the last row are the results by our approach.

Chapter 5

Monocular Depth Estimation with Augmented Ordinal Depth Relationships

5.1 Introduction

Predicting accurate depths from single monocular images is a fundamental task in computer vision and has been an active research topic for decades. Typical methods formulate depth estimation as a supervised learning task Saxena et al. [2005]; Liu et al. [2015a]; Eigen et al. [2014]. As a result, large amounts of metric ground-truth depths are needed. However, the acquisition of metric ground-truth depths requires depth sensors, and the collected RGB-D training data is limited in the size as well as the diversity of scenes due to the limitation of depth sensors. For example, the popular Microsoft Kinect can not obtain the depths of far objects in outdoor scenes.

In order to overcome the problem of limited metric ground-truth depths, some recent works manage to predict depths from stereo videos Garg and Reid [2016]; Godard et al. [2017]; Xie et al. [2016] without the supervision of ground-truth depths. Specifically, the model is trained by computing the disparity maps and minimizing an image reconstruction loss between training stereo pairs. The performance is not satisfactory due to the absence of ground-truth depths during training. However, the training stereo videos are easier to obtain than metric ground-truth depths and are plenty in terms of amount as well as scene diversity.

Driven by the aforementioned characteristics of recent depth estimation methods, a question arises: Is it possible to acquire large quantities of training data from stereo videos to improve the performance of monocular depth estimation?

Compared to metric depths, relative depths can be easily obtained from stereo videos using existing stereo matching algorithms Zbontar and LeCun [2016]; Spyropoulos et al. [2014]; Luo et al. [2016]; Zhang et al. [2009]. The recent works by Zoran et al. Zoran et al. [2015] and Chen et al. Chen et al. [2016] have revealed that it is possible to predict satisfactory metric depths with only relative ground-truth depths. In this chapter, we propose to improve the performance of metric depth estimation with relative depths generated from stereo movie videos. An overview of our approach is illustrated in Fig. 5.1. Our approach can be broadly divided into 3

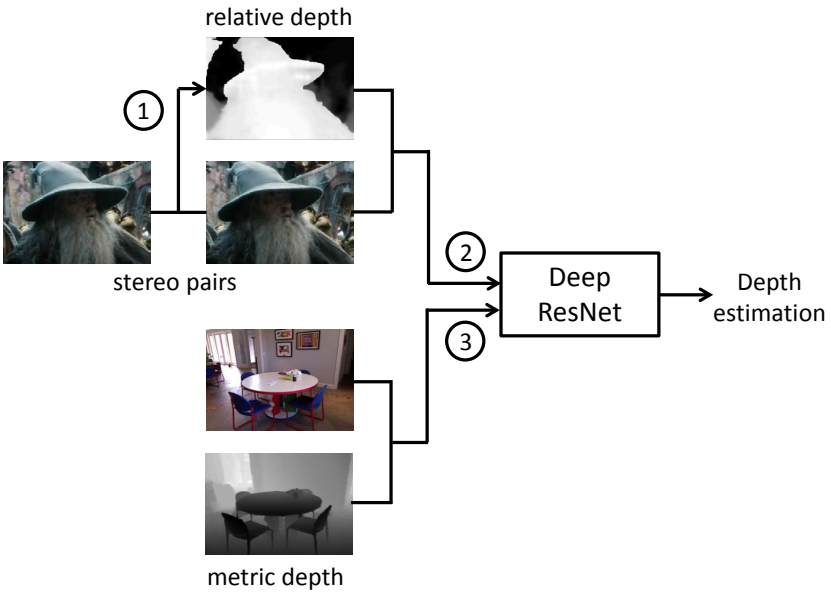

Figure 5.1: Overview of our proposed depth estimation method. We first generate relative depths from stereo pairs, then pretrain a deep residual network with the relative depths. Finally, we finetune the network with metric depths for monocular depth estimation.

steps: We first obtain ground-truth relative depths from stereo movie videos, then we pretrain a deep residual network with our relative ground-truth depths. Finally, we finetune our network on benchmark RGB-D datasets with metric ground-truth depths. Note that, as we exploit 3D movie stereo videos, which do not have the camera parameters and typically are re-calibrated for display, it is impossible to compute the metric depth.

Most existing methods formulate depth estimation as a regression problem due to the continuous property of depths Liu et al. [2015a]; Eigen et al. [2014]; Laina et al. [2016]. For human beings, we may find it difficult to tell the exact distance of a specific point in a natural scene, but we can easily give a rough distance range of that point. Motivated by this, we formulate depth estimation as a pixel-wise classification task by discretizing the continuous depth values into several discrete bins and show that this simple re-formulation scheme performs surprisingly well. More importantly, we can easily obtain the confidence of a depth prediction in the form of probability distribution. With this confidence, we can apply an information gain loss to make use of the predictions that are close to ground-truth during training.

To summarize, we highlight the contributions of our work as follows:

1. We formulate depth estimation as a classification task and propose an information gain loss.

2. We propose a new dataset Relative Depth in Stereo (RDIS) containing images labelled with dense relative depths. The relative depths are generated with

very low cost.

3. We show that our proposed RDIS dataset can improve the performance of metric depth estimation significantly and our proposed method outperforms state-of-the-art depth estimation methods on both indoor and outdoor benchmark RGB-D datasets.

5.2 Background

Traditional depth estimation methods are mainly based on geometric models. For example, the works of Hedau et al. [2010]; Gupta et al. [2010b]; Schwing and Urtasun [2012] rely on box-shaped models and try to fit the box edges to those observed in the image. These methods are limited to only model particular scene structures and therefore are not applicable for general-scene depth estimations. More recently, non-parametric methods Karsch et al. [2014] are explored. These methods consist of candidate images retrieval, scene alignment and then depth inference using optimizations with smoothness constraints. These methods are based on the assumption that scenes with semantically similar appearances should have similar depth distributions when densely aligned.

Most depth estimation algorithms in recent years achieve outstanding performance by training deep convolutional neural networks (CNN) Krizhevsky et al. [2012]; Simonyan and Zisserman [2014]; Long et al. [2015] with fully annotated RGB-D datasets Silberman et al. [2012]; Saxena et al. [2009]; Geiger et al. [2013]. Liu et al. Liu et al. [2015b] presented a deep convolutional neural field which jointly learns the unary and pairwise potentials of continuous conditional random fields (CRF) in a unified deep network. Eigen et al. Eigen and Fergus [2015] proposed a multi-scale network architecture to predict depths as well as surface normals and semantic labels. Li et al. Li et al. [2015a] and Wang et al. Wang et al. [2015] formulated depth estimation in a two-layer hierarchical CRF to enforce synergy between global and local predictions. Laina et al. Laina et al. [2016] applied the latest deep residual network He et al. [2016a] as well as an up-sampling scheme for depth estimation.

Other recent works managed to train deep CNNs for depth estimation in an unsupervised manner. To name a few, Garg et al. Garg and Reid [2016] and Clément et al. Godard et al. [2017] treated depth estimation as an image reconstruction Flynn et al. [2016] problem during training and output disparity maps during prediction. In order to construct a fully differentiable training loss, Taylor approximation and bilinear interpolation are applied in Garg and Reid [2016] and Godard et al. [2017] respectively. Since the network outputs of Garg and Reid [2016] and Godard et al. [2017] are disparity maps, camera parameters are needed to recover the metric depths. Similarly, the Deep3D model Xie et al. [2016] also applied an image reconstruction loss during training, where their goal is to predict the right view from the left view of a stereo pair, and the disparity map is generated internally.

Ordinal relationships and rankings have also been exploited for mid-level vision tasks including depth estimation in recent years. Zoran et al. Zoran et al. [2015] learned the ordinal relationships between pairs of points using a classification loss, then they solved a constrained quadratic optimization problem to map the ordinal

estimates to metric values. Chen et al. Chen et al. [2016] proposed to learn ordinal relationships through a ranking loss Cao et al. [2007] and retrieve the metric depth values by simple normalization. Notably, Chen et al. [2016] also proposed a new dataset Depth in the Wild (DIW) consisting of images in the wild labelled with relative depths.

Our work is mainly inspired by Chen et al.'s single-image depth perception in the wild Chen et al. [2016]. However, our approach is different in three distinct aspects. First, instead of manually labelling pixels with relative relationships, we acquire relative depths using existing stereo matching algorithm from stereo movie videos and thus can obtain large amount of training data with low cost. Second, instead of labelling only one pair of points per image with relative relationships, we generate dense relative depth maps. Finally, in order to retrieve metric depth predictions, they arbitrarily normalize the predicted relative depth maps such that the mean and standard deviation are the same with the metric ground-truth depths of training set, while we finetune our pretrained network with metric ground-truth depths for better performance.

5.3 Proposed Method

In this section, we elaborate our proposed method for monocular depth estimation. We first present the stereo matching algorithm that we used to generate relative ground-truth depth. Then we introduce our network architecture, followed by the introduction of our loss functions.

