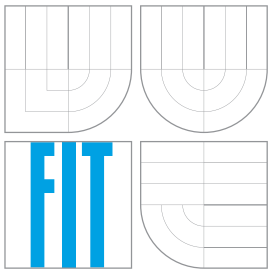
BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

DEPTH ESTIMATION BY CONVOLUTIONAL NEURAL NETWORKS

ODHAD HLOUBKY POMOCÍ KONVOLUČNÍCH NEURONOVÝCH SÍTÍ

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR
AUTOR PRÁCE

Bc. JÁN IVANECKÝ

SUPERVISOR
VEDOUCÍ PRÁCE

Ing. MICHAL HRADIŠ, Ph.D.

BRNO 2016

Zadání diplomové práce/18852/2015/xivane00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2015/2016

Zadání diplomové práce

Řešitel: **Ivanecký Ján, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Odhad hloubky pomocí konvolučních neuronových sítí**
Depth Estimation by Convolutional Neural Networks
Kategorie: Zpracování obrazu

Pokyny:

1. Prostudujte základy konvolučních neuronových sítí a jejich učení.
2. Vytvořte si přehled o současných metodách využívajících konvoluční sítě pro odhad hloubkové mapy z obrazu.
3. Navrhněte metodu odhadu hloubky z obrazu, nebo inovativním způsobem upravte existující metodu.
4. Obstarejte si databázi vhodnou pro experimenty.
5. Implementujte navrženou metodu a proveďte experimenty nad datovou sadou.
6. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte stručný plakát nebo video prezentující vaši práci, její cíle a výsledky.

Literatura:

- Krizhevsky, A., Sutskever, I. and Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012.
- Eigen et al.: Depth Map Prediction from a Single Image using a Multi-Scale Deep Network, NIPS 2014.
- Liu et al.: Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields. arXiv:1502.07411, 2015.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 - 3.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Hradiš Michal, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 25. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstract

This thesis deals with depth estimation using convolutional neural networks. I propose a three-part model as a solution to this problem. The model contains a global context network which estimates coarse depth structure of the scene, a gradient network which estimates depth gradients and a refining network which utilizes the outputs of previous two networks to produce the final depth map. Additionally, I present a normalized loss function for training neural networks. Applying normalized loss function results in better estimates of the scene's relative depth structure, however it results in a loss of information about the absolute scale of the scene.

Abstrakt

Táto práca sa zaoberá odhadom hĺbky s použitím konvolučných neurónových sietí. Pre vyriešenie tohto problému je v práci navrhnutý model skladajúci sa z troch častí. Model sa skladá zo siete globálneho kontextu, ktorá odhaduje hrubú štruktúru scény, gradientovej siete, ktorá odhaduje hĺbkové gradienty a zjemňujúcej siete, ktorá na základe výstupov z predchádzajúcich dvoch sietí odhadne konečnú hĺbkovú mapu. Ďalej v práci navrhujem normalizovanú chybovú funkciu na trénovanie neuronových sietí. Použitie tejto chybovej funkcie zlepšuje odhady relatívnej hlbkovej štruktúry scény, za cenu straty informácie o absolutnej hlbke v scéne.

Keywords

depth estimation, convolutional neural networks, global context network, gradient network, refining network, NYU Depth v2., data augmentation, normalized loss, Caffe

Klíčová slova

odhad hĺbky, konvolučné neuronové siete, sieť globálneho kontextu, gradientová sieť, zjemňujúca sieť, NYU Depth v2., rozšírovanie dát, normalizovaná chybová funkcia, Caffe

Reference

IVANECKÝ, Ján. *Depth Estimation by Convolutional Neural Networks.* Brno, 2016. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Hradiš Michal.

Depth Estimation by Convolutional Neural Networks

Declaration

I hereby declare that this bachelor's thesis was prepared as an original author's work under the supervision of Ing. Michal Hradiš, PhD. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Ján Ivanecký
May 25, 2016

Acknowledgements

I'd like to thank my supervisor Ing. Michal Hradiš, PhD. for the time he dedicate to me, his advice and guidance. Meetings with him were always productive and gave me a push in the right direction.

Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

Contents

Chapter 1

Introduction

One of the aims of computer vision is to understand a scene depicted in an image. This requires the ability to infer the semantics of the scene and the understanding of the geometric structure of the scene from a single image. An important part of solving the latter part of the problem involves the estimation of the corresponding depth map which can be subsequently used to infer the three-dimensional structure of the scene. Knowledge of the scene's depth map has also been applied to other problems in computer vision such as semantic labeling [11] or pose estimation [23]. In contrast to depth estimation from multiple images, estimating depth from a single image is not a well studied problem. Solutions estimating depth from a single image have usually relied on simplifying assumptions about the geometric structure of the scene, or some external information about the scene such as semantic labels [14]. Recent approaches remove the need for these assumptions and for the additional information and instead utilize supervised learning while relying entirely on cues that can be inferred from the input image.

This thesis focuses on the task of depth map estimation from a single image using convolutional neural networks. Convolutional neural networks were not used for this task until recently [5], but approaches utilizing them have quickly achieved state of the art performance and surpassed prior solutions. Aim of this project is to study different ways in which convolutional neural networks can be used to help in the task of depth estimation from a single image, propose an innovative solution to this problem, experiment with it and evaluate results of these experiments.

Chapter 2 introduces the problem of depth estimation from a single image and present the existing solutions employing convolutional neural networks. Chapter 3 contains the description of the proposed model a novel loss function used to improve performance of the model. Chapter 4 describes the implementation details of the proposed model and Chapter 5 describes the experiments conducted on the model and their results.

Chapter 2

Depth Estimation

Depth estimation can be defined as a mapping from an RGB image, or a series of RGB images to a depth map, or a series of depth maps. Estimated depth maps can then be used for 3D scene reconstruction. Most existing approaches estimate depth based on stereo vision [21] or video footage [1]. Given multiple view of the scene and accurate correspondence among images, depth estimation is deterministic. By contrast, estimating depth from a single image is an ill-posed problem, since the same image could be produced by an infinite amount of real world scenes. Considering only realistic ones, there's still an issue of scale invariance (see Section 2.1.4). To infer depth map from a single image, a solution has to rely only on monocular depth cues like perspective and line angles. This limitation has been avoided by utilizing outside information about the scene or by simplifying assumptions about the scene's structure, until recently.

Existing solutions to depth estimation from a single image usually rely on assumptions regarding the scene's general structure by fitting box-shaped model to the image [22]. This simple geometric model cannot predict more detailed depth structure. More recent solutions propose estimating depth directly for image patches - *superpixels*, but since these parts of image don't contain enough information to infer absolute depth, those approaches usually utilize other information such as semantic labels [14]. Most recent approaches try to predict depth from an RGB image without any additional information. These approaches usually use Markov Random Fields [20] or Conditional Random Fields [16]. Starting with work by Eigen *et al.* [5], convolutional neural networks (CNNs) are used for depth estimation and at the moment, approaches that utilize CNNs achieve state of the art performance.

This chapter serves as a brief introduction to the context of depth estimation from a single image using CNNs. In the first section I introduce concepts relevant to the task of depth estimation, like importance of the global context or scale invariance. In the second section I present commonly used metrics used for benchmarking performance of systems estimating depth. Third section lists commonly used RGB-D datasets. Fourth section is the main part of this chapter and presents existing solutions to depth estimation from a single image which utilize CNNs.

2.1 Important Concepts for Depth Estimation

In this section I introduce and briefly describe useful concepts that provide conceptual background for existing solutions which I present later in the chapter.

2.1.1 Logarithmic Space

Depth value y in logarithmic space (referred to as log space from now on) is equal to $log(y_{lin})$, where y_{lin} is the real world depth in linear space. Logarithmic function used in this thesis to convert from linear space is $f(x) = 0.179581 * ln(x) + 1$. This function is designed to be an identity for values $0.0039 \sim 1/255$ and 1. In this thesis, 1 represents 10 meters. For depth values in log space, Euclidean loss does not minimize the difference between estimated depth y and the ground truth y^*, but minimizes the difference between $logy$ and $logy^*$, which can be rewritten as $log\frac{y}{y^*}$. This means that loss in the log space optimizes ratio between y and y^* and achieves minimum when the ratio is 1. Experiments in Chapter 5 show that this achieves better prediction of relative depth structure.

2.1.2 Global Context

Estimating depth from a single image has to rely solely on visual cues which indicate the scene's spatial structure such as vanishing points, object locations and room alignment. These cannot be inferred from individual local patches [5] and require a global view of the input image as well. Eigen *et al.* [5] use a convolutional neural network with fully connected layers that estimates the global structure of the scene. More traditionally, Markov Random Fields or Conditional Random Fields are used to propagate local information through the whole scene [20].

2.1.3 Absolute Scale of the Scene and the Relative Structure of the Scene

In the context of depth estimation, absolute scale of the scene represents the size of the scene in the real world. Depth map which accurately reflects the absolute scale of the scene contains depth values whose absolute values are similar to the target depth values. Conversely, depth map which accurately reflects the relative structure of the scene has the same relationships between the pixels (higher/lower depth) as the target depth map, but the absolute depth values are different from the target depth values.

2.1.4 Scale Invariance

Estimating depth from a single image has to deal with ambiguity about the scene's scale that is caused by the projection from 3D space to the 2D image. Two images that look identically, can in fact depict different real world scenes - image of a circle with a radius of 5 meters captured from the distance 10 meters will look identically to the image of a circle with a radius of 5 centimeters captured from the distance of 10 centimeters. Since the information about the scene's absolute scale cannot be extracted from the image by means other than understanding the semantics of the depicted scene, it is reasonable to consider both cases to be identical. There are multiple approaches to achieve this, for example by normalizing the training dataset or by using a suitable loss function during the training convolutional neural network.

Error function that considers scale invariance should output the same error for the output depths $\mathbf{y}$ as for $\mathbf{y} * s, \forall s \in \mathbf{R}$. Eigen *et al.* [5] use such loss function and achieve better qualitative results.

2.1.5 Neighboring Pixels Relationship

When looking at an RGB image, it is intuitive that regions with similar appearance will have similar depth values in the resulting depth map. This can be incorporated into system for depth estimation by enforcing similarity between neighboring depth values based on similarity of the appearance of corresponding regions in the RGB image. This causes the resulting depth map to have sharper edges and to better align to local structure as seen in [15].

Liu *et al.* [15], Li *et al.* [13] and Wang *et al.* [25] use Conditional Random Fields (CRF) with an appropriate pair-wise potential to enforce similarity across homogeneous image regions (superpixels). Roy *et al.* [19] predict depth of a pixel as a weighted average of the neighboring pixels' depth values with weights being higher for neighboring pixels similar in appearance.

2.1.6 Utilizing Depth Map Gradients

Horizontal and vertical gradients of the depth map convey information about significant depth differences in the scene and local structure, which can be used to improve estimated depth maps.

Roy *et al.* [19] estimate the gradient of the depth map in addition to estimating global context depth map. Both are used for a final depth prediction. Eigen *et al.* [4] use a loss function that minimizes difference between gradients of the estimated depth map and gradients of the ground truth and achieve an increase in model's performance.

2.2 Metrics for Evaluating Performance

Evaluating results quantitatively is important for benchmarking performance and ability to compare existing solutions. In Table 2.1 I present common metrics used for evaluating the performance of systems performing depth estimation. d_i is the predicted depth at pixel i and d_i^* is the target depth for the pixel. Table 2.2 at the end of this chapter, compares performance of existing solutions using these metrics.

Relative error (rel)	$\frac{1}{\lvert N\rvert}\sum_{i\in N}\frac{\lvert d_i-d_i^*\rvert}{d_i^*}$
Square relative error (sqr rel)	$\frac{1}{\lvert N\rvert}\sum_{i\in N}\frac{\lvert d_i-d_i^*\rvert^2}{d_i^*}$
Root mean squared error (rms)	$\sqrt{\frac{1}{\lvert N\rvert}\sum_{i\in N}\lvert d_i-d_i^*\rvert^2}$
Root mean squared error log (rms-log)	$\sqrt{\frac{1}{\lvert N\rvert}\sum_{i\in N}\lvert\log(d_i)-\log(d_i^*)\rvert^2}$
Log10 error (log10)	$\frac{1}{\lvert N\rvert}\sum_{i\in N}\lvert\log_{10}(d_i)-\log_{10}(d_i^*)\rvert$
Threshold (δ)	% of d_i s.t. $\max(\frac{d_i}{d_i^*},\frac{d_i^*}{d_i})<thr$, where $thr\in\{1.25,1.25^2,1.25^3\}$

Table 2.1: Metrics used for evaluating performance in depth estimation

For every metric except for the threshold metric, lower values indicate better performance. For threshold metric, higher values mean lower error.

2.3 Datasets

This section describes RGB-D datasets that are commonly used to benchmark the performance of a system for depth estimation.

NYU Depth v2 [18] is comprised of video sequences of 464 indoor scenes recorded with Microsoft Kinect. The dataset contains two components, the labeled dataset with 1449 (795 in the training set, 654 in the test set) aligned RGB and depth images with a resolution of 640x480 and semantic labels for each object. The second component contains raw outputs from camera and Microsoft Kinect. This results in 407,024 unlabeled images total. The dataset also provides toolbox with MATLAB scripts that can be used to process the provided raw data and obtain synced, aligned and complete depth maps. Depth values for both components are in the range 0-10 meters.

Make3D [20] is a dataset of 534 outdoor images. 400 images are used for training, 134 are used for testing. Depth information is gathered using laser and resulting RGB images have a resolution of 2272x1704. Depth maps have a 55x305 resolution with depth ranging from 0 to 80 meters.

KITTI [7] consists of video sequences taken from a driving vehicle with depths captured by LIDAR. Provided RGB images have a resolution of 1392x512 and depth maps have to be extracted from LIDAR point cloud data. These point are provided only for the bottom portion of the image. Dataset consists of 56 scenes, and contains around 20,000 images overall.

2.4 Solutions Using Convolutional Neural Networks

This section presents existing solutions to the depth estimation task that employ convolutional neural networks.

2.4.1 Solutions Using Only Convolutional Neural Networks

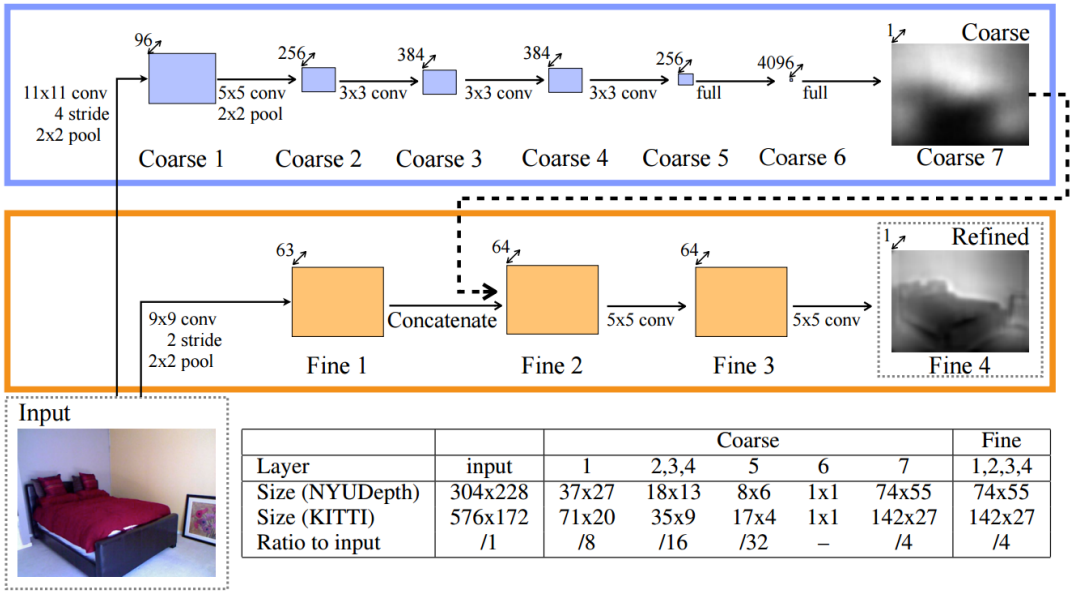
Depth Map Prediction from a Single Image Using a Multi-Scale Deep Network

Eigen *et al.* [5] were first to propose a solution to depth estimation that uses convolutional neural networks. They use two-scale architecture consisting of the *coarse-scale* network and the *fine-scale* network. Coarse-scale network is a convolutional neural network that identifies the global scene context. This is accomplished by using two fully connected layers that have a full view of the input image. Output of the coarse-scale network is a low resolution depth map. This depth map, along with the original input image is then fed to the fine-scale network. Fine-scale network is a fully convolutional network consisting of three convolutional layers and is used to refine the coarse prediction it receives. The whole model architecture can be seen in detail in the Figure 2.1.