5.3.1 Relative depth generation

The first step of our approach is to generate relative ground-truth depth from stereo videos using existing stereo matching algorithm. Stereo matching algorithms rely on computing matching costs to measure the similarities of stereo pairs. In this chapter, we choose the commonly-used absolute difference (AD) matching cost combined with a background subtraction by bilateral filtering (BilSub) which has been proven to perform well by Hirschmuller et al. Hirschmuller and Scharstein [2009]. For a pixel p in the left image, its corresponding pixel in the right image is represented as $p - d$, where d is the disparity. The absolute difference is represented as:

$$C_{ad}(p, d) = |I_L(p) - I_R(p - d)|, \tag{5.1}$$

where I_L and I_R are left and right images respectively. We sum the costs of all three channels of color images. The bilateral filtering effectively removes a local offset without blurring high contrast texture differences that may correspond to depth discontinuities.

As for the stereo algorithm, we use the semi-global matching (SGM) method Hirschmuller [2008]. It aims to minimize a global 2D energy function by solving a large number of

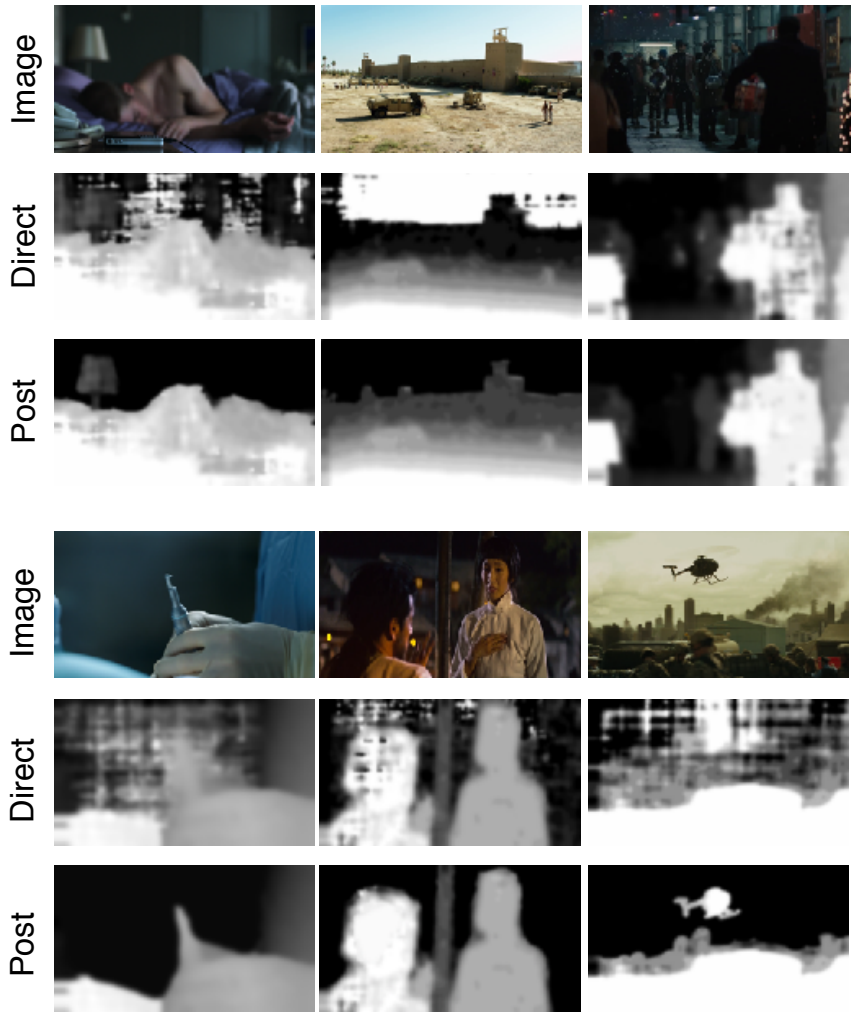

Figure 5.2: Some examples of our Relative Depth in Stereo (RDIS) dataset. The first row are RGB images, the second row are disparity maps directly generated by stereo algorithm. The last row are post-processed disparity maps, which are used as ground-truths.

1D minimization problems. The energy function is:

$$E(D) = \sum_{p}(C(p, D_p) + \sum_{q \in N_p} P_1 \mathrm{T}[|D_p - D_q| = 1] + \sum_{q \in N_p} P_2 \mathrm{T}[|D_p - D_q| > 1]), \tag{5.2}$$

where the first term calculates the sum of a pixel-wise matching cost for all pixels at their disparities D_p. The second term adds a constant penalty $P1$ for all pixels q in the neighborhood N_p of p, for which the disparity changes a little bit (i.e., 1 pixel). Similarly, the third term adds a larger constant penalty $P2$, for all larger disparity changes. The SGM calculates $E(D)$ along 1D paths from 8 directions towards each pixel of interest using dynamic programming. The costs of all paths are summed for each pixel and disparity. The disparity is then determined by winner-takes-all. During training, We label the pairs of points with ordinal relationships (farther, closer, equal) according to their disparities. Since the disparity values of two points can not be exactly the same, we apply a relaxed definition of equality. The ordinal relationship of a pair of points is equal if the disparity difference is smaller than a fixed threshold.

The direct output of stereo matching algorithm is a dense disparity map with the left image treated as the reference image. This disparity map can not be directly used for training due to the defects such as noise, discontinuities or incorrect values. Some examples are shown in Fig. 5.2. As a result, we need to post-process the disparity maps generated by stereo algorithm. The post-processing is done by experienced workers from movie production company using professional movie production software. Specifically, we first correct the vague or missing boundaries of objects using B-splines, then we smooth the disparity values within objects and background. It takes a median of 90 seconds to post-process an image of our dataset. Although the labelling of our dataset takes longer time than the DIW dataset, our dataset is densely labelled and contains more ordinal relationships than the DIW dataset. In terms of single ordinal relationship labelling our method is much more efficient. After post-processing, each disparity map is visually checked by two workers according to the intensities. The workers are required to assign “overall correct", “contain mislabelled parts" or “not sure" to each disparity map. We only keep the disparity maps which both workers assigned as “overall correct".

We also test several other stereo matching algorithms including the deep learning based. Although the qualities of these direct output disparities are different, they are all very coarse, furthermore the difference becomes negligible after human post-processing. So we pick the simplest stereo matching method.

We collect 70 3D movies produced in recent years. Since the stereo matching algorithm requires the stereo videos to be rectified, we only use 3D movies created by post-production instead of movies taken with stereo cameras. In order to avoid similar frames, we only select roughly 1500 frames in each movie. With the selected frames, we generate a new dataset Relative Depth in Stereo (RDIS) containing 97652 training images labelled with dense relative ground-truth depths. Notably, we can not obtain the metric depths from the relative depths because we do not have the camera parameters of these 3D movies. Our dataset has no test images because: 1)

The goal of our dataset is to improve the performance of metric depth estimation. 2) Our ground-truth disparities are obtained through stereo matching algorithm and inevitably contain noisy points.

5.3.2 Network architecture

Recently, a deep residual learning framework has been introduced by He et al. He et al. [2016a,b] and showing compelling accuracy and nice convergence behaviours. In our work, we follow the deep residual network architecture proposed by Wu et al. Wu et al. [2016] which contains fewer layers but outperforms the deep residual network with 152 layers in He et al. [2016a].

Instead of directly learning the underlying mapping of a few stacked layer, the deep residual network learns the residual mapping through building blocks. We consider two types of building blocks in our network architecture. The first is defined as:

$$\mathbf{y} = F(\mathbf{x}, \{W_i\}) + \mathbf{x}, \tag{5.3}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the input and output matrices of stacked layers respectively. The function $F(\mathbf{x}, \{W_i\})$ is the residual mapping that need to be learned. The dimensions of $\mathbf{x}$ and F need to be equal since the addition is element-wise. If this is not the case, we apply another building block defined as:

$$\mathbf{y} = F(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}. \tag{5.4}$$

Comparing to the shortcut connection in Eq. (5.3), a linear projection W_s is applied to match the dimensions of $\mathbf{x}$ and F.

We illustrate our detailed network structure in Fig. 5.3. Generally, it is composed of 6 convolution blocks. Each convolution block starts with a building block with linear projection followed by different numbers of building blocks with identity mapping. Two max pooling layers with stride of 2 are applied before the first and the second convolution blocks. The first convolutional layers of block 3, 4 and 5 have a stride of 2. The dilations of the first convolutional layers of block 4 and 5 are 2 and 4 respectively. As a result, our network takes as inputs of arbitrarily sized images and downsamples by a factor of 8. Batch normalizations (BNs) Ioffe and Szegedy [2015] and ReLUs are applied before weight layers. We initialize the layers up to block 6 with our model pretrained on the ImageNet Russakovsky et al. [2015] and Places365 Zhou et al. [2017] datasets. After block 6, we add 3 convolutional layers with randomly initialized weights. The channels of the first and second added convolutional layers are 1024 and 512 respectively. The channel number of last convolutional layer is determined by the loss function. The channel number is 1 for the pretraining using ranking loss. As for the finetuning, we discretize the continuous metric depths into several bins and formulate depth estimation as a classification task, the channel number is equal to the bin number. We give more details about the loss functions below.

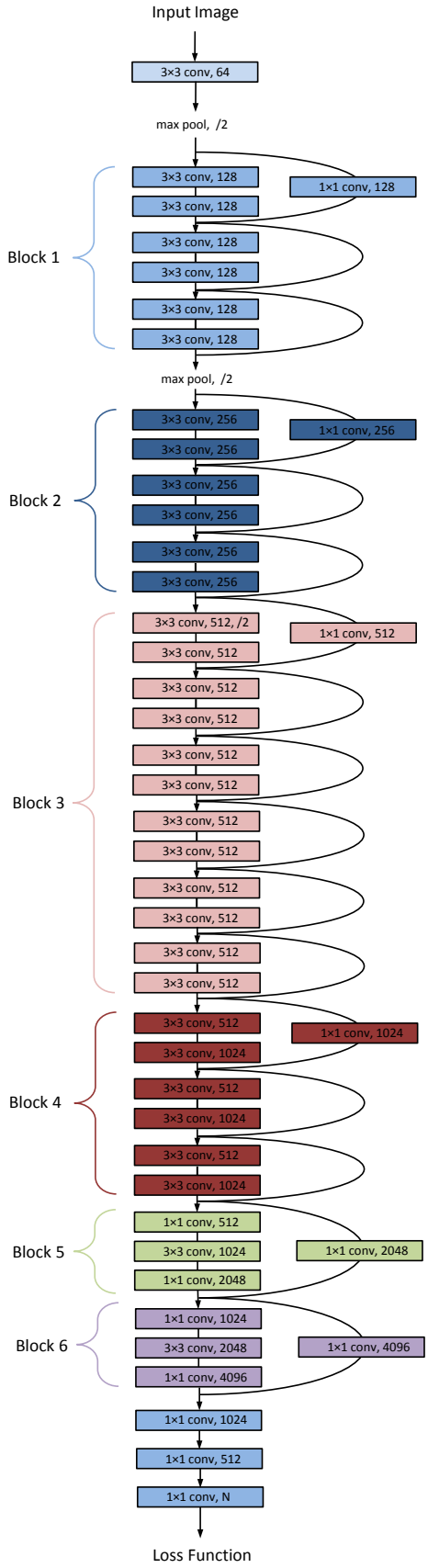

Figure 5.3: Detailed structure of our deep residual network. It has 6 convolution blocks, each with different numbers of residual units.

5.3.3 Loss function

Our proposed approach for depth estimation contains two training stages: pretraining with relative depths and finetuning with metric depths. For the pretraining, we employ the ranking loss which encourages a small difference between depths if the ground-truth ordinal relation is equality and encourages a large difference otherwise. Specifically, consider a training image I with K pairs of points with ground-truth ordinal relations $R = \{(i_k, j_k, r_k)\}, k \in [1, \ldots, K]$, where i_k and j_k are the two points of k-th pair, and r_k is the ground-truth depth relation between i_k and j_k: closer $(+1)$, farther (-1) and equal (0). Let z be the output depth map of our deep residual network and z_{i_k}, z_{j_k} be the predicted depth values of i_k and j_k. The ranking loss is defined as:

$$L_{rank}(I, R, z) = \sum_{k=1}^{K} E(I, i_k, j_k, r, z), \tag{5.5}$$

where $E(I, i_k, j_k, r, z)$ is the loss of the k-th pair:

$$E = \begin{cases} \log(1 + \exp(-z_{i_k} + z_{j_k})), & r_k = +1; \\ \log(1 + \exp(z_{i_k} - z_{j_k})), & r_k = -1; \\ (z_{j_k} - z_{i_k})^2, & r_k = 0. \end{cases} \tag{5.6}$$

After pretraining, we finetune our network with discretized metric depths. We use the pixel-wise multinomial logistic loss defined as:

$$L_{\log} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{D=1}^{B} H(D_i^*, D) \log(P(D|z_i)), \tag{5.7}$$

where $D_i^* \in [1, \ldots, B]$ is the ground-truth depth label of pixel i and B is the total number of discretization bins. N is the number of pixels. $P(D|z_i) = e^{z_{i,D}} / \sum_{d=1}^{B} e^{z_{i,d}}$ is the probability of pixel i labelled with D. $z_{i,d}$ is the output of last convolutional layer in the network.

Although we formulate depth estimation as a classification task by discretizing continuous depth values into several bins, the depth labels are different with the labels of other classification tasks (e.g., semantic segmentation). Predicted depth labels that are closer to ground-truth should have more contribution in updating network weights. This is achieved through the information gain matrix H in Eq. (5.7). It is a $B \times B$ symmetric matrix with elements $H(p, q) = \exp[-\alpha(p - q)^2]$ and α is a constant. During training, we equally discretize the continuous depths in the log space into several bins and during prediction, we set the depth value of each pixel to be the center of its corresponding bin.

5.4 Experiments

We organize our experiments into the following 3 parts: 1) We demonstrate the benefit of the pretraining on our proposed Relative Depth in Stereo (RDIS) dataset by

comparing with other pretraining schemes; 2) We evaluate the metric depth estimation on indoor and outdoor benchmark RGB-D datasets and analyze the contributions of some key components in our approach; 3) We evaluate both metric and relative depth estimation and compare with state-of-the-art results. During pretraining and finetuning, we apply online data augmentation including random scaling and flipping. We apply the following measures for metric depth evaluation:

- root mean squared error (rms): $\sqrt{\frac{1}{T}\sum_p (d_{gt} - d_p)^2}$
- average relative error (rel): $\frac{1}{T}\sum_p \frac{|d_{gt} - d_p|}{d_{gt}}$
- average $\log_{10}$ error (log10): $\frac{1}{T}\sum_p |\log_{10} d_{gt} - \log_{10} d_p|$
- root mean squared log error (rmslog) $\sqrt{\frac{1}{T}\sum_p (\log d_{gt} - \log d_p)^2}$
- accuracy with threshold *thr*:

percentage (%) of d_p s.t. $\max(\frac{d_{gt}}{d_p}, \frac{d_p}{d_{gt}}) = \delta < thr$ where d_{gt} and d_p are the ground-truth and predicted depths respectively of pixels, and T is the total number of pixels in all the evaluated images. As for the relative depth evaluation, we report the Weighted Human Disagreement Rate (WHDR) Zoran et al. [2015], the average disagreement rate with human annotators, weighted by their confidence (here set to 1). We implement our network training based on the MXNet Chen et al. [2015b].

5.4.1 Benefit of pretraining

In this section, we show the benefit of the pretraining with our proposed RDIS dataset. Since our proposed RDIS dataset is densely labelled with relative depths, we need first to determine the number of ground-truth pairs in each image during pretraining. We randomly sample 100, 500, 1K and 5K ground-truth pairs in each input image during pretraining and finetune on both the NYUD2 Silberman et al. [2012] and KITTI Geiger et al. [2013] datasets.

The standard NYUD2 training set contains 795 images. We split the 795 images into a training set with 400 images and a validation set with 395 images. We discretize the continuous metric depth values into 100 bins in the log space. As for the KITTI dataset, we apply the same split in Eigen et al. [2014] which contains 700 training images and 697 test images. We further evenly split the 700 training images into a training set and a validation set. We only use left images and discretize the continuous metric depth values into 50 bins in the log space. We cap the maximum depth to be 80 meters. For both the NYUD2 and KITTI datasets, we finetune on our split training sets and evaluate on our validation sets. During finetuning, we ignore the missing values in ground-truth depth maps and only evaluate on valid points. We do not apply the information gain matrix in this experiment. The results are illustrated in Table 5.1. As we can see from the table that for both indoor and outdoor datasets, the performance increases with the number of pairs and achieves the best when using 1K pairs per input image. Further increasing the number of pairs does not improve the performance. For pretraining with 5K pairs of points, we further add dropouts and evaluate the accuracy with $\delta < 1.25$ on the NYUD2 dataset. The accuracies are 63.7%, 63.1%, 62.6% and 62.3% with 32K, 35K, 40K and 43K iterations. It demonstrates that the performance decrease is caused by overfitting. In the follow-

Table 5.1: Comparison between different numbers of pairs during pretraining. The model is pretrained on our RDIS dataset and finetuned on NYUD2 and KITTI datasets. For each dataset, each row represents different numbers of ground-truth pairs in each input image during pretraining.

	Accuracy			Error		
	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	rel	log10	rms
NYUD2						
100 pairs	63.8%	90.2%	97.5%	0.202	0.090	0.816
500 pairs	68.2%	92.5%	98.6%	0.178	0.081	0.750
1K pairs	**71.1%**	**93.3%**	**98.6%**	**0.173**	**0.077**	**0.721**
5K pairs	63.0%	89.1%	97.1%	0.208	0.092	0.828
KITTI						
100 pairs	70.6%	88.3%	94.6%	0.230	0.088	6.357
500 pairs	74.1%	90.0%	95.3%	0.205	0.079	5.900
1K pairs	**74.2%**	**90.0%**	**95.5%**	**0.205**	**0.079**	**5.828**
5K pairs	70.2%	87.3%	94.1%	0.223	0.088	6.629

ing experiments, we all sample 1K ground-truth pairs in each input image during pretraining.