In addition, Eigen *et al.* consider the issue of scale invariance. They use a *scale-invariant error* for performance evaluation and a *scale-invariant loss function* $L(y, y*)$ for the training:

$$L(y, y^*) = \frac{1}{n}\sum_i (\log y_i - \log y_i^* + \frac{1}{n}\sum_j (\log y_j^* - \log y_j))^2. \tag{2.1}$$

y_i denotes the predicted depth value for the pixel i, $y*_i$ is the ground truth value for the pixel. The scale invariance is accomplished by the inner sum, which represents the mean



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

Figure 2.1: Multiscale architecture of Eigen *et al.* [5]

log difference between predicted and ground truth depth values. Why this results in scale invariance, can be seen if the equation is rewritten as

$$\begin{aligned} L(y, y^*) &= \frac{1}{n}\sum_i((\log y_i - \frac{1}{n}\sum_j(\log y_j)) - (\log y_i^* - \frac{1}{n}\sum_j(\log y_j^*)))^2 \\ &= \frac{1}{n}\sum_i((\log y_i - y_m) - (\log y_i^* - y_m^*))^2 \\ &= \frac{1}{n}\sum_i(\log\frac{y_i}{y_m} - \log\frac{y_i^*}{y_m^*})^2, \end{aligned} \tag{2.2}$$

where y_m is the mean predicted log depth and y_m^* is the mean ground truth log depth. It is clear from the last line of the equation 2.2 that the error will be the same for all scalar multiples of predicted depth values y_i, hence the scale invariance. Eigen *et al.* use another form of the equation 2.2,

$$L(y, y^*) = \frac{1}{n}\sum_i(\log y_i - \log y_i^*)^2 - \frac{\lambda}{n^2}(\sum_i \log y_i - \log y_i^*)^2, \tag{2.3}$$

which can be computed in linear time. λ controls the effect of scale invariant term. When $\lambda = 0$, the equation is reduced to an Euclidean loss. When $\lambda = 1$, equation is equivalent to scale invariant loss as defined in Equation 2.2. They find that using $\lambda = 0.5$ produces good absolute-scale predictions and still benefits from scale invariance.

They train and evaluate the network using NYU Depth v2 dataset (full raw data) [18] and KITTI dataset [7]. To increase training data variability, they augment training set on-line by using these random transformations (taken directly from [5]):

- *Scale*: Input and target images are scaled by $s \in [1, 1.5]$, and the depths are divided by s.

- *Rotation*: Input and target are rotated by $r \in [-5, 5]$ degrees.
- *Translation*: Input and target are randomly cropped to the input size of the network.
- *Color*: Input values are multiplied globally by a random RGB value $c \in [0.8, 1.2]^3$.
- *Flips*: Input and target are horizontally flipped with 0.5 probability.

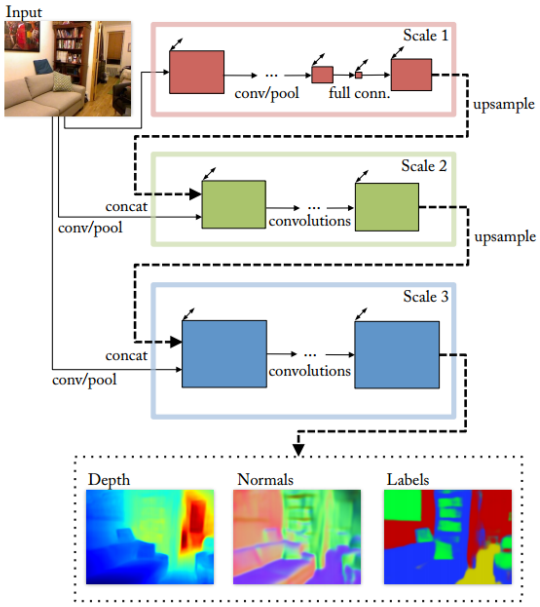
During training process, they train the coarse-scale network for 2M steps, after that they fix the coarse-scale network and train the fine-scale network for 1.5M steps.

Results show that at the time of writing the paper, the model used in this network achieves on average 35% performance increase compared to the runner-up (who is not using convolutional neural networks) on the NYU Depth v2 dataset and on average 31% performance increase on the KITTI dataset.

Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture

In [4], Eigen *et al.* build on their previous work and introduce a *three-scale* architecture for tasks of depth estimation, surface normals prediction and semantic labeling. The architecture is the same for each of the three tasks, but there are differences in datasets, losses and evaluation. I present only the information and results relevant to the depth estimation task.

Proposed three-scale architecture improves the architecture proposed in [5]. First-scale network is a network similar to one in [5]. Difference is that it is deeper, using more convolutional layers and its output is not a depth estimate but multichannel feature maps. Eigen *et al.* tried different sizes for this network, one based on AlexNet [10] and one based on VGG [24]. The *second-scale* network takes as input the original input image as well as feature maps from the first-scale network. Second-scale network is a fully convolutional network with 5 layers and its output is a depth estimate in a half the final resolution. This estimate is then fed to the third-scale network, along with the original input image and is refined to produce the final depth estimate. Third-scale network is also a fully convolutional network, with 4 layers. See Figure 2.2 for more detailed description of used architecture.



	Layer	1.1	1.2	1.3	1.4	1.5	1.6	1.7	upsamp
Scale 1 (AlexNet)	Size	37x27	18x13	18x13	18x13	8x6	1x1	19x14	74x55
	#convs	1	1	1	1	1	–	–	–
	#chan	96	256	384	384	256	4096	64	64
	ker. sz	11x11	5x5	3x3	3x3	3x3	–	–	–
	Ratio	/8	/16	/16	/16	/32	–	/16	/4
	l.rate	0.001	0.001	0.001	0.001	0.001	see text		

	Layer	1.1	1.2	1.3	1.4	1.5	1.6	1.7	upsamp
Scale 1 (VGG)	Size	150x112	75x56	37x28	18x14	9x7	1x1	19x14	74x55
	#convs	2	2	3	3	3	–	–	–
	#chan	64	128	256	512	512	4096	64	64
	ker. sz	3x3	3x3	3x3	3x3	3x3	–	–	–
	Ratio	/2	/4	/8	/16	/32	–	/16	/4
	l.rate	0.001	0.001	0.001	0.001	0.001	see text		

	Layer	2.1	2.2	2.3	2.4	2.5	upsamp
Scale 2	Size	74x55	74x55	74x55	74x55	74x55	147x109
	#chan	96+64	64	64	64	C	C
	ker. sz	9x9	5x5	5x5	5x5	5x5	–
	Ratio	/4	/4	/4	/4	/4	/2
	l.rate	0.001	0.01	0.01	0.01	0.001	

	Layer	3.1	3.2	3.3	3.4	final
Scale 3	Size	147x109	147x109	147x109	147x109	147x109
	#chan	96+C	64	64	C	C
	ker. sz	9x9	5x5	5x5	5x5	–
	Ratio	/2	/2	/2	/2	/2
	l.rate	0.001	0.01	0.01	0.001	

Figure 2.2: Three-scale architecture of Eigen *et al.* [4]

For training, a similar loss function is used as the one used in [5]:

$$L(y, y^*) = \frac{1}{2n}\sum_i (\log y_i - \log y_i^* + \frac{1}{n}\sum_j (\log y_j^* - \log y_j))^2 + \frac{1}{n}\sum_i [(\nabla_x d_i)^2 + (\nabla_y d_i)^2]. \quad (2.4)$$

$\nabla_x d_i$ and $\nabla_y d_i$ are the horizontal and the vertical gradient of the difference between log depths, d_i being $\log y_i - \log y_i^*$. This gradient term encourages estimates to have a similar local structure to target depth maps and Eigen *et al.* state, that including this term produces better results.

Networks are trained on NYU Depth v2 dataset (raw data), with data augmentation similar to [5]. First, the coarse-scale network and the second-scale network are trained jointly. After that their parameters are fixed and the third network is trained.

Results show that VGG based network significantly outperforms smaller, AlexNet based model, probably due to larger model size. In comparison to other solutions, VGG based network achieves the best performance in all metrics, qualitatively achieving results with better local structure than [5], but still not achieving transitions as sharp as solutions using Conditional Random Fields. The VGG based network is, at the time of writing this thesis, the state of the art in depth estimation from a single image.

2.4.2 Conditional Random Fields Combined with Convolutional Neural Networks

Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields

While in [5] and [4], the depth map is regressed directly from the input image using convolutional neural networks, Liu *et al.* [15] use **Deep Convolutional Neural Fields** (see

Figure 2.3) to explicitly model the relationship between neighboring parts of the depth map using Conditional Random Fields (CRFs). They use neural network to optimize parameters of the probability density function modeled by the CRF. CRF nodes are homogeneous regions of the input image called *superpixels*. The probability density function modeled by CRF is

$$Pr(\mathbf{y}|\mathbf{x}) = \frac{\exp(-E(\mathbf{y},\mathbf{x}))}{Z(\mathbf{x})}. \tag{2.5}$$

$\mathbf{x}$ is the input image and $\mathbf{y}$ is the vector of depth values corresponding to all the superpixels of the input image. $E(\mathbf{y},\mathbf{x})$ is the energy function whose value is being minimized and $Z(\mathbf{x})$ is the normalizing term used to ensure that $Pr(\mathbf{y}|\mathbf{x})$ is a probability density function. Energy function is then defined as

$$E(\mathbf{y},\mathbf{x}) = \sum_{p\in\mathcal{N}} U(y_p,\mathbf{x}) + \sum_{(p,q)\in\mathcal{S}} V(y_p,y_q,\mathbf{x}), \tag{2.6}$$

where $\mathcal{N}$ is the set of superpixels, $\mathcal{S}$ is the set of edges (p,q) connecting superpixels p and q and y_p is the predicted depth value for superpixel p. U is the *unary potential* that aims to regress depth values of individual superpixels and V is the *pairwise potential* that encourages neighboring superpixels with similar appearance to take on similar depth values. The proposed network then consist of two parts. The unary part, which is a convolutional neural network that optimizes the unary potential and the pairwise part, which is a fully connected neural network that optimizes pairwise potential. Input image is segmented into superpixels and an image patch centered around superpixel is then fed to the unary network for each superpixel separately. The unary network has a convolutional part and four fully connected layers following it. Convolutional part was originally taken from AlexNet [10], but later replaced with VGG-16 [24]. Its output is a regressed depth for the superpixel. The unary potential is defined as

$$U(y_p,\mathbf{x};\theta) = (y_p - z_p(\theta))^2, \tag{2.7}$$

where θ are the parameters of the unary convolutional neural network and $z_p(\theta)$ is the regressed depth for the superpixel p using parameters θ. Pairwise part of the model contains a network with one fully connected layer containing a single neuron. This network optimizes the pairwise potential V defined as

$$V(y_p,y_q,\mathbf{x};\beta) = \frac{1}{2}R_{pq}(\beta)(y_p - y_q)^2, \tag{2.8}$$

where β are the pairwise network's parameters and $R_{pq}(\beta)$ is a network's output, a single number. Input to the pairwise network is a vector of 3 similarity observations for each pair of neighboring superpixels. Similarity observations used are the colour difference, colour histogram difference and texture disparity in terms of local binary patterns.

Outputs of unary and pairwise networks are then passed to the CRF loss layer that minimizes the negative log-likelihood of the probability distribution function Pr. Predicting depth values of each superpixel means finding the depth values with the maximum posterior probability:

$$\mathbf{y}^* = \arg\max_{\theta} Pr(\mathbf{y}|\mathbf{x}). \tag{2.9}$$

Note that predicted depth values are in logarithmic space.

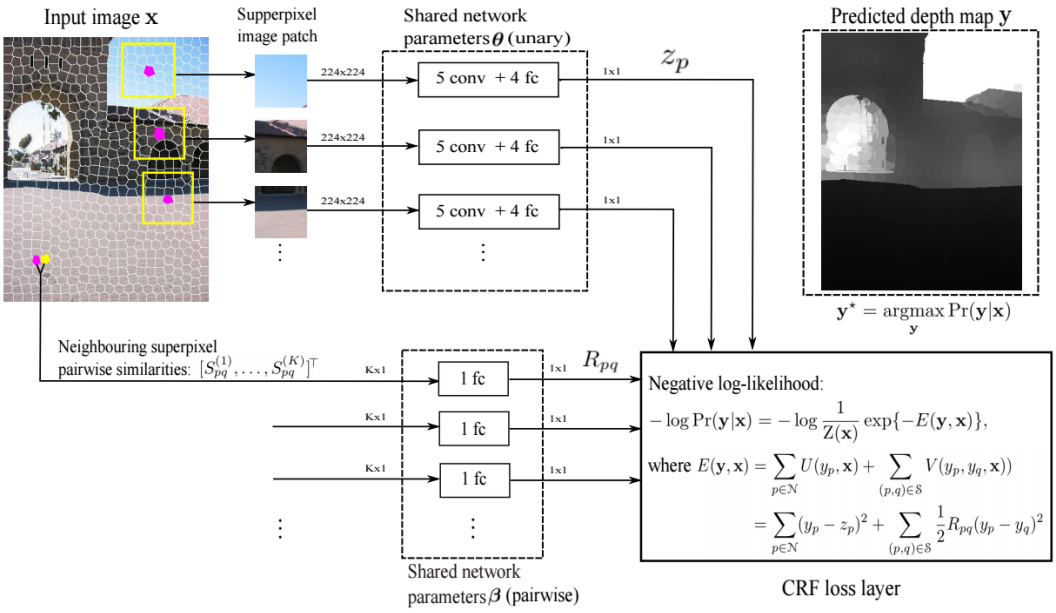


Figure 2.3: Deep Convolutional Neural Fields [15]

Training and evaluation is done on NYU Depth v2 [18], KITTI [7] and Make3D [20] datasets. Both networks are trained jointly with CRF using Stochastic Gradient Descent (SGD) to minimize the energy function E. Quantitative results are compared directly to the results of Eigen *et al.* [5] on NYU Depth v2 and KITTI datasets and proposed model achieves better performance in all metrics. They achieve better performance even though they train the model only on the labeled dataset containing 795 images, compared to the full raw dataset containing 100 000+ pictures used by Eigen *et al.* This is possibly due to more complicated model used by Liu *et al.*. Qualitatively, results are visually better then results in [5], with sharper transitions and are aligned to local structures. This can be attributed to the pairwise potential, causing parts of the input image with similar appearance to have similar depth values. More recent work done by Eigen *et al.* [4] achieves better results in all metrics, but this can be attributed to the larger architecture. Note that Liu *et al.* don't consider the issue of scale invariance.

Depth and Surface Normal Estimation from Monocular Images Using Regression on Deep Features and Hierarchical CRFs

Li *et al.* [13] use a combination of a convolutional neural network and Conditional Random Field to estimate depth on the pixel level. They also use the same architecture to estimate the surface normals. Convolutional neural network (Figure 2.4) estimates depth on the superpixel level, after that a *hierarchical CRF* is used to refine depth values to the pixel level. First, the input image is segmented into superpixels. Afterwards, for each superpixel, multiple patches centered around the superpixel are extracted from the input image, at different scales. In the final solution, patches of size 55x55, 121x121, 271x271, 407x407 are extracted and resized to a fixed size 227x227 pixels. These patches are each separately processed by a fixed, pre-trained convolutional neural network based on AlexNet [10] (all layers, except for the last fully connected layer are initialized using AlexNet). Outputs of

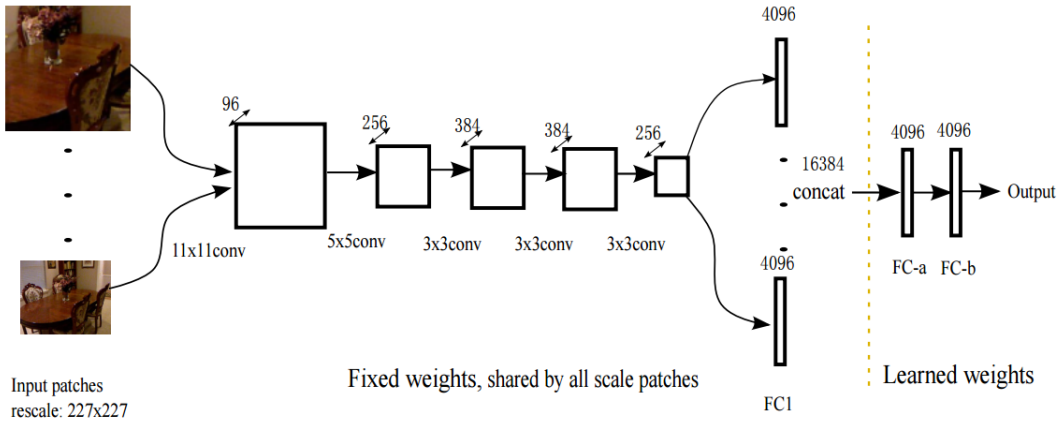


Figure 2.4: Convolutional neural network used to regress depth of superpixel from multi-scale patches [13]

the fully connected layer for each superpixel are then concatenated and fed to two fully connected layers with learned weights. Output of the network is the regressed depth value in the log space for the superpixel.