In order to demonstrate the quality of our proposed RDIS dataset, we conduct experimental comparisons against several pretraining shcemes: 1) Directly finetune our ResNet model on RGB-D datasets without pretraining (Direct); 2) Pretrain our ResNet model on the DIW Chen et al. [2016] dataset and finetune on RGB-D datasets (DIW); 3) Pretrain our ResNet model using our RDIS images and finetune on RGB-D datasets, the ground-truth relative depths for pretraining are generated using the Deep3D model Xie et al. [2016] (Deep3D).

We perform finetuning on the standard training set of the NYUD2 which contains 795 images and evaluate on the standard test set which contains 654 images. The continuous metric depth values are discretized into 100 bins in the log space. The parameter α of the information gain matrix defined in Eq. (5.7) is set to 2.0. As for the KITTI dataset, we finetune on the same 700 training images and evaluate on the same 697 test images as in Eigen et al. [2014]. The continuous metric depth values are discretized into 50 bins in the log space and the maximum depth value is capped to be 80 meters. The parameter α is set to 0.2. We ignore the missing ground-truth values during both finetuning and evaluation.

We show the results in Table 5.2. We can see from the table that the pretraining on our proposed RDIS dataset improves the depth estimation of both indoor and outdoor datasets significantly, and even outperforms the pretraining on the DIW dataset. Notably, compared to the DIW dataset which contains 421K training images with manually labelled relative depths, our RDIS dataset contains only 97652 images, and the relative ground-truth depths are generated by existing stereo matching

Table 5.2: Test results on the NYUD2 and KITTI datasets with different pretraining. For each dataset, the first row is the result without pretraining; the second row is the result with pretaining on the DIW dataset; the third row is the result with pretaining using our RDIS images but the ground-truth relative depths are generated by the Deep3D Xie et al. [2016] model; the last row is the result with pretraining on our RDIS dataset.

	Accuracy			Error		
	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	rel	log10	rms
NYUD2						
Direct	73.3%	93.5%	98.1%	0.186	0.075	0.666
DIW	77.3%	95.4%	98.9%	0.160	0.066	**0.600**
Deep3D	72.5%	92.8%	97.8%	0.191	0.077	0.683
Ours	**78.1%**	**95.4%**	**98.9%**	**0.157**	**0.065**	0.604
KITTI						
Direct	77.3%	92.1%	96.9%	0.173	0.070	5.890
DIW	79.7%	93.7%	97.8%	0.154	0.064	5.251
Deep3D	76.1%	91.9%	97.1%	0.178	0.072	5.765
Ours	**82.9%**	**94.3%**	**98.2%**	**0.142**	**0.058**	**5.066**

algorithm.

5.4.2 Component analysis

In this section, we evaluate metric depth estimation on the indoor NYUD2 and outdoor KITTI datasets and analyze the contributions of some key components of our approach. We use the same dataset settings with the second experiment in Sec. 5.4.1.

5.4.2.1 Network comparisons

In this part, we compare our deep residual network architecture against two baseline networks: deep residual network with 101 and 152 layers in He et al. [2016a]. We pretrain the 3 models on our RDIS dataset and finetune on the NYUD2 dataset. Similar to our network architecture, we replace the last 1000-way classification layers of ResNet101 and ResNet152 with one channel convolutional layers during pretraining and 100-way classification layers during finetuning. We also add two convolutional layers with 1024 and 512 channels respectively before the last layer. We do not apply the information gain matrix in this experiment. The results are illustrated in Table 5.3. From the table we can see that our network architecture outperforms the deeper ResNet101 and ResNet152.

Table 5.3: Test results on the NYUD2 dataset with different network architectures. The first row is the result of the ResNet101, the second row is the result of the ResNet152, the last row is the result of our network.

	Accuracy			Error		
	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	rel	log10	rms
Res101	76.1%	94.7%	98.5%	0.170	0.071	0.632
Res152	76.2%	94.9%	98.5%	0.169	0.070	0.626
Ours	**77.8%**	**95.3%**	**98.8%**	**0.159**	**0.066**	**0.606**

Table 5.4: Test results on the NYUD2 and KITTI datasets with and without information gain matrix. For each dataset, the first row is the result without information gain matrix, the following row is the result with information gain matrix.

	Accuracy			Error		
	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	rel	log10	rms
NYUD2						
Plain	77.8%	95.3%	98.8%	0.159	0.066	0.606
Infogain	**78.1%**	**95.4%**	**98.9%**	**0.157**	**0.065**	**0.604**
KITTI						
Plain	80.5%	93.9%	97.7%	0.158	0.064	5.415
Infogain	**82.9%**	**94.3%**	**98.2%**	**0.142**	**0.058**	**5.066**

5.4.2.2 Benefit of information gain matrix

In this part, we evaluate the contribution of the information gain matrix during finetuning. We pretrain the network on our RDIS dataset and finetune on both the NYUD2 and KITTI datasets with and without the information gain matrix. The parameter α defined in Eq. (5.7) is set to 2.0 and 0.2 respectively for the NYUD2 and KITTI datasets. The results are illustrated in Table 5.4. As we can see from the table that the information gain matrix improves the performance of both indoor and outdoor depth estimation.

5.4.2.3 Depth classification vs. depth regression

In this part, we compare our depth estimation by classification with the conventional regression. We directly train the ResNet101 model without pretraining on our RDIS dataset. For depth regression, we use the $L2$ loss. For our depth estimation as classification, we discretize the continuous depth values into 100 and 50 bins in the log space respectively for the NYUD2 and KITTI datasets. And we set the parameter α to 2.0 and 0.2 respectively. We show the results in Table 5.5, from which we can see that our depth estimation by classification outperforms the conventional regression.

Table 5.5: Test results of depth estimation by classification and regression on the NYUD2 and KITTI datasets. For each dataset, the first row is the result of regression, the following row is the result of classification.

	Accuracy			Error		
	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	rel	log10	rms
NYUD2						
regression	66.9%	92.1%	98.0%	0.215	0.084	0.730
classification	**72.3%**	**92.7%**	**98.3%**	**0.195**	**0.077**	**0.691**
KITTI						
regression	68.9%	89.4%	91.1%	0.256	0.092	7.160
classification	**79.9%**	**93.7%**	**97.6%**	**0.166**	**0.067**	**5.443**

Table 5.6: Comparison with state-of-the-art results on the NYUD2 dataset. The first 5 rows are the results by recent depth estimation methods, the last row is the result by our approach.

	Accuracy			Error		
	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	rel	log10	rms
Wang et al. Wang et al. [2015]	60.5%	89.0%	97.0%	0.210	0.094	0.745
Liu et al. Liu et al. [2015b]	65.0%	90.6%	97.6%	0.213	0.087	0.759
Eigen et al. Eigen and Fergus [2015]	76.9%	95.0%	98.8%	0.158	-	0.641
Laina et al. Laina et al. [2016]	81.1%	95.3%	98.8%	**0.127**	**0.055**	0.573
Ours	**83.1%**	**96.2%**	**98.8%**	0.132	0.057	**0.538**

5.4.3 State-of-the-art comparisons

In this section, we evaluate metric depth estimation on the NYUD2 and KITTI datasets and compare with recent depth estimation methods. During pretraining, we use 1K pairs of points in each input image. During finetuning, we discretize the continuous metric depths into 100 and 50 bins in log space for the NYUD2 and KITTI datasets respectively. We also evaluate relative depth estimation on the Depth in the Wild (DIW) Chen et al. [2016] dataset.

5.4.3.1 NYUD2

We finetune our model on the raw NYUD2 training set and test on the standard 654 images. We set the parameter α of the information gain matrix to be 2.0. We compare our approach against several prior works and report the results in Table 5.6, from which we can see that we achieve state-of-the-art results of 4 evaluation metrics without using any multi-scale network architecture, up-sampling or CRF post-processing. Fig. 5.4 illustrates some qualitative evaluations of our method compared against Liu et al. Liu et al. [2015b] and Eigen et al. Eigen and Fergus [2015].

Table 5.7: Comparison with state-of-the-art results on the KITTI dataset. We cap the maximum depth to 50 and 80 meters to compare with recent works. For the work in Godard et al. [2017], we also report their results with additional training images in the CityScapes dataset Cordts et al. [2016] and denote as Godard et al. CS.

	Accuracy			Error		
	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	rel	rmslog	rms
			Cap 80 meters			
Liu et al. Liu et al. [2015b]	65.6%	88.1%	95.8%	0.217	-	7.046
Eigen et al. Eigen et al. [2014]	69.2%	89.9%	96.7%	0.190	0.270	7.156
Godard et al. Godard et al. [2017]	81.8%	92.9%	96.6%	0.141	0.242	5.849
Godard et al. CS Godard et al. [2017]	83.6%	93.5%	96.8%	0.136	0.236	5.763
Ours	**89.0%**	**96.7%**	**98.4%**	**0.120**	**0.192**	**4.533**
			Cap 50 meters			
Garg et al. Garg and Reid [2016]	74.0%	90.4%	96.2%	0.169	0.273	5.104
Godard et al. Godard et al. [2017]	84.3%	94.2%	97.2%	0.123	0.221	5.061
Godard et al. CS Godard et al. [2017]	85.8%	94.7%	97.4%	0.118	0.215	4.941
Ours	**89.7%**	**96.8%**	**98.4%**	**0.117**	**0.189**	**3.753**

Table 5.8: Comparison with state-of-the-art results on the DIW dataset. The evaluation metric is Weighted Human Disagreement Rate (WHDR).