After depth of each superpixel is regressed, CRF is used to refine the estimate at the pixel level. Energy function for the CRF is specified as

$$E(d) = \sum_{i \in \mathcal{S}} \phi_i(d_i) + \sum_{(i,j) \in \mathcal{E}_s} \phi_{ij}(d_i, d_j) + \sum_{\mathcal{C} \in \mathcal{P}} \phi_{\mathcal{C}}(\mathbf{d}_{\mathcal{C}}). \tag{2.10}$$

$\mathcal{S}$ is the set of superpixels, $\mathcal{E}_s$ is the set of pairs of neighboring superpixels and $\mathcal{P}$ is the set of pixel level patches. d_i denotes the predicted depth value for the superpixel i and $\mathbf{d}_{\mathcal{C}}$ are predicted depth values for pixels in patch $\mathcal{C}$. First two terms capture constrains on predictions on the superpixel level, the third term on the pixel level. The first term measures a distance of predicted depth from the depth regressed by the convolutional neural network.

$$\phi_i(d_i) = (d_i - d_i^*)^2, \tag{2.11}$$

where d_i^* is the regressed depth. The second term is the pairwise potential and it serves a similar purpose as the pairwise potential in [15]:

$$\phi_{ij}(d_i, d_j) = w_1 (\frac{d_i - d_j}{\lambda_{ij}})^2. \tag{2.12}$$

λ_{ij} is the color difference between connected superpixels in the CIELUV color space [6] and it is used to weight the difference between depths of superpixels, enforcing that superpixels with similar colors have a similar depth values. w_1 specifies the weight of the term in the energy function. The third term is an auto-regression model, which is based on the assumption that the depth value of the pixel can be estimated from depth values of its neighbors. The auto-regression term is

$$\phi_{\mathcal{C}}(\mathbf{d}_{\mathcal{C}}) = w_2 * (d_u - \sum_{r \in \mathcal{N}} \alpha_{ru} d_r)^2, \quad \forall d_u \in \mathbf{d}_{\mathcal{C}}, \tag{2.13}$$

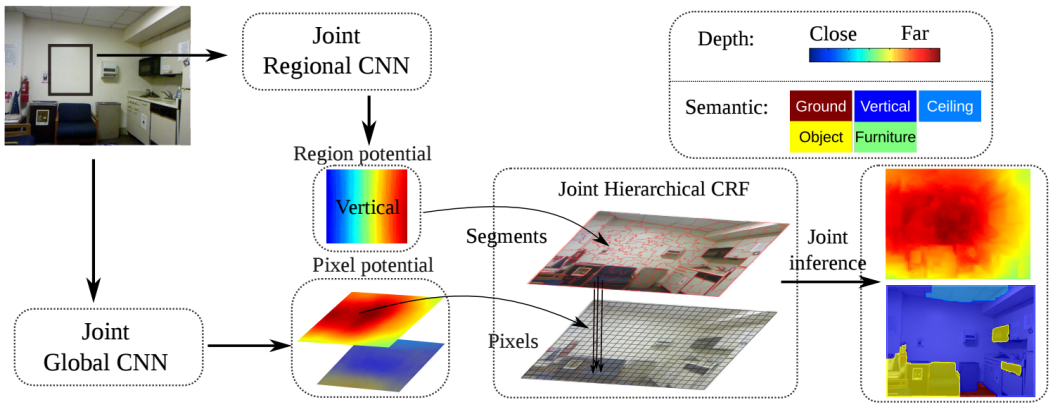


Figure 2.5: Architecture for joint inference of semantic labels and depths using CNN and CRF. [25]

where $\mathcal{N}$ is a set of pixels in a patch $\mathcal{C}$ and α_{ru} is a coefficient for an auto-regression model for the pixel r in the neighborhood of pixel u. α_{ru} is set to $\exp(-\frac{(g_u-g_r)^2}{2\sigma_u^2})$ with the condition $\sum_{r\in\mathcal{N}} \alpha_{ru} = 1$. g_i is the intensity of the pixel i and σ_u is the variance of intensities in the local patch around u. Parameter w_2 specifies the weight of the term in the energy function and along with parameter w_1, is specified by hand.

The model was trained and evaluated on NYU Depth v2 [18] and Make3D [20] datasets. CRF parameters were estimated by cross-validation, networks were trained using Euclidean loss. Results are compared to results of Eigen *et al.* [5] and achieve better results in five from six metrics. However compared to Liu *et al.* [15] they achieve worse performance in all metrics. Qualitatively, the resulting depth maps contain sharper edges and better fit local structure than the results of [5], which can be attributed to the pairwise potential. Li *et al.* suggest that using multi-scale patches improves performance and it is critical to use sufficiently large pixels, emphasizing the importance of large-scale context in depth estimation. Similarly to Liu *et al.* [15], they don't consider the issue of scale invariance.

Towards Unified Depth and Semantic Prediction From a Single Image

Wang *et al.* [25] combine approaches used by Liu *et al.* [15] and Eigen *et al.* [5] and use a global context CNN for estimating the scene layout, regional CNN for estimating the local depth structure and the results of these networks are combined in a *hierarchical CRF* (Figure 2.5). Additionally, they try to improve the performance by jointly training the network for the task of depth estimation and semantic labeling.

Global context network has the same structure as the one in [5], with the exception of accommodating the last layer for semantic label prediction. The output of this network is a depth estimate in log space and a semantic label for each pixel. The loss for this networks is a sum of the Euclidean loss for the depth estimation part and the Softmax loss for the semantic prediction part.

Regional CNN estimates depth and semantic label for each region of the image separately. These regions are extracted from the image using over-segmentation [17]. Regional CNN has the same architecture as the global CNN, in fact it is just a fine-tuned version of it. The output of the network is a semantic label l_s for the segment s and affinities to local

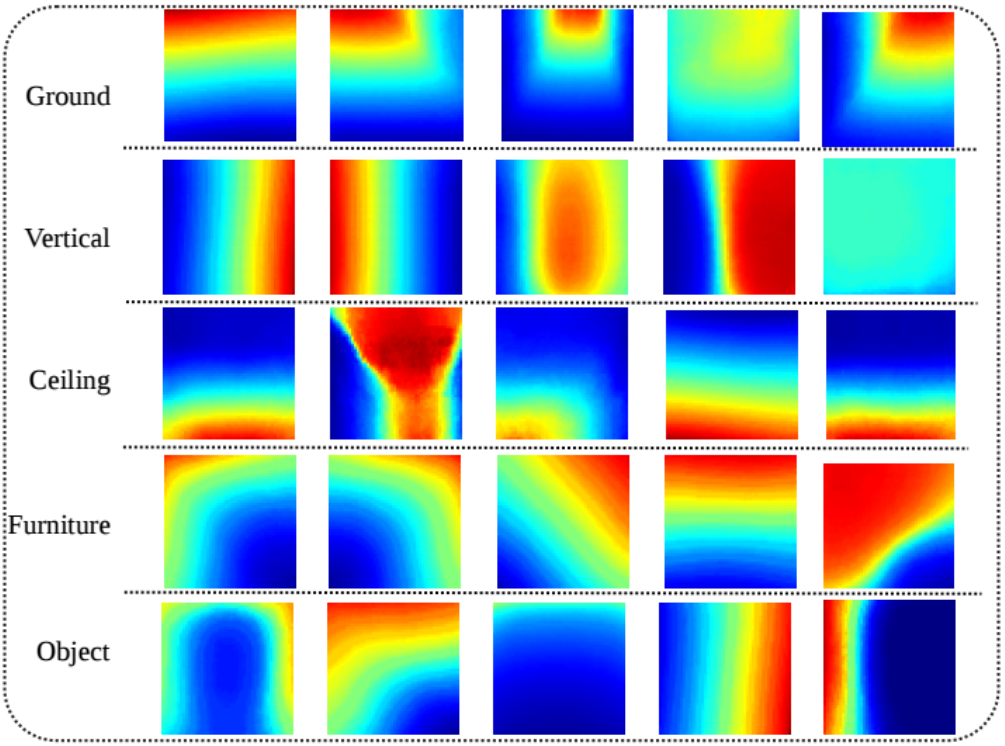


Figure 2.6: Examples of local region depth templates used in [25]

depth templates. These templates represent typical local structures in the depth map, such as corners, edges, planes, etc. and are pre-generated using ground truth. See Figure 2.6 for examples. Output of the network is thus an affinity to each template from the set of templates T and is defined during training as:

$$a(s, T_j) = \mathbb{1}_{\{l_{T_j}\}}(l_s) \frac{\exp(-\|\mathbf{d}_s - \mathbf{d}_{T_j}\|)}{\max_k \exp(-\|\mathbf{d}_s - \mathbf{d}_{T_k}\|)}, \tag{2.14}$$

where $\mathbb{1}$ is the indicator function enforcing that the affinity is zero if the predicted label and the template label are not the same, $\mathbf{d_s}$ are depths of pixels inside the segment and $\mathbf{d_{T_j}}$ are depths of pixels inside the template T_j. The loss for this network is a sum of cross-entropy loss over affinities. Since the absolute depth cannot be estimated only from the local segment, relative depth values are predicted. Conversion from absolute to relative depths for the ground truth targets is done by subtracting the absolute value of the pixel at the center of the segment and rescaling to range $< 0, 1 >$. Relative values can be converted to absolute values by multiplication by scale s_i and adding mean d_i. s_i and d_i are inferred by CRF.

Hierarchical CRF is used to infer depth and semantic label for each pixel from the outputs of the global and the regional CNN. The energy function E being minimized by CRF is

$$E = \sum_{i \in \mathcal{I}} \psi_{pu}(d_i, l_i) + \lambda_{pe} \sum_{i,j \in \mathcal{I}} \psi_{pe}(d_i, l_i, d_j, l_j) + \lambda_y (\sum_{s \in \mathcal{S}} \psi_c(\chi_s, \mathbf{y}_s) + \lambda_{se} \sum_{s,t \in \mathcal{S}} \psi_{se}(\mathbf{y}_s, \mathbf{y}_t)), \tag{2.15}$$

where ψ_{pu} is the unary potential on the pixel level, ψ_{pe} is the pairwise potential on the pixel level, ψ_c is the cross level potential that relates predictions $\mathbf{y}_s$ for segment s to pixels from the global context that are inside this segment χ_s. ψ_{se} is a pairwise potential on the segment level. λ_y is a balancing parameter specifying the weight of the segment level potentials and is estimated using maximum likelihood on ground truth. λ_{pe} and λ_{se} are balancing weights for the pairwise potentials and are learned through cross validation. Explaining in detail all the potentials is beyond the scope of this work.

Model is trained on NYU Depth v2 [18]. Since data for the task of semantic labeling are limited compared to data for depth estimation, the model is first trained to predict only the depth values on the full raw NYU Depth v2 dataset and after that the model is fine-tuned for semantic labeling on the labeled part of the dataset.

Quantitative results show better performance than [5] except on the threshold metric. Performance is better than [15] on the absolute relative difference and the RMSE metrics. For the global network used for this model, predicting jointly semantic labels and depth values show increase in quantitative performance compared to predicting only depth values. Enforcing both local and global consistency by using CRF shows slight quantitative improvement from using only the global network, but qualitatively shows visible difference, with edges of the predicted depth map being much sharper and better aligned to local details. Wang *et al.* speculate that the worse performance on threshold metric than [5] is due to not considering scale invariance.

Unified Depth Prediction and Intrinsic Image Decomposition from a Single Image via Joint Convolutional Neural Fields

Two scene properties which are important for understanding the structure of the scene are 3D geometry and a lighting. These two are of course not independent, since geometric structure determines the illumination and shading of the scene and lighting conveys information about scene's 3D structure. Kim *et al.* [9] aims to improve the task of predicting depth and image intrinsic (albedo and shading) by jointly predicting both. They introduce **Joint Convolutional Neural Field (JCNF)** which couples convolutional neural networks with Conditional Random Fields to predict depth and image intrinsics both. Proposed architecture (Figure 2.7) contains four important parts - Depth prediction network, Intrinsic prediction network, Gradient scale network and Joint Conditional Random Field. Additionally, they perform learning in gradient domain, where there are stronger correlations between depth and intrinsic images.

Depth prediction network consist of global depth network and depth gradient network. Global depth network takes an input image and its output is an estimate of overall scene's depth map. It is a convolutional network based on AlexNet [10] and with pre-trained weights. It contains five convolutional and two fully connected layers. The depth gradient network is a fully convolutional network, consisting of five convolutional layers, first one being identical to the first layer of AlexNet. Its input is an input image and the output of the first convolutional layer is concatenated with the output of the global depth network, thus considering global scene context in gradient prediction. Furthermore, output of the second convolutional layer of the intrinsic prediction network is concatenated with the output of the second convolutional layer of depth gradient network.

Intrinsic prediction network has a similar structure to the depth gradient network. Its input is an input image and it outputs the albedo gradient and shading gradient for the input image. Since both task rely on similar properties, first three convolutional layers are

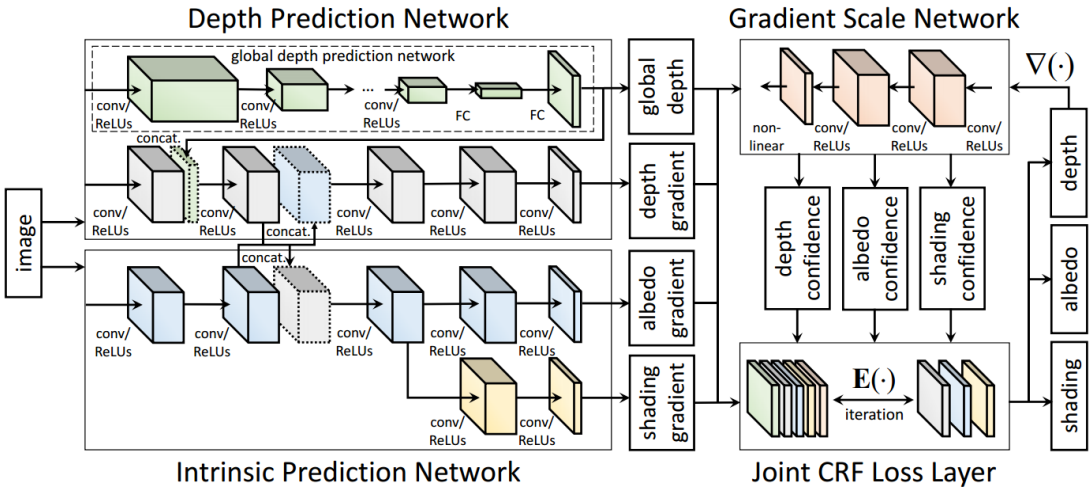


Figure 2.7: Joint Convolutional Neural Field [9]

shared for both tasks and only last two are separate. To take advantage of jointly infering depth and intrinsic gradient, output of the second convolutional layer is concatenated with the output of second convolutional layer of the depth gradient network.

To help reduce the influence of potential errors in depth and intrinsic gradient predictions, *gradient scale network* is used. Its inputs are the predicted gradients for depths, albedo and shading and its output is a confidence in each estimated gradient. During training it finds consistencies between different types of gradients and uses this information to estimate the confidence in a gradient based on input RGB image gradient and other two predicted gradients (e.g., for the depth gradient, confidence is estimated based on an input image, albedo and shading gradients). It contains three convolutional layers and its output is an image with the same resolution as predicted gradients, with confidence weights as values for each pixel.

Previously introduced networks are learned jointly by minimizing the energy function E of the *Joint Conditional Random Field*.

$$\begin{aligned}\mathbf{E}(D,A,S) = \mathbf{E}_u(D) + \mathbf{E}_u(A,S) \\ + \lambda_D \mathbf{E}_s(D|A,S) + \lambda_A \mathbf{E}_s(A|D,S) + \lambda_S \mathbf{E}_s(S|D,A),\end{aligned} \tag{2.16}$$

where D, A, S are the predicted depth, albedo and shading map respectively. I is the input image, $\mathbf{E}_u$ are unary potentials, $\mathbf{E}_s$ are pairwise potentials and $\lambda_D, \lambda_A, \lambda_S$ are weights for the pairwise potentials. The unary potential for depth $\mathbf{E}_u(D|I)$ is defined as

$$\mathbf{E}_u(D) = \sum_p (D_p - F_{dp})^2, \tag{2.17}$$

where D_p is the predicted depth at pixel p and F_{dp} is the depth estimated by global depth network for that pixel. Unary potential for intrinsical images $\mathbf{E}_u(A,S)$ is specified as

$$\mathbf{E}_u(A,S) = \sum_p (L_p(I_p - A_p - S_p))^2, \tag{2.18}$$

where $L_p(I)$ is a luminance of I, A_p, S_p are the values of pixel p of predicted albedo and shading maps respectively. The unary potential is based on the image formation equation

$I = A + S$. Note that this term does not depend on any parameters learned by any of the networks.

Pairwise potential enforces the similarity between gradients predicted by CRF and the gradients estimated by the depth gradient network and intrinsic prediction networks, weighted by confidence in estimations:

$$\mathbf{E}_s(D|A,S) = \sum_p \|\nabla D_p - \mathcal{G}_p(\nabla I_p, \nabla A_p, \nabla S_p) * G_{dp}\|^2, \tag{2.19}$$

where $\nabla I_p, \nabla D_p, \nabla A_p, \nabla S_p$ are the gradients of the image and predicted depth, albedo and shadow map respectively at the pixel p by CRF. G_{dp} is the depth gradient estimated by the depth gradient network for the pixel p and $\mathcal{G}_p$ is the confidence in the estimated gradient at pixel p. Potentials $\mathbf{E}_s(A|D,S)$ and $\mathbf{E}_s(S|D,A)$ are defined in the same manner.

Model is trained on NYU Depth v2 [18] and Make3D [20] datasets. First, the depth prediction network and the intrinsic prediction networks are trained with the gradient scale network fixed. Global depth network is trained using Euclidean loss, gradient depth, albedo and shading networks are trained using Euclidean loss defined identically to corresponding pairwise potentials. After training the depth and the intrinsic prediction networks, their weights are fixed and the gradient scale network is trained using Euclidean loss similar to the gradient prediction networks. This process is repeated until convergence. Kim *et al.* use only the labeled part of the NYU dataset for training, but used data augmentation to decrease overfitting. They achieve better performance on all metrics than [5] and [15], but worse performance than [4]. Qualitatively, as it is with other approaches using Conditional Random Fields, the estimated depth maps contain sharp transitions and are well aligned to local structures.

Kim *et al.* embed a prior knowledge into the system by considering global scene context using global depth network and local structure using gradients instead of the absolute value images. Additionally they improve the prediction by jointly training their model on image intrinsics prediction as they show in their results. However, they do not deal with the scale invariance.

2.4.3 Convolutional Neural Networks in Decision Trees

Monocular Depth Estimation Using Neural Regression Forest

Roy *et al.* [19] propose a decision tree based model that utilizes convolutional neural networks. **Neural Regression Forrest** (Figure 2.8) is a combination of convolutional neural networks and random regression forest with binary regression trees. When predicting the depth for a pixel p, image window with the pixel p at the center is processed by a convolutional neural network at the root tree node. Features from the last convolutional layers of the network at the node are used as an input to the child nodes. Since used CNNs contain pooling layers, they decrease the resolution of features as they progress down the tree. This results in multi-scale view of the input (nodes at different depth of the tree see different scale). The same process is repeated for each split node. CNN at each node also contains fully connected layers that output a probability of passing the output to the left or right child node. Note that features are passed to both child nodes, but result predicted by taking each path is weighted by the probability of taking that path. Leaf nodes don't contain a CNN, rather they contain parameters of a Gaussian distribution (mean μ and standard deviation σ). Probability distribution over depths estimated by tree τ is defined as

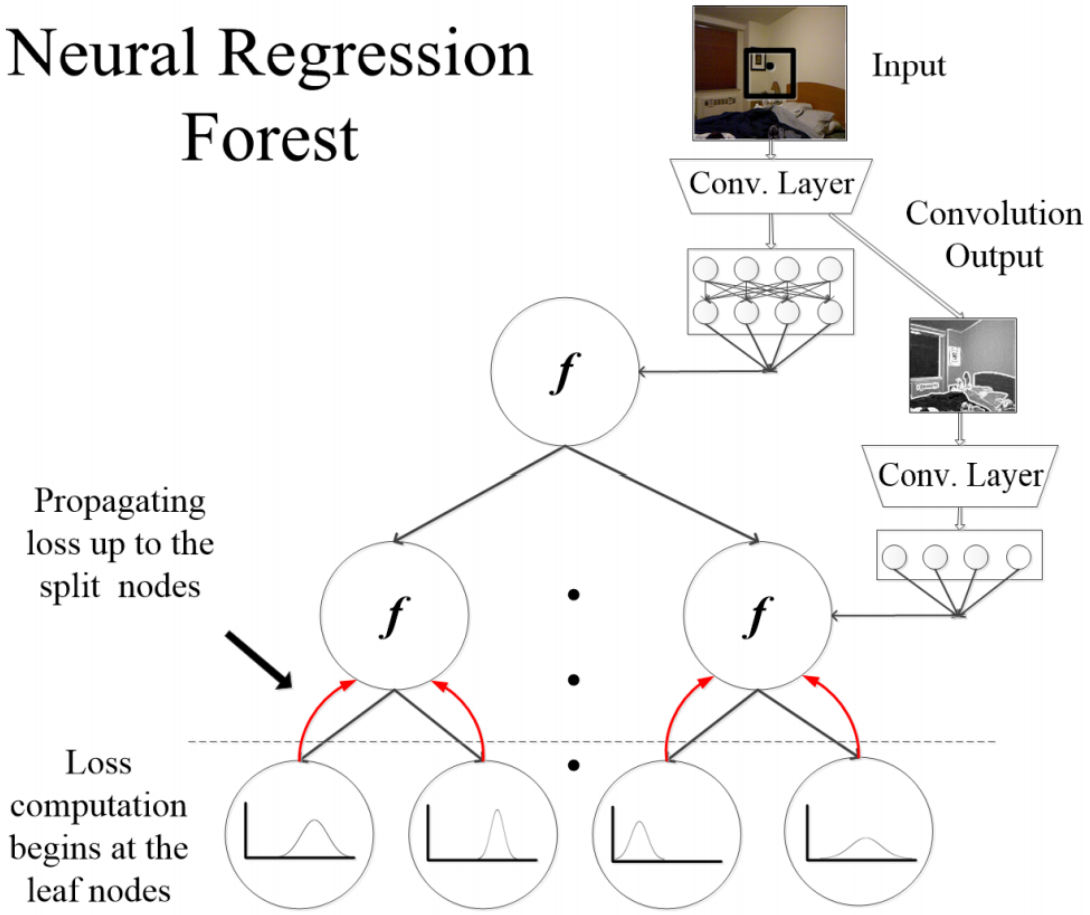


Figure 2.8: Neural Regression Forest [19]

$$p_\tau(d|x) = \sum_{l \in L_\tau} p(d; \theta_l) P(l|x; \mathcal{W}_\tau), \tag{2.20}$$

where L_τ is the set of all leaf nodes, θ_l are the parameters of the Gaussian distribution for the leaf node l and $\mathcal{W}_\tau$ are the parameters of all CNNs in the forest. $p(d; \theta_l)$ is the probability distribution over depths given parameters θ in the leaf node l. $P(l; \mathcal{W}_\tau)$ is a probability of reaching the leaf node l from the root and is computed as a product of probabilities of taking the path to the leaf node computed at each split node by its CNN. Additionally, to enforce smoothness in depth predictions, the probability distribution predicted by the tree for pixel p is modified to consider probability distributions predicted by the tree for pixels in a neighborhood of p by bilateral filtering:

$$\overline{p}_\tau(d|x_i) = \sum_{j \in N(i)} \kappa_{ij} p_\tau(d|x_j). \tag{2.21}$$

$N(i)$ is the neighborhood of pixel i and κ_{ij} is the weight of bilateral filter that is estimated based on Euclidean distance between locations of pixels i and j and Euclidean distance of the HSV histograms of windows with pixels i and j at the center. Bilateral filtering in this case means that only neighboring windows with similar appearance have an

	rel	sqr rel	rms	rms-log	log10	$\delta <$ 1.25	$\delta <$ 1.25^2	$\delta <$ 1.25^3
[5]	0.215	0.212	0.907	0.285	-	0.611	0.887	0.971
[4]	**0.158**	**0.121**	**0.641**	0.214	-	**0.769**	**0.950**	**0.988**
[15]	0.213	-	0.759	-	0.087	0.650	0.906	0.976
[13]	0.232	-	0.821	-	0.094	0.621	0.886	0.968
[25]	0.220	0.210	0.745	0.262	0.094	0.605	0.890	0.970
[9]	0.201	-	0.711	**0.212**	**0.077**	0.690	0.910	0.979
[19]	0.187	-	0.744	-	0.078	-	-	-

Table 2.2: Comparison of existing solutions (best in each metric in bold)

influence over depth prediction. The overall probability distribution over depths predicted by the forest is specified as

$$p_{\mathcal{F}}(d|x) = \frac{1}{|\mathcal{F}|} \sum_{\tau \in \mathcal{F}} p_{\tau}(d|x), \tag{2.22}$$

where $\mathcal{F}$ is the set of all trees. Convolutional neural networks used at individual split nodes have a different number of layers depending on the vertical position in the tree. Top third of nodes from the root have 2 convolutional and 2 fully connected layers, middle third of nodes have 2 convolutional and 1 fully connected layer and the last third of nodes contains only 1 convolutional and 1 fully connected layer.

For training, negative log-likelihood function L is used to regress parameters $\mathcal{W}$ for all CNNs on all trees and parameters for all the leaf nodes Θ.

$$L(\mathcal{W}, \Theta; x, d) = -\log p_{\mathcal{F}}(d|x) \tag{2.23}$$

Training process starts with fixing $\mathcal{W}$ and optimizing for Θ. In the second step, Θ is fixed and $\mathcal{W}$ is optimized. These steps are alternated until convergence. During experiments, forest contains 100 trees, each with a depth 10.

Model is trained on NYU Depth v2 [18] and Make3D [20] datasets. For the NYU Depth v2 dataset, model is trained only on the labeled part of the dataset. Quantitative measurements show better performance than [5] and [15], but Neural Regression Forest performs worse than [4]. Metrics also show that enforcing smoothness of depth values over neighboring pixels with similar appearance has a positive impact on the performance. Roy *et al.* do not utilize global context of the scene or consider scale invariance.

Chapter 3

Proposed Model for Depth Estimation

This chapter describes a model I propose for the task of depth estimation from a single image. I follow the work done by Eigen *et al.* [5] and use a solution utilizing only convolutional neural networks. Similarly to Eigen *et al.*, I utilize global context of the scene and improve training using a loss function that is scale invariant. In addition, I explicitly incorporate information about gradients of the depth map to the process of depth estimation.

Input to the model is an RGB image, its output is a corresponding depth map estimated by the model. Proposed model consists of three parts: **global context network** that estimates the rough global depth map from an input RGB image, **gradient network** which estimates horizontal and vertical gradients of the global depth map from an input RGB image and **refining network** that uses estimates from previous networks, along with an input RGB image and refines depth map locally to produce a more detailed depth map. Figure 3.1 shows a general architecture of the model. Additionally, all parts are trained using normalized loss function to accomplish scale and translation invariance.

In the first section I introduce a normalized loss function that is scale and translation invariant and compare it to the loss function used in [5]. In the remaining sections I describe in detail architectures of the global context network, gradient network, their joint architecture and the architecture of the refining network.

3.1 Normalized Loss Function

As explained in Section 2.1.4, it is suitable to consider scale invariance when training model for depth estimation. Inspired by Eigen *et al.* [5], I propose a modified loss function which considers scale invariance.

Scale invariant loss function used by Eigen *et al.* is defined as

$$L_{sc-inv}(y, y^*) = \frac{1}{2n}\sum_i(\log y_i - \log y_i^* + \frac{1}{n}\sum_j(\log y_j^* - \log y_j))^2, \tag{3.1}$$

which amounts to subtracting mean value from estimated depth map y and ground truth y^* in log space. In linear space this means dividing both the ground truth and the estimated depth map by their respective maximum values. The loss stays the same for every scalar multiple of estimated depth y. This loss function is valid in a sense that it produces the

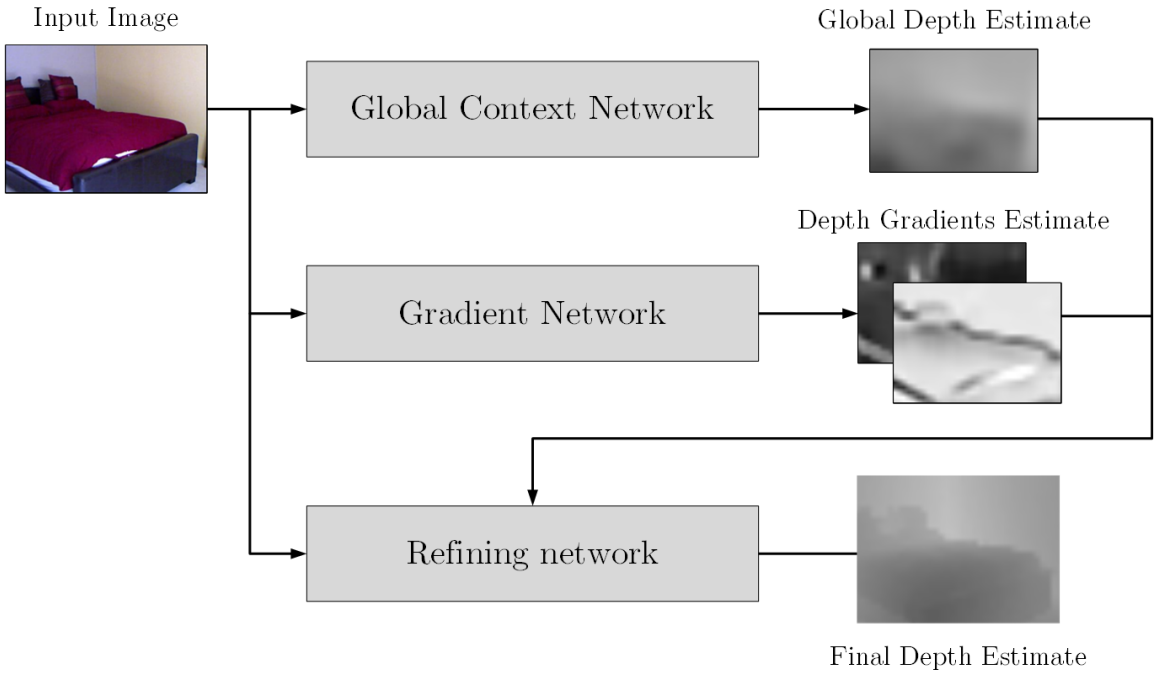


Figure 3.1: Architecture of the proposed model

same loss only in those cases where the input RGB image is the same, but underlying scene scale is different.

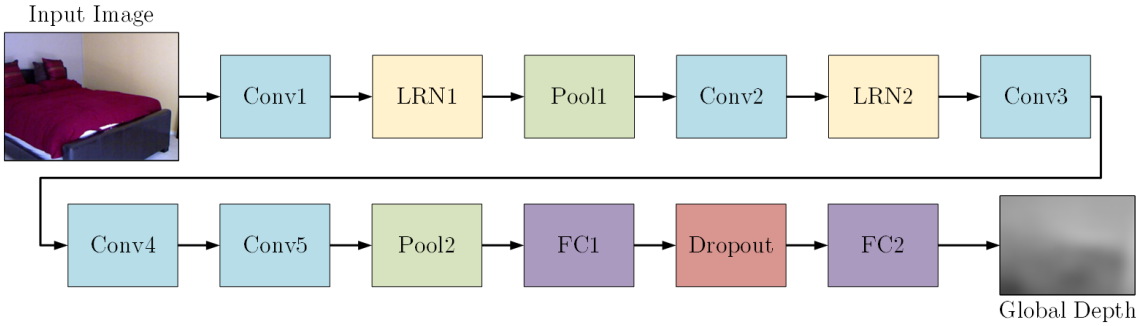
In this work I try to show that if we relax requirements for the loss function and allow it to produce the same loss even when the input RGB image is not identical but depicts the same real world structure viewed from different distance, we gain better estimates of the scene's depth structure. To achieve this I propose normalized loss function L:

$$L(y, y^*) = \frac{1}{N} \sum_i \left(\frac{y_i - y_m}{\sqrt{y_v}} - \frac{y_i^* - y_m^*}{\sqrt{y_v^*}}\right)^2, \tag{3.2}$$

where y is the estimated depth map, y^* is the ground truth, y_m and y_m^* denotes mean values of respective depth maps and y_v and y_v^* denotes respective variances of these depth maps. L computes difference between mean-variance normalized output depth and the ground truth, meaning that for every s, t, y, the value $L(y^s, y^*)$ for $y^s = s * y + t$ is the same as $L(y, y^*)$. This achieves scale and translation invariance.

Problem with using this function is that estimated depth values y are not required to be in any absolute range and therefore it is difficult to compare output of neural network using loss L with the ground truth in terms of absolute values. To be able to reasonably measure performance of models using loss L, I propose normalized error function $E(y, y^*)$ that is defined identically to L. This error function evaluates how well the model predicts the structure of the scene irrespective of scale and translation.