Method	WHDR
Baseline Chen et al. [2016]	31.37%
Eigen Chen et al. [2016]	25.70%
Chen-NYU Chen et al. [2016]	31.31%
Chen-DIW Chen et al. [2016]	22.14%
Chen-NYU-DIW Chen et al. [2016]	14.39%
Ours-RDIS	18.05%
Ours-NYU-RDIS	**11.55%**

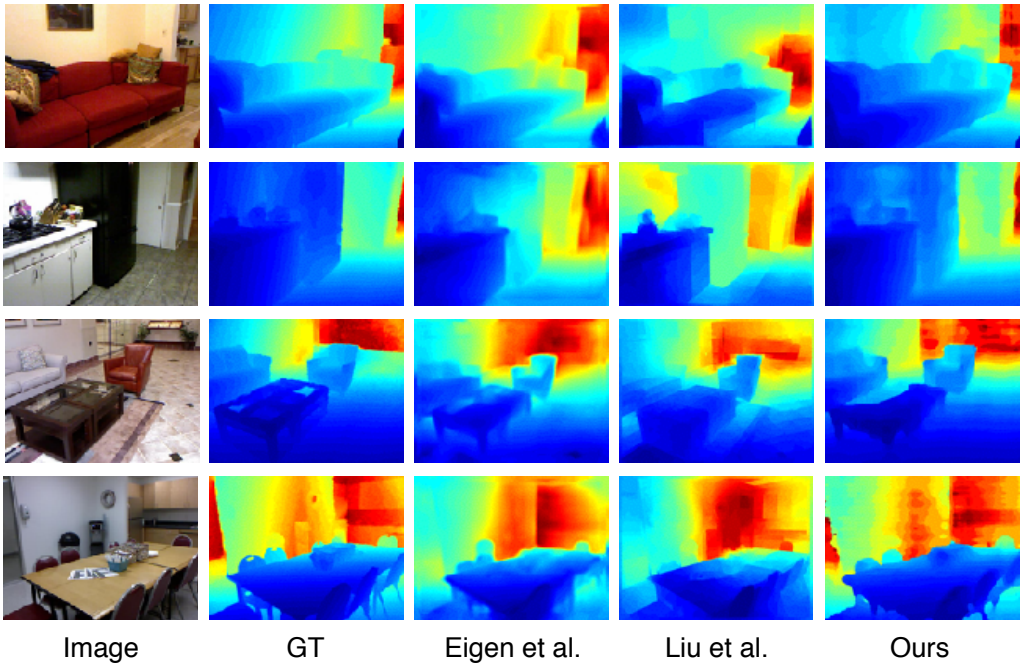

Figure 5.4: Qualitative comparisons with state-of-the-art results on the NYUD2 dataset. The first two columns are RGB images and ground-truth depths respectively. The third row are predictions by Eigen and Fergus [2015], the fourth row are predictions by Liu et al. [2015b], the last row are our predictions. Depths are shown in color (red is far, blue is close).

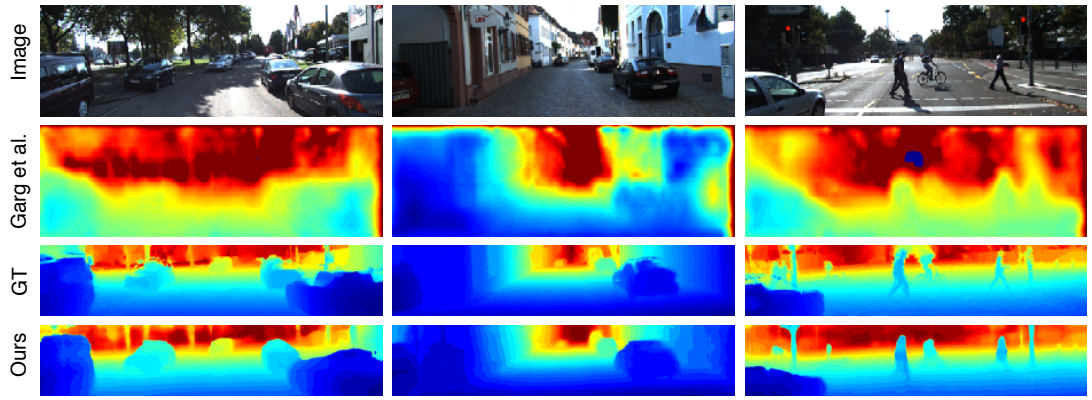

Figure 5.5: Qualitative comparisons with state-of-the-art results on the KITTI dataset. The first row are RGB images, the second row are predictions by Garg and Reid [2016], the last two rows are ground-truth depths and our predictions respectively. Depths are shown in color (red is far, blue is close). Since the ground-truth captured by the velodyne is very sparse, we inpaint the ground-truth for visualization purposes. We also crop the ground-truth and our predictions to mask out the vast sky regions.

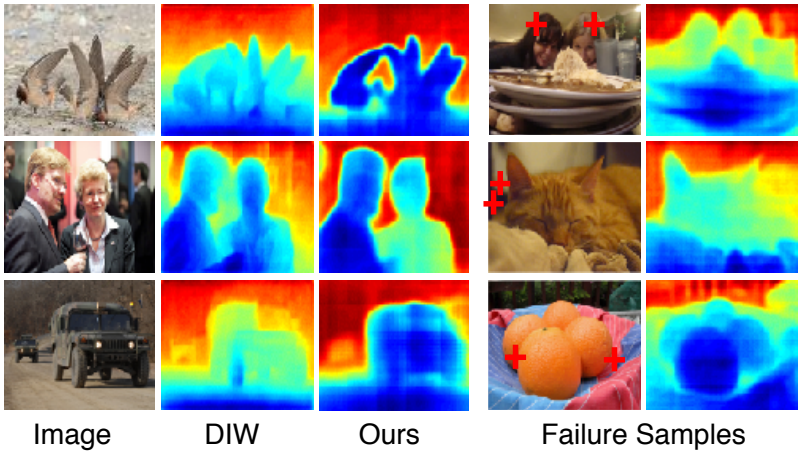

Figure 5.6: Some examples of relative depth estimation of the DIW dataset. The first column are the RGB images, the second column are the predictions of Chen et al. Chen et al. [2016], the third column are our predictions. The last two columns are some failure samples of our approach. The pairs of points labelled with ground-truth ordinal relations are marked as red crosses.

5.4.3.2 KITTI

We finetune our model on the same training set in Godard et al. [2017] which contains 33131 images and test on the same 697 images in Eigen et al. [2014]. But different with the depth estimation method proposed in Godard et al. [2017] which applies both the left and right images in stereo pairs, we only use the left images. The missing values in the ground-truth depth maps are ignored during finetuning and evaluation. We set the parameter α of the information gain matrix to be 0.2. In order to compare with the recent state-of-the-art results, we cap the maximum depth to both 80 meters and 50 meters and present the results in Table 5.7. We outperform state-of-the-art results of all evaluation metrics significantly. Some qualitative results are illustrated in Fig. 5.5. Our method yields outstanding visual predictions.

5.4.3.3 DIW

We evaluate relative depth estimation on the DIW test set and report the WHDR of 7 methods in Table 5.8: 1) a baseline method that uses only the location of the query points: classify the lower point to be closer or guess randomly if the two points are at the same height (Baseline); 2) the model trained by Eigen et al. Eigen and Fergus [2015] on the raw NYUD2 dataset (Eigen); 3) the model trained by Chen et al. Chen et al. [2016] on the raw NYUD2 dataset (Chen-NYU); 4) the model trained by Chen et al. Chen et al. [2016] on the DIW dataset (Chen-DIW). 5) the model by Chen et al. Chen et al. [2016] pretrained on the raw NYUD2 dataset and finetuned on the DIW dataset (Chen-NYU-DIW). 6) our model trained on our RDIS dataset (Ours-RDIS). 7) our model pretrained on the raw NYUD2 dataset and finetuned on our RDIS dataset (Ours-NYU-RDIS). From the table we can see that even though we do

not train our model on the DIW training set, we achieve state-of-the-art result on the DIW test set. We show some of our predicted relative depth maps as well as some failure samples in Fig. 5.6, from which we can see that our predicted relative depth maps are visually better. As for the failure samples, we can also predict satisfactory relative depth maps. Notably, the ground-truth pairs in failure samples are those points with almost equal distance. Given the fact that the equal relation is absent in DIW, we can conclude that we reach the nearly perfect performance on the DIW test set.

5.5 Conclusion

We have proposed a new dataset Relative Depths in Stereo (RDIS) containing images labelled with dense relative depths. The ground-truth relative depths are obtained through existing stereo algorithm as well as manual post-processing. We have shown that augmenting benchmark RGB-D datasets with our proposed RDIS dataset, the performance of single-image depth estimation can be improved significantly.

Note that the goal of this work is to predict depths from single monocular images. However the application of our proposed RDIS dataset is not limited to this. With the learning scheme based on relative depths, we can perform 2D-to-3D conversion like Deep3D Xie et al. [2016]. We leave this as our future work.