In case there is no need to obtain accurate absolute values, mean-variance normalized output of the network can be used, as is the case for the global context network. If absolute values are required to be correct, it is possible to train the network using modified loss



	conv1	pool1	conv2	conv3	conv4	conv5	pool2	fc1	fc2
size	11x11	3x3	5x5	3x3	3x3	3x3	3x3	-	-
channels	96	-	256	384	384	256	-	1024	999
stride	4	2	1	1	1	1	2	-	-
pad	0	0	2	1	1	1	0	-	-
resolution	72x52	36x26	36x26	36x26	36x26	36x26	18x13	-	-
rel. lr	0.02	-	0.02	0.02	0.02	0.02	-	1	0.2

Figure 3.2: Architecture of the global context network

$$L_a(y, y^*) = L(y, y^*) + \lambda \frac{1}{N} \sum_i (y_i - y_i^*)^2, \tag{3.3}$$

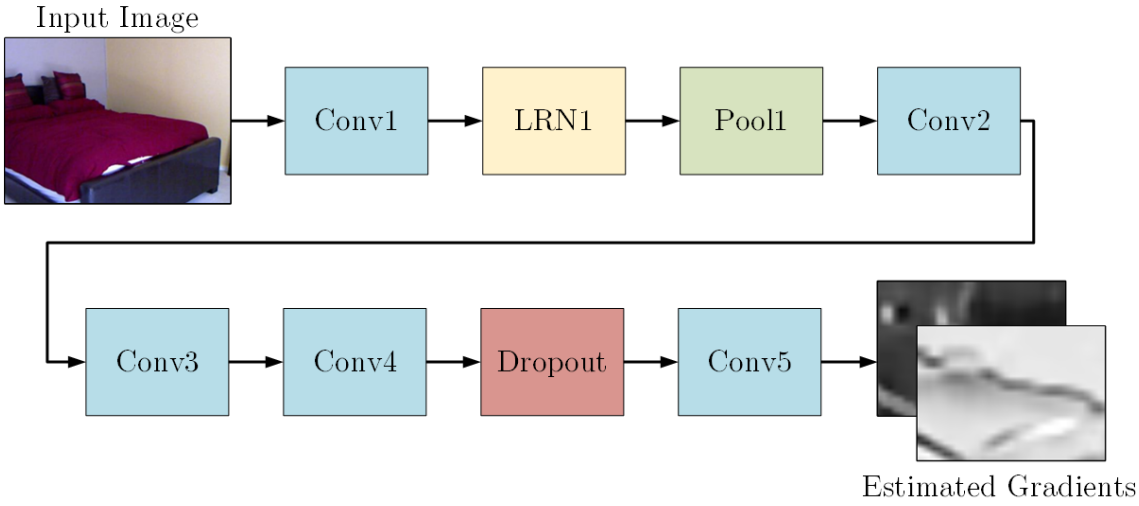
which additionally minimizes the difference between absolute values. Influence of the absolute value term can be fine-tuned using parameter λ. Another possible approach is to jointly estimate the normalized depth map, corresponding mean m and variance v. Mean and variance obtained in this way can then be utilized to produce the final estimate $y_f = y * \sqrt{(v)} + m$.

In Chapter 5, I compare the performance of a model using the normalized loss function with that of a model using Euclidean distance loss and a model using scale-invariant loss function proposed by Eigen *et al.*.

3.2 Global Context Network

As explained in Section 2.1.2, depth estimation can benefit from having a knowledge of the global structure of the scene depicted on the input image. To take advantage of having this knowledge I take an approach similar to Eigen *et al.* [5] and my model contains **global context network**. Global context network estimates the rough depth map of the whole scene from the input RGB image. It takes advantage of fully connected layers that have a full field of view to estimate the scene's global context. Input to this network is an RGB image with a resolution of 298x218 pixels. Its output is a depth map with a resolution of 37x27. Raw input has a range of values [0,255], but before being fed to the network, value 127 is subtracted from the input. Raw ground truth depth maps also have a range [0,255], but are scaled to be in range [0,1]. During training, normalized loss function was used.

Network structure is derived from AlexNet [10]. It contains five convolutional layers with Rectified Linear Units (ReLUs) used for activation units. First two layers convolutional layers are followed by Local Response Normalization (LRN) layer which divides input values



	conv1	pool1	conv2	conv3	conv4	conv5
size	11x11	4x4	5x5	5x5	5x5	5x5
channels	96	-	256	384	384	256
stride	4	2	1	1	1	1
pad	0	0	2	2	2	2
resolution	72x52	35x25	35x25	35x25	35x25	35x25
rel. lr	0.001	-	1	1	1	0.1

Figure 3.3: Architecture of the gradient network

by $(1+(\alpha/n\sum_i x_i^2)^\beta)$, where n is the size of the local region that the sum is being evaluated over. These local regions extend across channels. In this and both gradient and refining networks, parameters for LRN layers are $\alpha = 0.0001$, $\beta = 0.75$ and $n = 5$. Max pooling layers are placed after the first and the last convolutional layer. Original AlexNet model contains one more max pooling layer placed after the second convolutional layer, but I removed it, to obtain feature maps with higher resolution from the last convolutional layer. These convolutional layers have weights initialized by Imagenet [3] pre-trained AlexNet and have a relative learning rate set to 0.02.

Convolutional layers are followed by two fully connected layers. The first layer contains 1024 neurons and has a ReLU activation unit. It is initialized using Xavier initializer in Caffe [8] and has a relative learning rate 1. To increase network's capability to generalize, first fully connected layer is followed by a dropout unit with a dropout ratio 0.5. The second fully connected layer is an output layer and it contains 999 neurons, which is the same as the pixel count of the output depth map. Output layer is initialized using Gaussian noise with a mean of 0.5 and a standard deviation of 0.001. Its relative learning rate is 0.2. See Figure 3.2 for detailed structure of the global context network with numbers of channels and kernel sizes of convolutional layers.

3.3 Gradient Network

Proposed model incorporates explicitly information about significant changes of depth in the scene. This is accomplished by using the **gradient network** that estimates horizontal

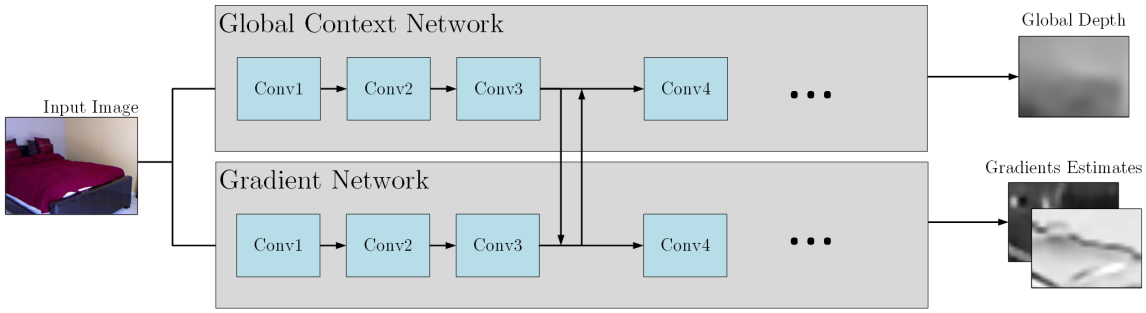


Figure 3.4: Architecture A for a joint training of the global context network and the gradient network

and vertical gradients of the depth map globally, for the whole RGB image. Input for the gradient network is the same as for the global context network - an RGB image with a resolution of 298x218. Gradient network outputs two depth map gradients, one horizontal and one vertical, both with resolution of 35x25. Similarly to global context network, normalized loss was used for training, input is converted into range [-127,128] and ground truth is scaled to range [0,1].

Gradient network is a fully convolutional network with five convolutional layers that are based on the convolutional part of AlexNet with a few differences. Max pooling and LRN layers after the second convolutional layer are removed and convolutional layers 3,4 and 5 have an increased kernel size 5x5. To prevent overfitting, dropout unit is placed between last two layers with a dropout ratio 0.5. First convolutional layer is initialized using AlexNet model pre-trained on ImageNet with a relative learning rate 0.001. Convolutional layers 2, 3, 4 are initialized using Xavier initializer and have a relative learning rate 1. Last convolutional layer is the output layer and it is initialized using Gaussian noise with a mean of 0.0 and a standard deviation of 0.1. Its relative learning rate is 0.1. Since randomly initialized output of the network returns on average estimates that are two orders higher than values of the ground truth, the output of the network is multiplied by 0.01. This prevents exploding gradients at the beginning of the training. See Figure 3.3 for more detail on the network structure.

3.4 Joint Global Context Network and Gradient Network Architecture

Training network for multiple tasks can improve performance for all tasks [2] and since estimating depth map and corresponding gradients of the depth map are related tasks, I tried to jointly train the global context network and gradient network. I tried two different architectures.

In the architecture A (see Figure 3.4), global context network and gradient network stay structurally the same, but the outputs of third convolutional layers from both networks are concatenated and after that passed to fourth convolutional layers of each network. This accomplishes transfer of information between networks. There is a slight difference in the Global context network - since fourth convolutional layer has more channels on its input, it cannot be easily initialized with weights from pre-trained AlexNet model. Xavier initializer is used for this layer.

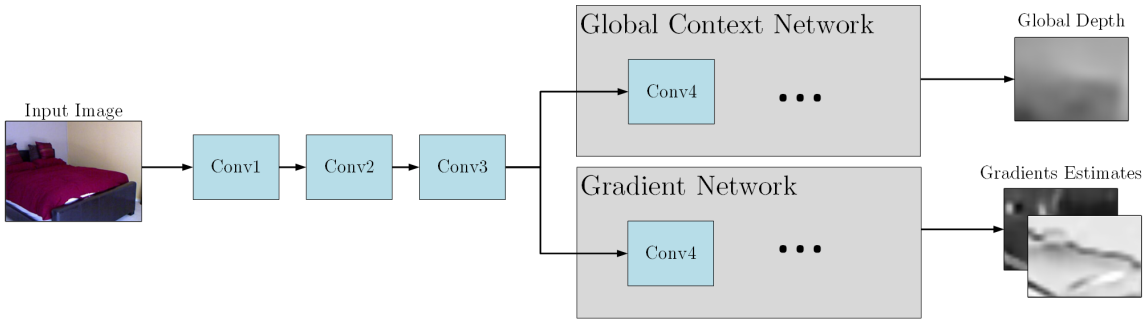


Figure 3.5: Architecture B for a joint training of the global context network and the gradient network

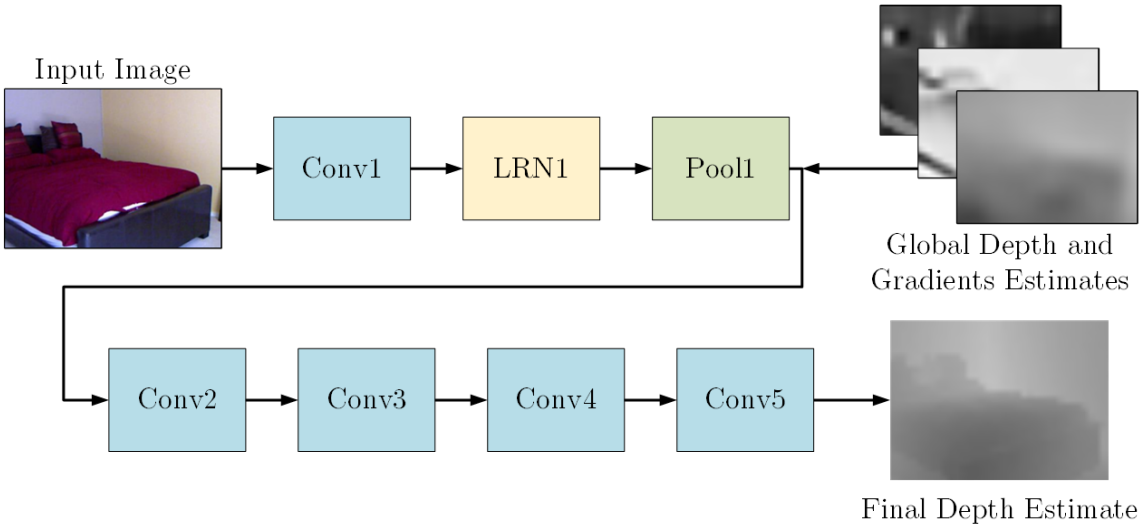
In the second architecture B (see Figure 3.5), I join first three convolutional layer from both networks. The network is then split and the output of the third convolutional layer is passed to remaining parts of global context network and gradient network that are structured as described in Section 3.2 and Section 3.3. First three convolutional layers are initialized in the same way as in the global context network, using pre-trained AlexNet model.

When trained separately, the global context network uses learning rate twice as high as learning rate for gradient network. This is reflected in the joint model by dividing relative learning rates of all layers that are part of the gradient network by half.

3.5 Refining Network

Since Global context network provides only a rough estimate of the depth map, I use **refining network**. Refining network improves the rough estimate from the global context network, utilizing gradients estimated by the gradient network and an input RGB image. Input to the network is thus an RGB image with a resolution of 298x218 and mean-variance normalized outputs of the global context network and the gradient network, upsampled using bilinear sampling to 74x54 resolution. Output is a depth map with a resolution of 74x54. As used in both previous networks, input is in range [-127,128] and ground truth is in range [0,1]. Loss function is L_a defined in Equation 3.3 with $\lambda = 1$.

Refining network is a fully convolutional network and similarly to the other parts of the model, it is also based on AlexNet and thus contains five convolutional layers. First convolutional layer processes an input RGB image, uses ReLU as an activation unit and is followed by LRN layer and max pooling layer that outputs features maps with a resolution 74x54. These feature maps are concatenated with outputs of the global context network and the gradient network and are fed to remaining four convolutional layers. All layers use ReLU as an activation unit and the second layer is followed by max pooling which does not downsample feature maps. All layers except the last, output layer are initialized using Xavier initializer. The last layer is initialized using Gaussian noise with a mean of 0.5 and a standard deviation of 0.01. The first and the last layers have a relative learning rate 0.1, layers 2-4 have a relative learning rate set to 1. See Figure 3.6 for more detail.



	conv1	pool1	conv2	conv3	conv4	conv5
size	11x11	2x2	5x5	5x5	5x5	3x3
channels	96	-	64	64	64	64
stride	2	2	1	1	1	1
pad	2	1	2	2	2	2
resolution	146x106	74x54	74x54	74x54	74x54	74x54
rel. lr	0.001	-	1	1	1	0.01

Figure 3.6: Architecture of the refining network

Chapter 4

Implementation of the Proposed Model

In this chapter, I describe the implementation details of the model introduced in Chapter 3. First section briefly describes Caffe framework. Second section covers NYU Depth v2 dataset that was used for training and testing. In this section I also describe how I processed data and how I used data augmentation to increase the size of the dataset. In the next section, I describe a training procedure I used and the last section covers how I implemented various loss functions that were used in experiments.

4.1 Caffe

I used Caffe [8] framework for training and testing convolutional neural networks. Caffe is an open source deep learning framework developed by Berkeley Vision and Learning Center. It is written in C++, with CUDA used for GPU acceleration and contains bindings to Python and MATLAB. Commonly used network models are available to use with Caffe with pre-trained weights and network definition files.

Caffe uses *prototxt* (plaintext protocol buffer schema) file format to define network and hyper-parameters used for training. Networks are defined by specifying network layers and their parameters in the network definition file. Caffe contains most of commonly used layers and custom layers can be implemented and added without difficulty. Images are provided to Caffe from either HDF5 files, or LMDB/LevelDB databases.

Caffe also provides various tools to help with setting up a network for training e.g., a script for converting images to LMDB/LevelDB databases and a script to visualize network from a network definition file.

4.2 Dataset

As presented in Section 2.3, there are multiple datasets used for training and evaluating performance of systems performing depth estimation. For this thesis, I chose to use NYU Depth v2 [18] dataset.

I used the component of dataset that contains raw Kinect data and raw RGB output from camera, both in 640x480 resolution, as my training set. I used toolbox provided with the dataset containing MATLAB scripts to prepare data for usage. I first selected only image from scenes that are part of the training set. Then, for each depth map, RGB image

that was captured at the similar time moment had to be selected. Since data comes from a video footage with 20-30 FPS, consequent frames are similar. For this reason I used only every fifth frame, to save time needed to process images. After obtaining the synchronized RGB images and depth maps, each depth map was projected onto a corresponding RGB image to obtain depth values in meters. Missing values from the depth map were filled in using Levin's *et al.* colorization method [12]. Finally, both the RGB image and the depth map were cropped to a rectangle in the depth map that contains projected depth signal. By this process I obtained 47575 pairs of RGB and depth images, with a resolution of 561x427. Resulting depth maps were stored as 8-bit images, where value 255 denotes distance 10 meters.

To increase the size and variability of the training set, I use data augmentation with following transformations:

- Scale: Input and target images are scaled by s from the interval i_{sc}, and depths are divided by s.
- Rotation: Input and target are rotated by θ degrees, where θ is sampled from a Gaussian distribution with a mean of 0 and a standard deviation σ.
- Translation: Random parts of the input and target images of size 420x320 were cropped. (Note that when rotation is used, the image is cropped at the center to prevent corners of rotated image containing area outside of the original image).
- Flips: Input and target are horizontally flipped. Both the flipped and original image are used.
- HSV shift: Hue, saturation and value of the input image is shifted by a random amounts from intervals i_h, i_s, i_v.
- Change of contrast: Values of the input image are scaled to be in range [0,c] where c is from interval i_c, middle value is subtracted and resulting values are cropped to the range [0,255].

Each original pair of images is augmented separately five times and flipped horizontally, so each pair is used to create 10 augmented pairs. I created three versions of dataset, *Data0, Data1, Data2*, with different transformation parameters, to test the effects of data augmentation and magnitude of transformations on performance. *Data0* represents original dataset with no transformations. Since NYU Depth v2 dataset contains only indoor scenes, general composition of scenes is very similar across dataset. Using only parts of images changes this composition on average, so to ensure that images in all version depict roughly the same compositions of the scene, parts of input and target images of size 420x320 are cropped from the center in *Data0* version. *Data1* contains images with mild transformations, *Data2* with greater transformations. Table 4.1 shows parameters for all versions of dataset. Comparison of the performance when training with each dataset is presented in Section 5.3.

For testing, I used 654 images from the labeled part of dataset. I did not augment the testing set, the only preprocessing I used was cropping 420x320 pixels large parts from the center to obtain images with similar composition as in the training set.

Finally, each dataset was converted into LMDB database and resized to fit the network input (298x218) or network output (37x27 for the global context network and the gradient network and 75x54 for the refining network).

	Data0	*Data1*	*Data2*
Scale (i_{sc})	[1,1]	[0.875,1.25]	[0.75, 1.25]
Rotation (σ)	0	2.5	5.0
Flips	No	Yes	Yes
Hue shift (i_h)	[0,0]	[-0.06, 0.06]	[-0.1, 0.1]
Saturation shift (i_s)	[0,0]	[-0.06, 0.06]	[-0.1, 0.1]
Value shift (i_i)	[0,0]	[-0.06, 0.06]	[-0.1, 0.1]
Contrast change (i_c)	[255,255]	[205,305]	[175,335]

Table 4.1: Parameters used for data augmentation used to obtain different versions of the dataset

4.3 Training

In this section, I describe the process of training proposed model. First, I train global context network and gradient network, either separately in parallel or jointly. After that, I fix the weights in these networks and train refining network.

I used Stochastic Gradient Descent (SGD) with a momentum for training, with a momentum weight 0.9. To reduce overfitting, I complemented dropout with a weight decay. Decay weight is set to 0.005. Learning rate was set to be as high as possible for each training. Base learning rate is different for each part of the model. During training of the global context network, base learning rate 0.0005 was used, for the gradient network, base learning rate 0.00025 was used and for training the refining network, I used base learning rage 0.000025. Base learning rate is fixed for the whole duration of the training. Batch size also differs by part of the model trained. It is 32 for the global context network and the gradient network and 16 for the refining network.

4.4 Implementation of Various Loss Functions

To compare performance of using normalized loss function with performance of other loss functions that are commonly used, I needed to train the model with different loss functions. In this section, I describe how I implemented them using Caffe.

For **absolute value loss**, the Euclidean loss layer provided by Caffe is used.

Absolute value loss in log space requires ground truth to be converted into log space. I used function $f(x) = 0.179581 * ln(x) + 1$ as presented in section 2.1.1. $f(x)$ is implemented by first applying Log layer in Caffe to the ground truth. After that, Power layer is applied with scale 0.179581 and shift 1.0.

To implement **scale-invariant loss function** from [5], I first use the same conversion of ground truth to the log space as described in paragraph above. After that Caffe's MVN layer is used with variance normalization parameter set to false to process both the output of the network and the ground truth in the log space.

Normalized loss function is implemented by running both the ground truth in linear space and output of the network through Caffe's MVN layer with the variance normalization turned on. Loss function used for refining net is implemented as two loss layers, one is normalized loss and the other Euclidean distance in the linear space, each with a weight 0.5.

-1	0	1
-1	0	1
-1	0	1

-1	-1	-1
0	0	0
1	1	1

Figure 4.1: Vertical and horizontal gradient filters

To obtain ground truth for the gradient network from the ground truth depth maps, I use a convolutional layer with 2 filters and kernel size 3x3. I set the weights of this layer at the start of the training manually, so it performs convolution with kernels that are shown in Figure 4.1. Learning rate for this layer is set to 0. When ground truth depth map is fed to this layer, it outputs two maps, one containing horizontal gradient and one containing vertical gradient. Resolution of these gradient maps is smaller by 2 in each direction compared to the ground truth, due to filtering without zero padding.

Chapter 5

Experiments and Results

To investigate effects of various parts and configurations of the proposed model on the resulting performance, I experimented with multiple setups of the model. In this chapter, I describe individual experiments I conducted and results arising from them. All experiments were conducted on NYU Depth v2 dataset, processed as described in Section 4.2. I evaluated results of each experiment quantitatively, as described in the first section and qualitatively, by visualizing the output depth map or output depth gradients.

Particularly important for this thesis is the experiment testing the performance of the normalized loss function for the refining network, because this loss function is a novel contribution to the problem of depth estimation. Equally important is the experiment which tests the effect of utilizing horizontal and vertical gradients as inputs to the refining network.

First two sections describe how I evaluated performance of each model and how I visualized output depth maps and gradients. Following sections address individual experiments and their results.

5.1 Evaluating Performance

To have a quantitative measure of the performance of different setups, I used metrics presented in Chapter 2, scale-invariant error proposed by Eigen *et al.* [5]

$$E(y, y^*) = \frac{1}{2n}\sum_i (\log y_i - \log y_i^* + \frac{1}{n}\sum_j (\log y_j^* - \log y_j))^2, \tag{5.1}$$

(denoted *sc-inv* in the tables of results) and normalized error described in Chapter 3 (denoted *norm.* in the tables of results). When evaluating performance, I had to either compare estimated depth maps with ground truth depth maps, or estimated gradients with ground truth gradients. In case of output being depth map, both output and ground truth are converted into meters before comparison by multiplying by 10. In case of output in the log space, the output is converted into linear space using function $g(y) = e^{(y-1)/0.179581}$ where y is the estimated depth, before being converted into meters. As explained in Section 4.2, the testing set contains images with a resolution of 420x320. I upscale the output of networks to this resolution before evaluating.

For evaluating the gradient network, only root mean square error and normalized error metrics were used.

5.2 Visualizing the Network's Outputs

When visualizing depth maps, I used two approaches. In experiments with the refining network, where the absolute scale of the depth map is important, I visualize the raw output of the network. In experiments with the global context network, the absolute scale of the output depth map is not important, because it is processed inside the model. Normalized loss produces depth values that can be orders of magnitude different than the ground truth depth values, or it can even prodcue negative depths. To be able to visualize these output depth maps, I fit the estimated depth map to the ground truth by first performing mean-variance normalization (subtracting mean and dividing by standard deviation) and afterwards multiplying by the standard deviation of the ground truth and adding the mean of the ground truth. Additionally, before visualization, depths are converted from linear space to log space using function $f(x) = 0.359162 * ln(x) + 1$. This produces images with better distribution of depth values and resulting depth maps are easier to understand.

Gradient visualization is done only during experiments with the gradient network, where there is no need for accurate absolute scale. I take similar approach as I used for visualizing depth map produced by the global context network, and fit the predicted gradient to the ground truth. Since the ground truth can contain negative values, I first convert ground truth to range $[0, 1]$ by subtracting minimum value and dividing by the difference between maximum and minimum values. After that, I fit the predicted gradients to the ground truth as described in the paragraph above.

5.3 Influence of Data Augmentation on the Performance of the Global Context Network

To test the effects of data augmentation on network's performance, I trained the global context network using normalized loss with each of the three versions of dataset mentioned in Section 4.2. Network was trained for 100,000 iterations each experiment. Table 5.1 shows a quantitative comparison of results achieved with each version of the dataset and Figure 5.1 shows a qualitative comparison of the output in each case.

	Data0	*Data1*	*Data2*
norm.	0.74621	0.70947	0.68924

Table 5.1: Performance of the global context network trained with different versions of the dataset

From Table 5.1 it is clear that data augmentation has a positive impact on the performance of the model. This is not unexpected, as data augmentation is a common practice. This experiment also shows that stronger transformations improve performance of the model more than mild transformations. It can be expected that increasing range of values for parameters of transformation does not improve performance indefinitely, and there is a certain point after which increasing the range of transformations introduce too much noise into data and performance will decline. Further experiments could find this point. Qualitative results in Figure 5.1 also confirm that data augmentation has a positive impact on performance. In this case however, it is not evident that stronger transformations result in visibly more accurate depth maps. Since quantitative results favor *Data2* version of the dataset, that is the version I used for training in the rest of the experiments.

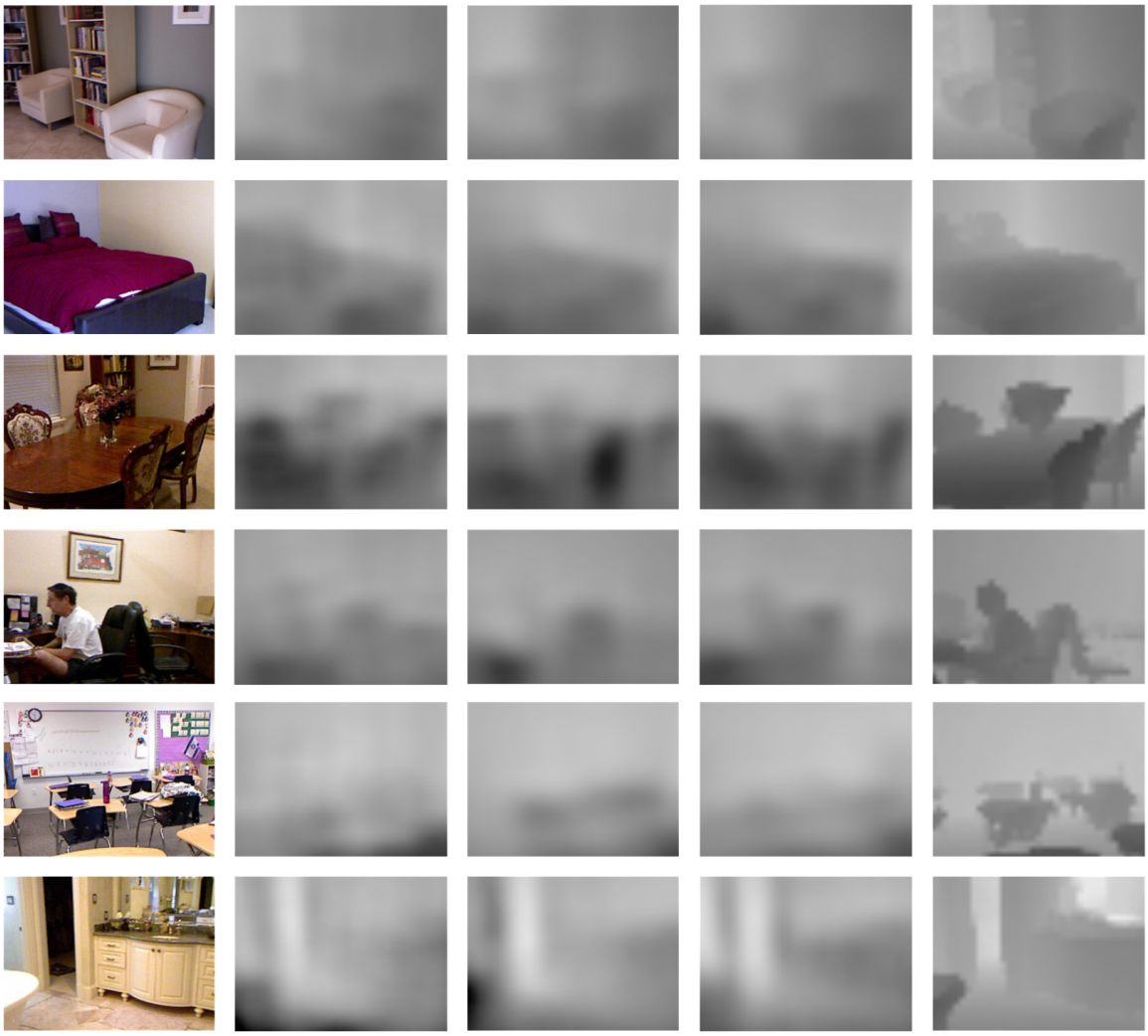


Figure 5.1: Qualitative comparison of depth maps estimated by a model trained using different versions of dataset. From left to right: input image, output of the network trained using *Data0* version, *Data1* version, *Data2* version, the ground truth

5.4 Comparing Different Loss Functions of the Global Context Network

In this experiment, I compared a performance achieved by using normalized loss function and other commonly used loss functions. I trained the global context network for 100,000 iterations using *Data2* version of the dataset, in each instance with a different loss function. Loss functions I used besides normalized loss, were scale-invariant loss (used by [4]), absolute depth difference (Euclidean distance) for baseline comparison and absolute depth difference in log space (used by [13]). Since neither normalized loss nor scale-invariant loss enforce output absolute values of estimated depths to be in an actual scale of the scene, it is suitable to compare their performance using metrics that are scale invariant. I also did comparison on absolute metric RSME to have an idea about the absolute scale of output depths. Quantitative results are presented in Table 5.2 and qualitative comparison in Figure 5.2

	abs loss	abs log loss	sc-inv loss	norm. loss
RMSE	**0.84530**	0.89869	2.43889	109.62362
sc-inv	0.17056	0.16944	**0.16336**	39.17647
normalized	0.73565	0.72686	0.71856	**0.68924**

Table 5.2: Performance of the global context network trained with different loss functions (best in each metric in bold)

As expected, absolute depth values estimated by the networks trained using scale invariant loss and normalized loss are at different scale then the ground truth depth. Output from the network trained using normalized loss produces large scale invariant error, which suggests that depths produced by the network are not just scaled, but also translated compared to the ground truth. Qualitative results show that the global context network trained using normalized loss visibly performs better in estimating rough structure of the depth map irrespective of scale and translation.

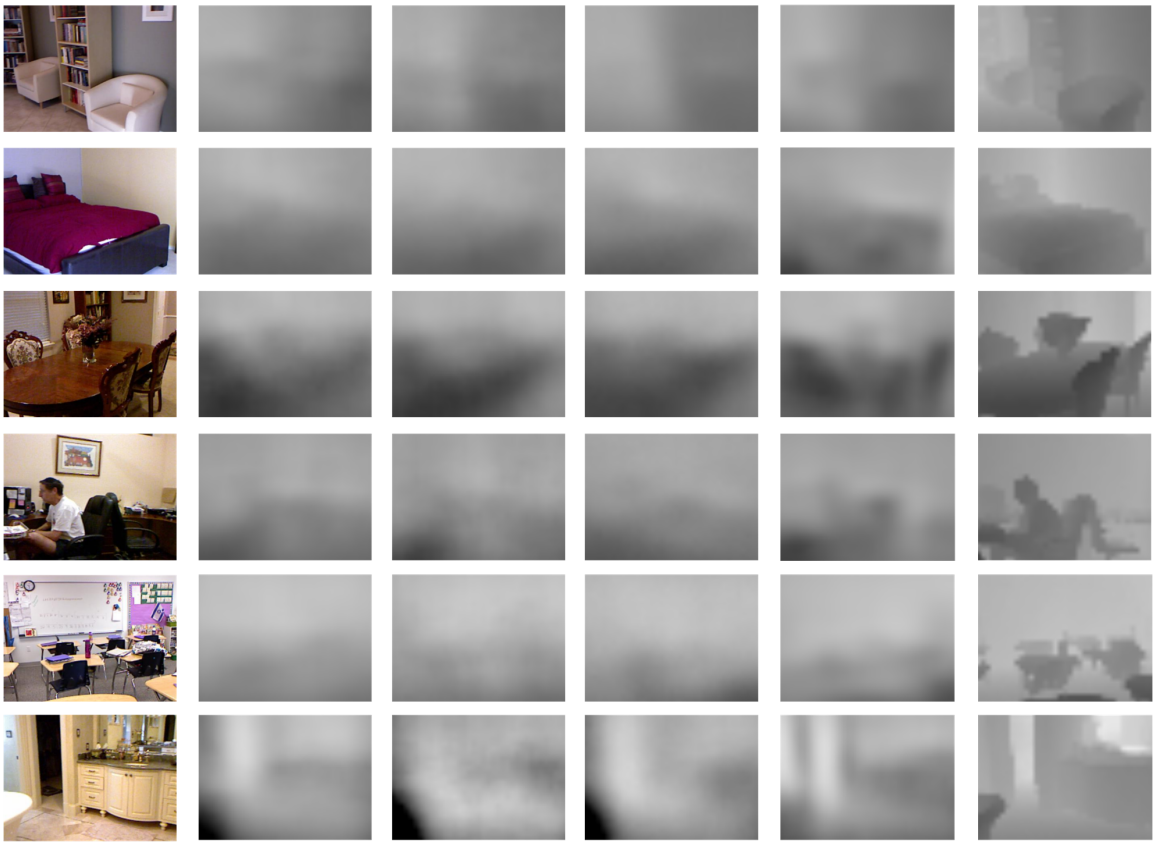


Figure 5.2: Qualitative comparison of depth maps estimated by models trained using different loss functions. From left to right: input image, output of the network trained using abs loss, log abs loss, scale-invariant loss, normalized loss, the ground truth

5.5 Effect of the Normalized Loss Function on the Performance of the Gradient Network

Similarly to experiment described in previous section, in this experiment I compare performance of normalized loss function, this time used during training of the gradient network. Training lasted 100,000 iterations and *Data2* version of the dataset was used. I compared normalized loss function to the Euclidean loss. Table 5.3 shows quantitative results and Figure 5.3 presents qualitative comparison.

	eucl. loss	norm. loss
rms	**0.06274**	13.44908
norm.	1.31700	**1.30201**

Table 5.3: Performance of the gradient network trained with different loss functions (best in each metric in bold)

Performance on root mean square metric shows that network trained with normalized loss outputs values that are on different scale than desired values. When comparison is made using normalized error, this network achieves better performance, however not with a significant lead. This is confirmed by qualitative comparison, where there is only a slight difference between different loss functions and there is no clearly better model.