Chapter 6

Conclusion

6.1 Conclusion

In this thesis, we have proposed a set of algorithms for RGB-D vision tasks, including monocular depth estimation, RGB-D object detection and semantic segmentation, all built upon state-of-the-art deep learning architectures. To sum up, they are:

- Two RGB-D object detection algorithms and two RGB-D semantic segmentation algorithms that are presented in Chapter 3. These algorithms are based on deep convolutional neural fields (DCNF) model which exploits depths from single monocular images. The exploited depths are combined with RGB data in different ways to improve the performance of object detection and semantic segmentation.
- A monocular depth estimation algorithm which formulates depth estimation as classification is presented in Chapter 4. By formulating depth estimation as classification, we can easily obtain the confidence of a depth prediction. The confidence can be used during training as well as post-processing.
- A monocular depth estimation algorithm which employs relative depths as additional training data is presented in Chapter 5. The relative depths are extracted from stereo videos using existing stereo matching algorithm. By doing so we have proposed a new Relative Depth in Stereo (RDIS) dataset that densely labelled with relative depths.

6.2 Future directions

Even though we have made considerable progress in several RGB-D vision tasks in this thesis, several possible directions are still worth exploring. We list some future directions as follows.

- We have proposed to employ relative depths from stereo videos in Chapter 5. The relative depths are only used to improve the performance of metric depth estimation. With our proposed Relative Depth in Stereo (RDIS) dataset, we can train CNNs that are able to predict relative depth maps (disparities) from a given image, and thus generate a new view to formulate a stereo vision. We

can also embed the depth image based rendering (DIBR) in the network so the CNNs can be trained in an end-to-end style.

- The generative adversarial network (GAN) Goodfellow et al. [2014] has become a popular research topic and has been applied to many unsupervised learning tasks such as image super-resolution, image generation and image denoising. In recent years, GAN has been used in supervised dense prediction tasks such as semantic segmentation and achieves outstanding performance. With the models proposed in this thesis being viewed as generative models, we can also apply GAN to our monocular depth estimation or the aforementioned stereo view generation.

Bibliography

Ameesh, M.; IV, P. A.; and Kostas, D., 2006. Fully automatic registration of 3d point clouds. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 14)

Appia, V. and Batur, U., 2014a. Fully automatic 2d to 3d conversion with aid of high-level image features. In *Proc. Stereo. Disp. App.* (cited on page 13)

Appia, V. and Batur, U., 2014b. Fully automatic 2d to 3d conversion with aid of high-level image features. 9011 (02 2014).

Bo, L.; Ren, X.; and Fox, D., 2011. Depth kernel descriptors for object recognition. In *Proc. IEEE/RSJ Int. Conf. Intell. Robt. Syst.* (cited on pages 17 and 19)

Brachmann, E.; Michel, F.; Krull, A.; Yang, M. Y.; Gumhold, S.; and Rother, C., 2016. Uncertainty-driven 6d pose estimation of objects and scenes from a single RGB image. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 14)

Cao, Z.; Qin, T.; Liu, T.-Y.; Tsai, M.-F.; and Li, H., 2007. Learning to rank: from pairwise approach to listwise approach. In *Proc. Int. Conf. Mach. Learn.* (cited on page 62)

Chen, L.; Papandreou, G.; Kokkinos, I.; Murphy, K.; and Yuille, A. L., 2015a. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *Proc. IEEE Int. Conf. Learn. Rep.* (cited on page 42)

Chen, T.; Li, M.; Li, Y.; Lin, M.; Wang, N.; Wang, M.; Xiao, T.; Xu, B.; Zhang, C.; and Zhang, Z., 2015b. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. In *Proc. Adv. Neural Inf. Process. Syst.* (cited on page 68)

Chen, W.; Fu, Z.; Yang, D.; and Deng, J., 2016. Single-image depth perception in the wild. In *Proc. Adv. Neural Inf. Process. Syst.* (cited on pages xvii, 59, 62, 69, 72, 73, and 75)

Chu, X.; Yang, W.; Ouyang, W.; Ma, C.; Yuille, A. L.; and Wang, X., 2017. Multi-context attention for human pose estimation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 14)

Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; and Schiele, B., 2016. The cityscapes dataset for semantic urban scene understanding. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages xx, xxi, 55, and 73)

Csurka, G.; Dance, C. R.; Fan, L.; Willamowski, J.; and Bray, C., 2004. Visual categorization with bags of keypoints. In *Proc. Eur. Conf. Comp. Vis.*, 1–22. (cited on pages 3 and 20)

Dalal, N. and Triggs, B., 2005. Histograms of oriented gradients for human detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 3 and 20)

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L., 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 7 and 10)

Eckart, B.; Kim, K.; Troccoli, A.; Kelly, A.; and Kautz, J., 2016. Accelerated generative models for 3d point cloud data. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 13)

Eigen, D. and Fergus, R., 2015. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proc. IEEE Int. Conf. Comp. Vis.* (cited on pages xvi, xvii, 11, 41, 42, 43, 53, 54, 56, 61, 72, 74, and 75)

Eigen, D.; Puhrsch, C.; and Fergus, R., 2014. Depth map prediction from a single image using a multi-scale deep network. In *Proc. Adv. Neural Inf. Process. Syst.*, 2366–2374. (cited on pages 11, 18, 19, 42, 50, 54, 55, 59, 60, 68, 69, 73, and 75)

Elbaz, G.; Avraham, T.; and Fischer, A., 2017. 3d point cloud registration for localization using a deep neural network auto-encoder. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 13)

Fan, R.; Chang, K.; Hsieh, C.; Wang, X.; and Lin, C., 2008. LIBLINEAR: A library for large linear classification. *J. Mach. Learn. Res.*, (2008). (cited on page 25)

Farabet, C.; Couprie, C.; Najman, L.; and LeCun, Y., 2013. Learning hierarchical features for scene labeling. *IEEE Trans. Pattern Anal. Mach. Intell.*, (2013). (cited on page 20)

Fehn, C., 2004. Depth-image-based rendering (dibr), compression, and transmission for a new approach on 3d-tv. In *Proc. Stereo. Disp. Virt. Real. Sys.* (cited on page 13)

Felzenszwalb, P. F.; Girshick, R. B.; McAllester, D.; and Ramanan, D., 2010. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32, 9 (2010), 1627–1645. (cited on pages 3 and 20)

Flynn, J.; Neulander, I.; Philbin, J.; and Snavely, N., 2016. Deepstereo: Learning to predict new views from the world's imagery. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 61)

Frome, A.; Huber, D.; Kolluri, R.; Bulow, T.; and Malik, J., 2004. Recognizing objects in range data using regional point descriptors. In *Proc. Eur. Conf. Comp. Vis.* (cited on page 14)

Funes-Mora, K. A. and Odobez, J.-M., 2016. Gaze estimation in the 3d space using rgb-d sensors. *Int. J. Comp. Vis.*, 118, 2 (2016), 194–216. (cited on page 14)

Gal, Y. and Ghahramani, Z., 2016. Bayesian convolutional neural networks with Bernoulli approximate variational inference. In *Proc. Int. Conf. Learn Repr.* (cited on page 41)

Garg, R. and Reid, I., 2016. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *Proc. Eur. Conf. Comp. Vis.* (cited on pages xvi, xvii, 12, 55, 57, 59, 61, 73, and 74)

Ge, L.; Liang, H.; Yuan, J.; and Thalmann, D., 2016. Robust 3d hand pose estimation in single depth images: From single-view cnn to multi-view cnns. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 14)

Ge, L.; Liang, H.; Yuan, J.; and Thalmann, D., 2017. 3d convolutional neural networks for efficient and robust hand pose estimation from single depth images. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 15)

Geiger, A.; Lenz, P.; Stiller, C.; and Urtasun, R., 2013. Vision meets robotics: The kitti dataset. *Int. J. Robt. Res.*, (2013). (cited on pages 48, 61, and 68)

Girshick, R., 2015. Fast r-cnn. In *Proc. IEEE Int. Conf. Comp. Vis.* (cited on pages 3, 4, 20, 23, and 33)

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 3, 4, 20, 23, 25, 32, and 34)

Godard, C.; Mac Aodha, O.; and Brostow, G. J., 2017. Unsupervised monocular depth estimation with left-right consistency. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages xx, xxi, 12, 54, 55, 59, 61, 73, and 75)

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y., 2014. Generative adversarial nets. In *Proc. Adv. Neural Inf. Process. Syst.* (cited on page 78)

Gould, S.; Baumstarck, P.; Quigley, M.; Ng, A. Y.; and Koller, D., 2008. Integrating Visual and Range Data for Robotic Object Detection. In *Proc. Eur. Conf. Comp. Vis.* (cited on page 19)

Gupta, A.; Efros, A. A.; and Hebert, M., 2010a. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *Proc. Eur. Conf. Comp. Vis.* (cited on page 11)

Gupta, A.; Hebert, M.; Kanade, T.; and Blei, D. M., 2010b. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *Proc. Adv. Neural Inf. Process. Syst.* (cited on pages 1, 11, 18, 43, and 61)

Gupta, S.; Arbeláez, P. A.; Girshick, R. B.; and Malik, J., 2015. Aligning 3D models to RGB-D images of cluttered scenes. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 14)