5.6 Influence of Joint Training on the Global Context Network and the Gradient Network

As described in Section 3.4, I tried to jointly train the global context network and the gradient network with the aim of improving performance of both networks. I tested both joint architectures specified in Section 3.4 (architecture A is referred to as *jointA* and the architecture B is referred to as *jointB* in the tables containing comparison of the performance) and compared their results among themselves and with the results obtained by training each network separately. Training ran for 100,000 iterations using *Data2* version of dataset. Loss function in joint and separate training was normalized loss for both networks. Table 5.4 compares performance of jointly trained global context network to performance of separately trained global context network, Table 5.5 compares performance of jointly trained gradient network to performance of separately trained gradient network and Figures 5.4 and 5.5 show qualitative comparison for depth map and gradient estimation respectively.

	jointA	jointB	separate
norm.	0.70286	0.69712	0.68924

Table 5.4: Comparison of the performance of the global context network trained separately and jointly

Joint configuration performs better than separate configuration in the task of depth gradients estimation and worse in the task of depth estimation, according to normalized error metric from Tables 5.4 and 5.5. This may suggest that depth estimation task does not benefit from features that are used to estimate depth gradients, but for depth gradient

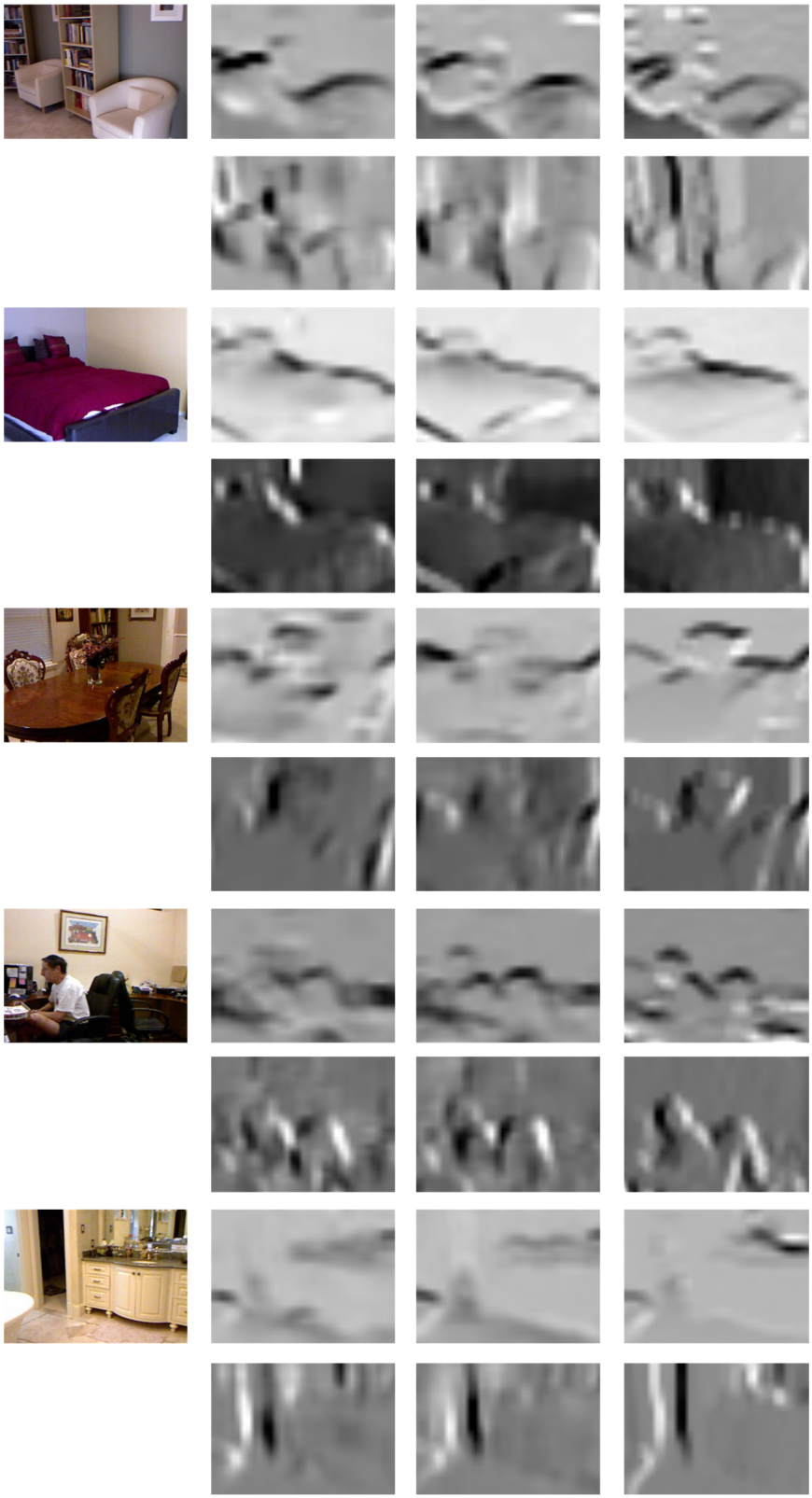


Figure 5.3: Qualitative comparison of horizontal and vertical gradients estimated by the gradient network trained using different loss functions. From left to right: input image, output of network trained using abs loss, normalized loss, the ground truth

	joint1	joint2	separate
rms	473.22585	1734.33860	**13.44908**
norm.	**1.29819**	1.30028	1.30201

Table 5.5: Comparison of the performance of the gradient network trained separately and jointly (best in each metric in bold)

estimation it is advantageous to utilize features used in depth estimation. It is interesting to note that architecture A, which contains less convolutional layers, performs better on depth gradient estimation than architecture B, which uses the same first three convolutional layers for both tasks, but the reverse is true for depth estimation.

Qualitatively, there are no significant differences between any of the configurations in either task.

5.7 Comparing Loss Functions of the Refining Network

This experiment investigates how the performance of the model changes with respect to loss function used for training the refining network. Separately trained global context network and gradient network utilizing normalized loss are used to produce rough estimates that are fed to the refining network. Loss functions tested in this experiments where absolute depth difference, absolute depth difference in log space, modified loss function L_a from Equation 3.3 and scale-invariant loss as defined in Equation 2.3 with $\lambda = 0.5$. Training ran for 60,000 iterations using *Data2* version of the dataset. Table 5.6 shows quantitative results and Figure 5.6 presents the qualitative comparison.

	abs	abs log	sc-inv + abs	norm. + abs
rel	0.35222	0.34202	**0.32031**	0.36544
sqr rel	0.50497	0.48068	**0.44049**	0.53726
rms	**1.11718**	1.15023	1.11758	1.17232
rms-log	0.68412	0.71098	**0.67993**	0.99812
log10	**0.13747**	0.14521	0.13967	0.18163
sc-inv	0.31390	0.31734	**0.25939**	0.81123
norm.	1.00571	1.06343	0.92391	**0.68475**
$\delta < 1.25$	**0.43280**	0.40845	0.41681	0.38253
$\delta < 1.25^2$	**0.74504**	0.71034	0.73548	0.67506
$\delta < 1.25^3$	**0.90485**	0.88887	0.90379	0.84161

Table 5.6: Comparison of the performance of different loss function used for training of the refining network (best in each metric in bold)

From qualitative evaluation, it is clear that the network trained using normalized loss produces depth estimates that are significantly sharper than the rest of the networks. Produced depth maps are on the level of depth maps produced by the state of the art approaches in terms of sharpness and alignment to local details. From Figure 5.6, it can be also inferred,

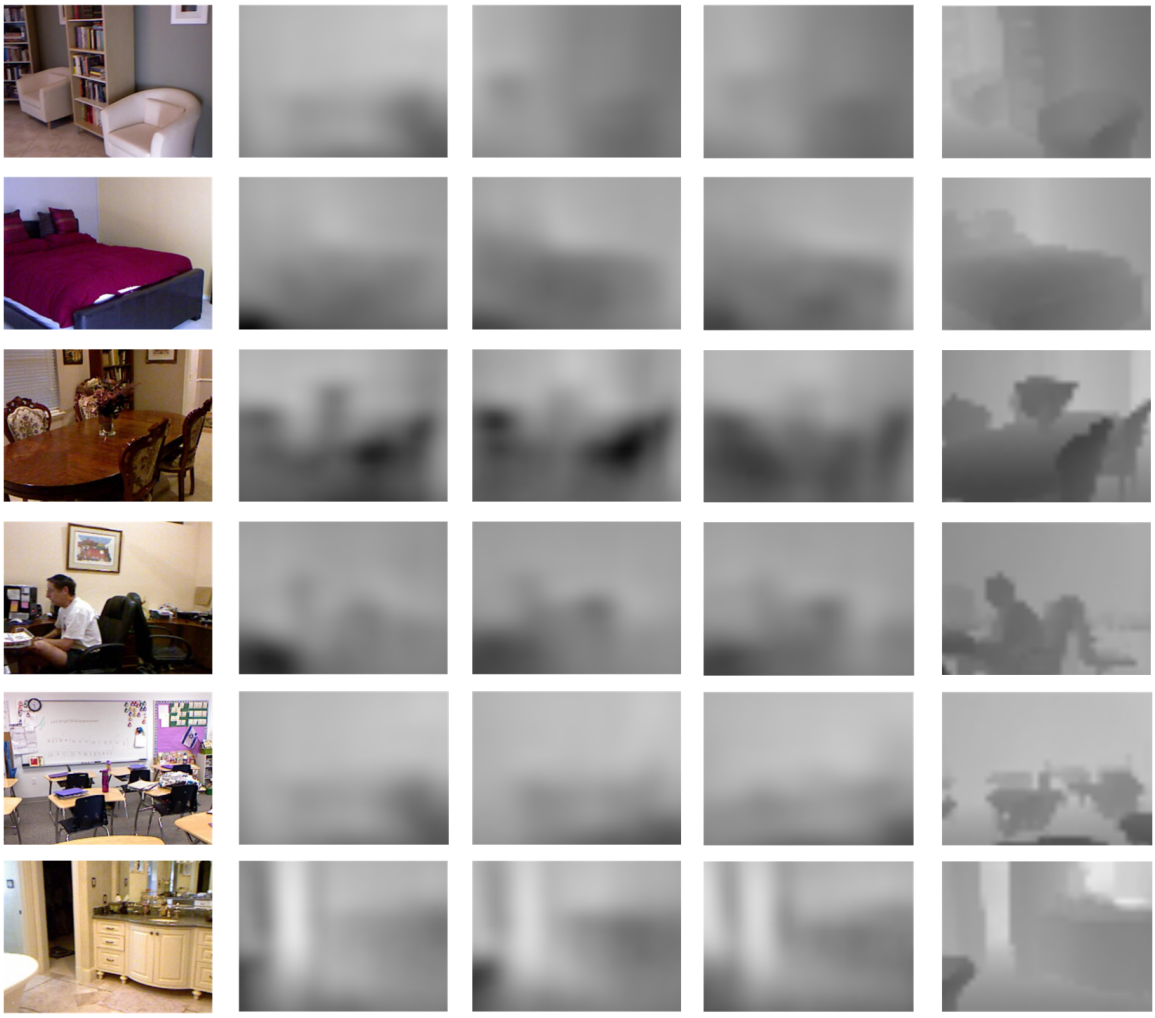


Figure 5.4: Qualitative comparison of the outputs of the global context network trained jointly with the gradient network and separately. From left to right: input image, output of network architecture A, architecture B, global context network trained separately, the ground truth

that the model using normalized loss too strongly correlates change in color in the input image with the change in depth in the resulting depth map. As quantitative evaluation shows that the network using normalized function performs worse than the other networks in all metrics except for the normalized error metric. This is caused by the network's inability to accurately predict absolute scale of depth values, as I shown in the Section 5.9.

It is possible that the low quality of depth maps produced by models utilizing other loss functions than normalized loss is caused by the short training procedure. Running the training longer would probably improve the results, but this was not possible for this thesis, due to time constrains. Nonetheless, it can be concluded, that the model that uses normalized loss learned faster to produce higher quality depth maps with a more detail.

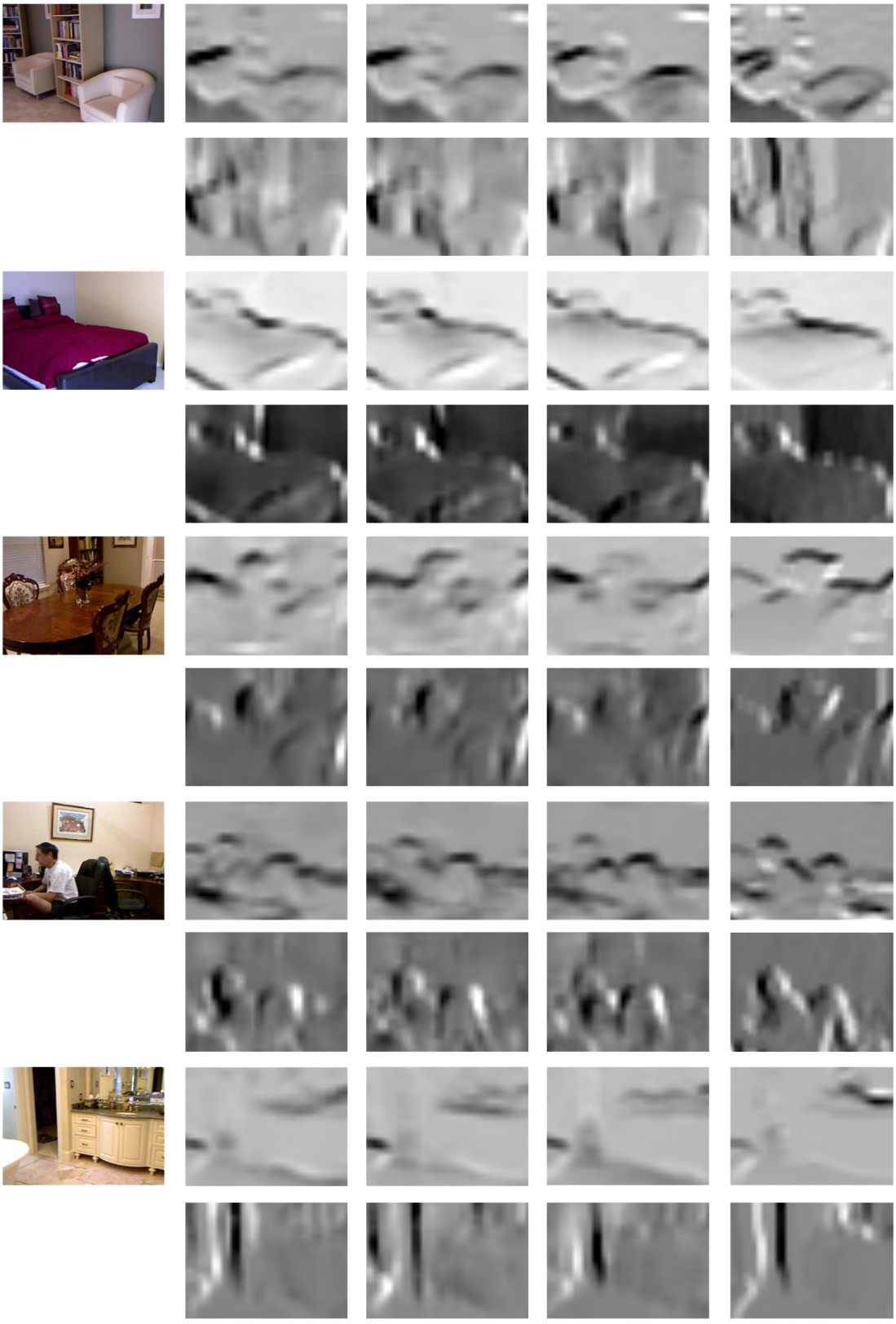


Figure 5.5: Qualitative comparison of the outputs of the gradient network trained jointly with the global context network and separately. From left to right: input image, output of network architecture A, architecture B, gradient network trained separately, the ground truth

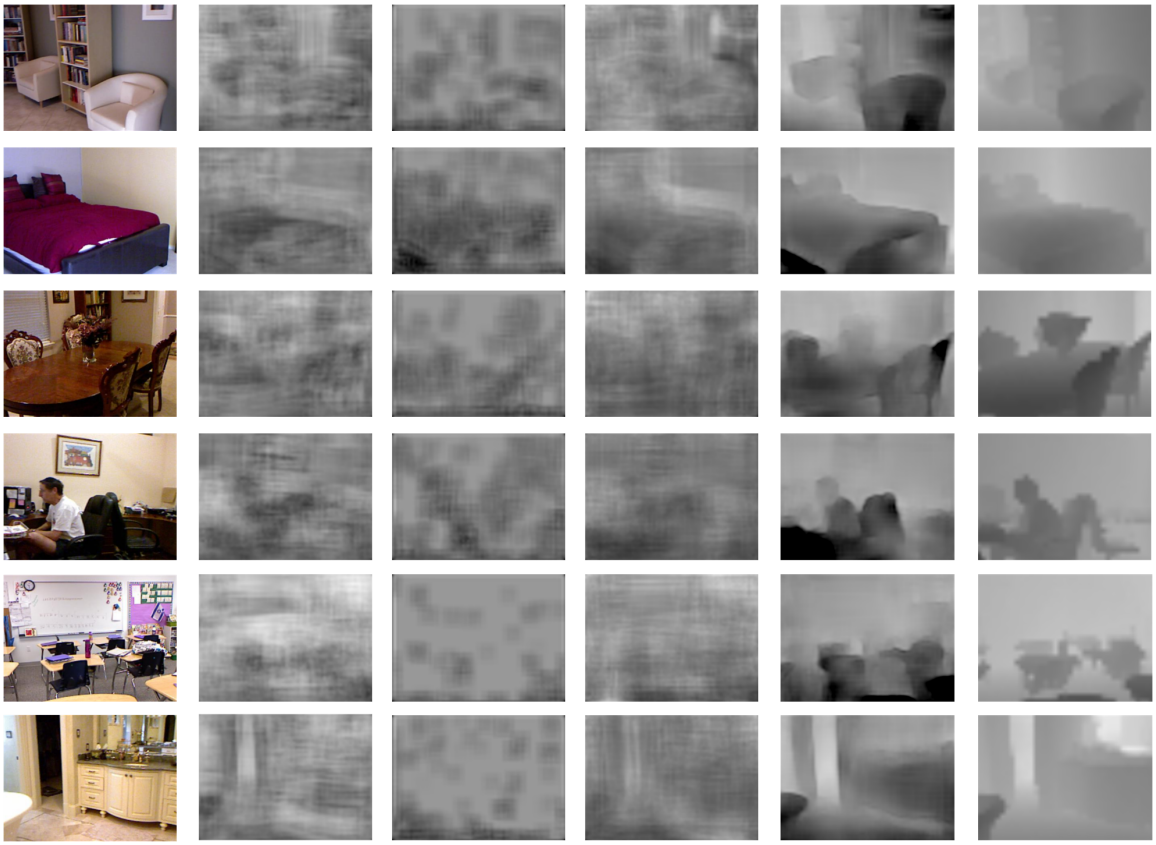


Figure 5.6: Qualitative comparison of the outputs of the refining network trained with different loss functions. From left to right: input image, output of the network trained using abs loss, abs log loss, sc-inv + abs loss, norm. + abs loss, the ground truth

5.8 Influence of the Estimated Gradients on the Refining Network

Refining network refines the coarse estimate from the global context network based on an input RGB image and depth gradients estimated by the gradient network. In this experiment I test the effects of the additional information about depth gradients on the performance of the refining network. I train the refining network in two setups, in both I use identical global context network trained separately from the gradient network and using normalized loss. First setup also contains the gradient network trained separately and using normalized loss, the second setup does not contain this network. Both the gradient network and the global context network are fixed in this experiment and only the refining network is trained. Normalized loss with absolute depth difference is used and the training runs for 60,000 iterations on *Data2* dataset. Quantitative comparison is shown in Table 5.7 and qualitative comparison of the results is in Figure 5.7.