Gupta, S.; Girshick, R.; Arbelaez, P.; and Malik, J., 2014. Learning rich features from RGB-D images for object detection and segmentation. In *Proc. Eur. Conf. Comp. Vis.* (cited on pages 17, 19, 30, and 31)

Hariharan, B.; Arbelaez, P.; Bourdev, L.; Maji, S.; and Malik, J., 2011. Semantic contours from inverse detectors. In *Proc. IEEE Int. Conf. Comp. Vis.* (cited on page 36)

Hariharan, B.; Arbeláez, P.; Girshick, R.; and Malik, J., 2014. Simultaneous detection and segmentation. In *Proc. Eur. Conf. Comp. Vis.* (cited on page 36)

Hariharan, B.; Arbeláez, P.; Girshick, R.; and Malik, J., 2015. Hypercolumns for object segmentation and fine-grained localization. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 56)

Harman, P., 2000. Home-based 3d entertainment:an overview. In *Proc. IEEE Int. Conf. Image Process.* (cited on page 13)

He, K.; Zhang, X.; Ren, S.; and Sun, J., 2016a. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 10, 42, 44, 45, 50, 53, 61, 65, and 70)

He, K.; Zhang, X.; Ren, S.; and Sun, J., 2016b. Identity mappings in deep residual networks. In *Proc. Eur. Conf. Comp. Vis.* (cited on page 65)

Hedau, V.; Hoiem, D.; and Forsyth, D., 2010. Thinking inside the box: Using appearance models and context based on room geometry. In *Proc. Eur. Conf. Comp. Vis.*, 224–237. (cited on pages 1, 11, 18, 43, and 61)

Heitz, D. and Koller, G., 2008. Learning spatial context: Using stuff to find things. In *ECCV*. (cited on page 3)

Hinton, G. E. and Salakhutdinov, R. R., 2006. Reducing the dimensionality of data with neural networks. *Science*, 313 (2006). (cited on page 7)

Hirschmuller, H., 2008. Stereo processing by semiglobal matching and mutual information. *IEEE Trans. Pattern Anal. Mach. Intell.*, (2008). (cited on page 62)

Hirschmuller, H. and Scharstein, D., 2009. Evaluation of stereo matching costs on images with radiometric differences. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31, 9 (2009), 1582–1599. (cited on page 62)

Huang, G.; Liu, Z.; van der Maaten, L.; and Weinberger, K. Q., 2017. Densely connected convolutional networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 10)

Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. Int. Conf. Mach. Learn.* (cited on pages 9 and 65)

Jaderberg, M.; Simonyan, K.; Zisserman, A.; and kavukcuoglu, k., 2015. Spatial transformer networks. In *Proc. Adv. Neural Inf. Process. Syst.* (cited on page 12)

Janoch, A.; Karayev, S.; Jia, Y.; Barron, J. T.; Fritz, M.; Saenko, K.; and Darrell, T., 2011. A category-level 3-d object dataset: Putting the kinect to work. In *Proc. IEEE Int. Conf. Comp. Vis.* (cited on pages 19, 30, and 32)

Joachims, T., 1999. Transductive inference for text classification using support vector machines. In *Proc. Int. Conf. Mach. Learn.* (cited on pages 3 and 7)

Johnson, A. E. and Hebert, M., 1999. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21, 5 (1999). (cited on page 14)

Karsch, K.; Liu, C.; and Kang, S. B., 2014. Depthtransfer: Depth extraction from video using non-parametric sampling. *IEEE Trans. Pattern Anal. Mach. Intell.*, (2014). (cited on pages 1, 43, and 61)

Kendall, A.; Badrinarayanan, V.; and Cipolla, R., 2015. Bayesian SegNet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. http://arxiv.org/abs/1511.02680. (cited on pages 41 and 44)

Kolev, K.; Brox, T.; and Cremers, D., 2012. Fast joint estimation of silhouettes and dense 3D geometry from multiple images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34, 3 (2012). (cited on page 15)

Kong, C.; Lin, C.-H.; and Lucey, S., 2017. Using locally corresponding cad models for dense 3d reconstructions from a single image. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 15)

Konrad, J.; Wang, M.; Ishwar, P.; Wu, C.; and Mukherjee, D., 2013. Learning-based, automatic 2d-to-3d image and video conversion. *IEEE Trans. Image Proc.*, 22, 9 (2013). (cited on page 13)

Krähenbühl, P. and Koltun, V., 2011. Efficient inference in fully connected CRFs with gaussian edge potentials. In *Proc. Adv. Neural Inf. Process. Syst.* (cited on pages 42, 44, 47, and 48)

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, 1097–1105. (cited on pages 1, 7, 9, 25, 42, 43, 45, and 61)

Krull, A.; Brachmann, E.; Michel, F.; Yang, M. Y.; Gumhold, S.; and Rother, C., 2015. Learning analysis-by-synthesis for 6d pose estimation in RGB-D images. In *Proc. IEEE Int. Conf. Comp. Vis.* (cited on page 14)

Ladicky, L.; Shi, J.; and Pollefeys, M., 2014. Pulling things out of perspective. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 19 and 43)

Laina, I.; Rupprecht, C.; Belagiannis, V.; Tombari, F.; and Navab, N., 2016. Deeper depth prediction with fully convolutional residual networks. In *Proc. IEEE Int. Conf. 3D Vision*. (cited on pages 43, 44, 53, 54, 55, 60, 61, and 72)

Lazebnik, S.; Schmid, C.; and Ponce, J., 2006. Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 3 and 20)

LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; and Jackel, L. D., 1989. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1, 4 (1989), 541–551. (cited on page 7)

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P., 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*. (cited on page 7)

Li, B.; Shen, C.; Dai, Y.; van den Hengel, A.; and He, M., 2015a. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 11, 41, 43, and 61)

Li, B.; Shen, C.; Dai, Y.; van den Hengel, A.; and He, M., 2015b. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 19)

Lin, D.; Fidler, S.; and Urtasun, R., 2013. Holistic scene understanding for 3d object detection with rgbd cameras. In *Proc. IEEE Int. Conf. Comp. Vis.* (cited on page 19)

Lin, G.; Shen, C.; Reid, I. D.; and van den Hengel, A., 2016. Efficient piecewise training of deep structured models for semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 20, 42, and 52)

Liu, B.; Gould, S.; and Koller, D., 2010. Single image depth estimation from predicted semantic labels. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 11, 19, 21, and 43)

Liu, F.; Shen, C.; and Lin, G., 2015a. Deep convolutional neural fields for depth estimation from a single image. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 59 and 60)

Liu, F.; Shen, C.; Lin, G.; and Reid, I. D., 2015b. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, (2015). (cited on pages xvi, xvii, 4, 11, 18, 19, 20, 41, 42, 43, 53, 54, 55, 56, 61, 72, 73, and 74)

Liu, M.; Salzmann, M.; and He, X., 2014. Discrete-continuous depth estimation from a single image. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 19, 21, and 43)

Long, J.; Shelhamer, E.; and Darrell, T., 2015. Fully convolutional networks for semantic segmentation. *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, (2015). (cited on pages 3, 20, 27, 56, and 61)

Lowe, D. G., 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comp. Vis.*, 60, 2 (2004), 91–110. (cited on pages 3 and 20)

Luo, W.; Schwing, A. G.; and Urtasun, R., 2016. Efficient deep learning for stereo matching. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 59)

Michel, F.; Kirillov, A.; Brachmann, E.; Krull, A.; Gumhold, S.; Savchynskyy, B.; and Rother, C., 2016. Global hypothesis generation for 6d object pose estimation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 14)

Moreno-Noguer, F., 2017. 3d human pose estimation from a single image via distance matrix regression. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 14)

Nguyen, Q. H.; Do, M. N.; and Patel, S. J., 2009. Depth image-based rendering from multiple cameras with 3d propagation algorithm. In *Proc. Int. Conf. Immers. Telecom.* (cited on page 13)

Ning, F.; Delhomme, D.; LeCun, Y.; Piano, F.; Bottou, L.; and Barbano, P. E., 2005. Toward automatic phenotyping of developing embryos from videos. *IEEE Trans. on Image Process*, 14, 9 (2005), 1360–1371. (cited on page 20)

Oquab, M.; Bottou, L.; Laptev, I.; and Sivic, J., 2014. Learning and transferring mid-level image representations using convolutional neural networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 25)

Ouyang, W.; Wang, X.; Zeng, X.; Qiu, S.; Luo, P.; Tian, Y.; Li, H.; Yang, S.; Wang, Z.; Loy, C.-C.; and Tang, X., 2015. Deepid-net: Deformable deep convolutional neural networks for object detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 20)

Pathak, D.; Krähenbühl, P.; Yu, S. X.; and Darrell, T., 2016. Constrained structured regression with convolutional neural networks. http://arxiv.org/abs/1511.07497/. (cited on pages 41 and 44)

Pavlakos, G.; Zhou, X.; Derpanis, K. G.; and Daniilidis, K., 2017. Coarse-to-fine volumetric prediction for single-image 3d human pose. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 14)

Pepik, B.; Stark, M.; Gehler, P.; and Schiele, B., 2012. Teaching 3d geometry to deformable part models. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 19)

Perronnin, F.; Sánchez, J.; and Mensink, T., 2010. Improving the fisher kernel for large-scale image classification. In *Proc. Eur. Conf. Comp. Vis.*, 143–156. (cited on pages 3 and 20)

Pinheiro, P. H. O. and Collobert, R., 2014. Recurrent convolutional neural networks for scene labeling. In *Proc. Int. Conf. Mach. Learn.* (cited on page 20)

Qian, Y.; Gong, M.; and Yang, Y.-H., 2017. Stereo-based 3d reconstruction of dynamic fluid surfaces by global optimization. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 15)

Ren, S.; He, K.; Girshick, R.; and Sun, J., 2015a. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proc. Adv. Neural Inf. Process. Syst.* (cited on page 3)

Ren, S.; He, K.; Girshick, R.; and Sun, J., 2015b. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proc. Adv. Neural Inf. Process. Syst.* (cited on page 33)

Ren, Z. and Sudderth, E. B., 2016. Three-dimensional object detection and layout prediction using clouds of oriented gradients. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 14)

Roy, A. and Todorovic, S., 2016. Monocular depth estimation using neural regression forest. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 43, 53, and 54)

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J., 1986. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, 318–362. MIT Press. (cited on page 7)

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L., 2015. Imagenet large scale visual recognition challenge. *Int. J. Comp. Vis.*, 115, 3 (2015), 211–252. (cited on pages 8, 10, and 65)

Russell, B. C. and Torralba, A., 2009. Building a database of 3d scenes from user annotations. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 19 and 43)

Rusu, R. B.; Blodow, N.; Marton, Z. C.; and Beetz, M., 2008. Aligning Point Cloud Views using Persistent Feature Histograms. In *Proc. IEEE/RSJ Int. Conf. Intell. Robt. Syst.* (cited on page 14)

Saxena, A.; Chung, S. H.; and Ng, A. Y., 2007. 3-d depth reconstruction from a single still image. *Int. J. Comp. Vis.*, (2007). (cited on pages 19, 30, and 43)

Saxena, A.; Ng, A.; and Chung, S., 2005. Learning Depth from Single Monocular Images. *Proc. Adv. Neural Inf. Process. Syst.*, (2005). (cited on pages 11, 19, 43, and 59)

Saxena, A.; Sun, M.; and Ng, A. Y., 2009. Make3d: Learning 3d scene structure from a single still image. *IEEE Trans. Pattern Anal. Mach. Intell.*, (2009). (cited on pages 19, 21, 43, and 61)

Schwarz, M.; Schulz, H.; and Behnke, S., 2015. RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features. In *Proc. IEEE Conf. Robt. Auto.* (cited on page 19)

Schwing, A. G. and Urtasun, R., 2012. Efficient Exact Inference for 3D Indoor Scene Understanding. In *Proc. Eur. Conf. Comp. Vis.* (cited on pages 18, 43, and 61)

Schwing, A. G. and Urtasun, R., 2015. Fully connected deep structured networks. http://arxiv.org/abs/1503.02351. (cited on page 20)

Silberman, N.; Hoiem, D.; Kohli, P.; and Fergus, R., 2012. Indoor segmentation and support inference from rgbd images. In *Proc. Eur. Conf. Comp. Vis.* (cited on pages 30, 48, 61, and 68)

Simonyan, K.; Vedaldi, A.; and Zisserman, A., 2013. Deep fisher networks for large-scale image classification. In *Proc. Adv. Neural Inf. Process. Syst.* (cited on pages 3 and 20)

Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. In *Proc. IEEE Int. Conf. Learn. Rep.* (cited on pages 10, 26, 27, 42, 45, 51, and 61)

Sinha, A.; Choi, C.; and Ramani, K., 2016. Deephand: Robust hand pose estimation by completing a matrix imputed with deep features. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 14)

Sivic, J. and Zisserman, A., 2003. Video Google: A text retrieval approach to object matching in videos. In *Proc. IEEE Int. Conf. Comp. Vis.*, vol. 2, 1470–1477. (cited on page 20)

Song, S.; Lichtenberg, S. P.; and Xiao, J., 2015. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 35 and 54)

Song, S. and Xiao, J., 2014. Sliding shapes for 3d object detection in depth images. In *Proc. Eur. Conf. Comp. Vis.* (cited on pages 14, 17, and 19)

Song, S. and Xiao, J., 2016. Deep sliding shapes for amodal 3d object detection in RGB-D images. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 14)

Spyropoulos, A.; Komodakis, N.; and Mordohai, P., 2014. Learning to detect ground control points for improving the accuracy of stereo matching. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 1621–1628. (cited on page 59)

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15, 1 (2014). (cited on page 9)

Stefan, W.; Konrad, S.; and Schiele, B., 2010. Disparity statistics for pedestrian detection: Combining appearance, motion and stereo. In *Proc. Eur. Conf. Comp. Vis.* (cited on page 19)

Sun, M.; Bradski, G.; Xu, B.-X.; and Savarese, S., 2010. Depth-encoded hough voting for coherent object detection, pose estimation, and shape recovery. In *Proc. Eur. Conf. Comp. Vis.* (cited on page 19)

Sung-Ho, B.; Mohamed, E.; Mohamed, H.; and Wojciech, M., 2017. Efficient and scalable view generation from a single image using fully convolutional networks. https://arxiv.org/pdf/1705.03737.pdf. (cited on page 13)

Suykens, J. A. K. and Vandewalle, J., 1999. Least squares support vector machine classifiers. *Neural Process. Lett.*, 9, 3 (1999). (cited on page 7)

Toeppe, E.; Nieuwenhuis, C.; and Cremers, D., 2013. Relative volume constraints for single view 3d reconstruction. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 15)

Tombari, F.; Salti, S.; and Di Stefano, L., 2010. Unique signatures of histograms for local surface description. In *Proc. Eur. Conf. Comp. Vis.* (cited on page 14)

Uijlings, J.; van de Sande, K.; Gevers, T.; and Smeulders, A., 2013. Selective search for object recognition. *Int. J. Comp. Vis.*, (2013). (cited on page 30)

Wan, C.; Probst, T.; Van Gool, L.; and Yao, A., 2017. Crossing nets: Combining gans and vaes with a shared latent space for hand pose estimation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 14)

Wang, P.; Shen, X.; Lin, Z.; Cohen, S.; Price, B.; and Yuille, A. L., 2015. Towards unified depth and semantic prediction from a single image. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 11, 41, 43, 53, 54, 61, and 72)

Wu, Z.; Shen, C.; and van den Hengel, A., 2016. Wider or deeper: Revisiting the resnet model for visual recognition. https://arxiv.org/pdf/1611.10080.pdf. (cited on page 65)

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J., 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 14)

Xie, J.; Girshick, R.; and Farhadi, A., 2016. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In *Proc. Eur. Conf. Comp. Vis.* (cited on pages xxi, 13, 59, 61, 69, 70, and 76)

Yu, C.-N. J. and Joachims, T., 2009. Learning structural svms with latent variables. In *Proc. Int. Conf. Mach. Learn.* (cited on pages 3 and 7)

Zbontar, J. and LeCun, Y., 2016. Stereo matching by training a convolutional neural network to compare image patches. *J. Mach. Learn. Res.*, 17 (2016). (cited on page 59)

Zeng, A.; Song, S.; Niessner, M.; Fisher, M.; Xiao, J.; and Funkhouser, T., 2017. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 14)

Zhang, J.; Lazebnik, S.; and Schmid, C., 2007. Local features and kernels for classification of texture and object categories: a comprehensive study. *Int. J. Comp. Vis.*, 73 (2007), 2007. (cited on pages 3 and 20)

Zhang, K.; Lu, J.; and Lafruit, G., 2009. Cross-based local stereo matching using orthogonal integral images. *IEEE Trans. Circuits Syst. Video Technol.*, 19, 7 (2009), 1073–1079. (cited on page 59)

Zhang, L.; Vázquez, C.; and Knorr, S., 2011. 3d-tv content creation: Automatic 2d-to-3d video conversion. *IEEE Trans. Broad.*, 57, 2 (2011). (cited on page 13)

Zhang, Y.; Sohn, K.; Villegas, R.; Pan, G.; and Lee, H., 2015. Improving object detection with deep convolutional networks via bayesian optimization and structured prediction. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 20)

Zhou, B.; Lapedriza, A.; Khosla, A.; Oliva, A.; and Torralba, A., 2017. Places: A 10 million image database for scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, (2017). (cited on page 65)

Zhou, B.; Lapedriza, A.; Xiao, J.; Torralba, A.; and Oliva, A., 2014. Learning deep features for scene recognition using places database. In *Proc. Adv. Neural Inf. Process. Syst.* (cited on page 35)

Zhou, X.; Zhu, M.; Leonardos, S.; Derpanis, K. G.; and Daniilidis, K., 2016. Sparseness meets deepness: 3d human pose estimation from monocular video. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 14)

Zoran, D.; Isola, P.; Krishnan, D.; and Freeman, W. T., 2015. Learning ordinal relationships for mid-level vision. In *Proc. IEEE Int. Conf. Comp. Vis.* (cited on pages 59, 61, and 68)