Results from Table 5.7 and Figure 5.7 indicate that utilizing depth map gradients has a positive impact on the performance of the refining network, since the network utilizing these gradients performs better in every metric and produces depth maps that are smoother and better aligned to local details.

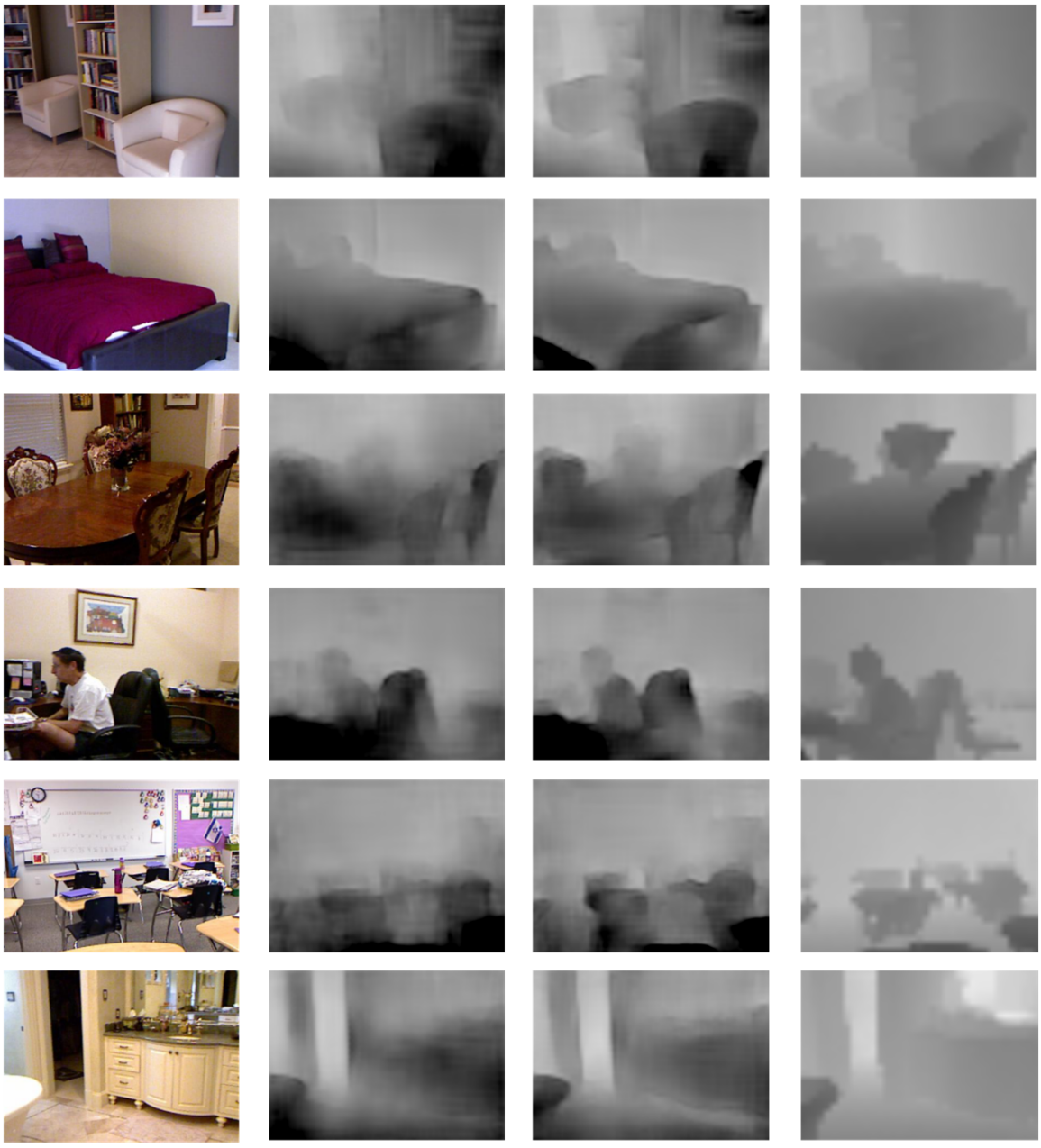


Figure 5.7: From left to right: input image, output of the refining network not utilizing depth gradients, output of the refining network utilizing depth gradients, the ground truth

	without gradient network	with gradient network
rel	0.38051	**0.36544**
sqr rel	0.57399	**0.53726**
rms	1.23184	**1.17232**
rms-log	1.18986	**0.99812**
log10	0.21422	**0.18163**
sc-inv	1.13405	**0.81123**
normalized	0.68597	**0.68475**
$\delta < 1.25$	0.34782	**0.38253**
$\delta < 1.25^2$	0.61948	**0.67506**
$\delta < 1.25^3$	0.79368	**0.84161**

Table 5.7: Influence of the estimated gradients on the performance of the refining network (best in each metric in bold)

5.9 Comparison to the Existing Approaches

In this section, I compare the proposed model to the existing solutions for depth estimation from a single image that utilize convolutional neural networks. I compare the proposed model to the model by Eigen *et al.* [5] that is similar in structure and size to the proposed model. Additionally, to assess the extend to which the errors of the proposed model are caused by the incorrect absolute scale of the estimated depths, I present a model with an oracle. This model is the same as the proposed model, with an exception that it already knows the true mean and variance of the target depth map and output depth map is modified to fit this value. This is done by subtracting the mean of the estimated depth map, dividing by the standard deviation of the estimated depth map, multiplying by the standard deviation of the target depth map and adding the mean of the target depth map. The proposed model was trained for 100,000 iterations. Table 5.8 presents performance comparison and Figure 5.8 contains outputs produced by the proposed model and the model in [5].

	model from [5]	proposed model	proposed model w/ an oracle
rel	0.215	0.38088	**0.16802**
sqr rel	0.212	0.58348	**0.15862**
rms	0.907	1.16903	**0.56936**
$\delta < 1.25$	0.611	0.40682	**0.76497**
$\delta < 1.25^2$	0.887	0.70939	**0.93206**
$\delta < 1.25^3$	0.971	0.87478	**0.97525**

Table 5.8: Comparison to the existing model by Eigen *et al.* [5] (best in each metric in bold)

Quantitative evaluation shows that the proposed model is inferior to the existing so-

lutions, considering that model by Eigen *et al.* that was used for comparison is the first model utilizing convolutional neural networks and all other existing approaches utilizing CNNs achieve better performance. Low performance of the proposed model can be attributed to the incorrect prediction of the absolute depth values. This can be seen from the results of the model with an oracle. For this model, the absolute scale is substituted from the ground truth and the model with an oracle achieves better performance than the model by Eigen *et al.* in every metric and even achieves better results than the current state of the art model in root mean squared error metric. This confirms that the model predicts the relative structure very well but lacks in estimating the absolute scale. Additionally, all parts of the proposed model were trained for 100,000 iterations, while the individual networks used in the model by Eigen *et al.* were trained for 1,500,000 iterations and more. This can also contribute to the lower performance.

Figure 5.8: From left to right: input image, produced by model by Eigen *et al.* [5], output of the proposed model, the ground truth

Chapter 6

Conclusion

This thesis dealt with a problem of depth estimation from a single image by convolutional neural networks. To that end, I proposed a three-part model that is derived from the work by Eigen *et al.* [5]. This model explicitly utilizes vertical and horizontal gradients of the depth map. Furthermore I proposed a novel normalized loss function for training convolutional neural networks that considers scale invariance problem that arises when estimating depth from a single image.

Results of the experiments show that utilizing vertical and horizontal gradients improves the performance of the network. This is evident in the quantitative evaluation of the performance of the performance of the model (Table 5.7) and in the qualitative assessment of the output depth maps (Figure 5.7).

Furthermore, results show that the normalized loss function improves the model's ability to estimate the relative depth structure of the scene compared to the other commonly used loss functions. Disadvantage of the normalized loss is that it does not force depth values to represent the true absolute scale of the scene. This can be partly compensated for by using a combined loss function that contains additional term that minimizes the absolute depth difference. More specifically, results show that the model utilizing normalized loss function produces much sharper depth maps with better alignment to local details than models utilizing other loss functions (Figure 5.6). Depth maps are qualitatively comparable to depth maps estimated by the state of the art approaches. However, the proposed model does not estimate the correct absolute scale of the scene very well, even when using the combined loss function. Quantitative comparison reflects this fact and the proposed model achieves worse performance than models utilizing other loss functions in every metric except for the normalized error, which evaluates correctness of the depth structure of the estimated depth map irrespective of the scale and translation (Table 5.6). The same is true for the comparison to the existing approaches. In case of substituting the true absolute scale of the scene into the resulting depth maps, the model achieves performance on par with the state of the art approaches and even achieves better results in the root mean squared error metric (Table 5.8). This shows that the model is able to estimate relative depth structure of the scene well, even with its training time being more than 10 times shorter than that of the comparable existing approaches. It also confirms that the weak point of the model is the estimation of the absolute scale of the scene.

Improvements in the model can possibly be made by using a larger architecture, e.g., using VGG network instead of AlexNet, or running the training process longer. This was not possible for this thesis because of the time constrains.

Since the weak point of the model is the estimation of the absolute scale of the scene,

future work should address this deficiency. One way to address this is to jointly estimate the mean and the variance of the depth map in addition to estimating the relative depth structure by normalized loss. Further, another network at the third level could be designed to refine estimates of the refining network. Depth estimate from this network could then be redirected to its input and it could iteratively refine its estimates until convergence.

Bibliography

[1] Brikbeck, N.; Cobzas, D.; Jägersand, M.: *Depth and scene flow from a single moving camera*. vol. 2. 3DPVT. 2010.

[2] Caruana, R.: Multitask Learning. *Machine Learning*. vol. 28, no. 1. 1997: pp. 41–75. ISSN 1573-0565. doi:10.1023/A:1007379606734.
Retrieved from: http://dx.doi.org/10.1023/A:1007379606734

[3] Deng, J.; Dong, W.; Socher, R.; et al.: ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*. 2009.

[4] Eigen, D.; Fergus, R.: Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture. *CoRR*. vol. abs/1411.4734. 2014.
Retrieved from: http://arxiv.org/abs/1411.4734

[5] Eigen, D.; Puhrsch, C.; Fergus, R.: Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. *CoRR*. vol. abs/1406.2283. 2014.
Retrieved from: http://arxiv.org/abs/1406.2283

[6] Fulkerson, B.; Vedaldi, A.; Soatto, S.: Class Segmentation and Object Localization with Superpixel Neighborhoods. In *Proc. ICCV*. 2009.

[7] Geiger, A.; Lenz, P.; Stiller, C.; et al.: Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*. 2013.

[8] Jia, Y.; Shelhamer, E.; Donahue, J.; et al.: Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*. 2014.

[9] Kim, S.; Park, K.; Sohn, K.; et al.: Unified Depth Prediction and Intrinsic Image Decomposition from a Single Image via Joint Convolutional Neural Fields. *CoRR*. vol. abs/1603.06359. 2016.
Retrieved from: http://arxiv.org/abs/1603.06359

[10] Krizhevsky, A.; Sutskever, I.; Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, edited by F. Pereira; C. J. C. Burges; L. Bottou; K. Q. Weinberger. Curran Associates, Inc.. 2012. pp. 1097–1105.
Retrieved from: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[11] Ladický, L.; Shi, J.; Pollefeys, M.: Pulling Things out of Perspective. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR

'14. Washington, DC, USA: IEEE Computer Society. 2014. ISBN 978-1-4799-5118-5. pp. 89–96. doi:10.1109/CVPR.2014.19.
Retrieved from: http://dx.doi.org/10.1109/CVPR.2014.19

[12] Levin, A.; Lischinski, D.; Weiss, Y.: Colorization Using Optimization. In *ACM SIGGRAPH 2004 Papers*. SIGGRAPH '04. New York, NY, USA: ACM. 2004. pp. 689–694. doi:10.1145/1186562.1015780.
Retrieved from: http://doi.acm.org/10.1145/1186562.1015780

[13] Li, B.; Shen, C.; Dai, Y.; et al.: Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*. 2015.

[14] Liu, B.; Gould, S.; Koller, D.: Single image depth estimation from predicted semantic labels. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. June 2010. ISSN 1063-6919. pp. 1253–1260. doi:10.1109/CVPR.2010.5539823.

[15] Liu, F.; Shen, C.; Lin, G.; et al.: Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields. *CoRR*. vol. abs/1502.07411. 2015.
Retrieved from: http://arxiv.org/abs/1502.07411

[16] Liu, M.; Salzmann, M.; He, X.: Discrete-Continuous Depth Estimation from a Single Image. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. June 2014. ISSN 1063-6919. pp. 716–723. doi:10.1109/CVPR.2014.97.

[17] Liu, M. Y.; Tuzel, O.; Ramalingam, S.; et al.: Entropy rate superpixel segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. June 2011. ISSN 1063-6919. pp. 2097–2104. doi:10.1109/CVPR.2011.5995323.

[18] Nathan Silberman, P. K., Derek Hoiem; Fergus, R.: Indoor Segmentation and Support Inference from RGBD Images. In *ECCV*. 2012.

[19] Roy, A.; Todorovic, S.: Monocular depth estimation using Neural Regression Forest. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*. 2016.

[20] Saxena, A.; Sun, M.; Ng, A.: Make3D: Learning 3D Scene Structure from a Single Still Image. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. vol. 31, no. 5. May 2009: pp. 824–840. ISSN 0162-8828. doi:10.1109/TPAMI.2008.132.

[21] Scharstein, D.; Szeliski, R.: A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision*. vol. 47, no. 1. 2002: pp. 7–42. ISSN 1573-1405. doi:10.1023/A:1014573219977.
Retrieved from: http://dx.doi.org/10.1023/A:1014573219977

[22] Schwing, A. G.; Urtasun, R.: *Computer Vision – ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI*. chapter Efficient Exact Inference for 3D Indoor Scene Understanding. Berlin, Heidelberg: Springer Berlin Heidelberg. 2012. ISBN 978-3-642-33783-3. pp. 299–313. doi:10.1007/978-3-642-33783-3_22.
Retrieved from: http://dx.doi.org/10.1007/978-3-642-33783-3_22

[23] Shotton, J.; Girshick, R.; Fitzgibbon, A.; et al.: *Decision Forests for Computer Vision and Medical Image Analysis*. chapter Efficient Human Pose Estimation from Single Depth Images. London: Springer London. 2013. ISBN 978-1-4471-4929-3. pp. 175–192. doi:10.1007/978-1-4471-4929-3_13.
Retrieved from: http://dx.doi.org/10.1007/978-1-4471-4929-3_13

[24] Simonyan, K.; Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*. vol. abs/1409.1556. 2014.

[25] Wang, P.; Shen, X.; Lin, Z.; et al.: Towards Unified Depth and Semantic Prediction From a Single Image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.