Learning Fine-Scaled Depth Maps from Single RGB Images

Jun Li
University of Bonn

Reinhard Klein
University of Bonn

Angela Yao
University of Bonn

Abstract

Inferring depth from a single RGB image is an ill-posed and inherently ambiguous problem. State-of-the-art deep learning methods can now estimate accurate depth maps, but when the maps are projected into 3D, they lack local detail and are often highly distorted. We propose a fast-to-train multi-scale convolutional neural network to predict depth and add two novel contributions which add accuracy and detail to the resulting depth maps. First, we define a set loss over multiple images. By regularizing the estimation between a common set of images, we prevent the network from over-fitting and can achieve better accuracy than competing methods. We further enhance our depth prediction with local gradient estimates that preserve sharp edges and detailing. Experiments on the NYU Depth v2 dataset show that our depth predictions outperform state-of-the-art and lead to 3D reconstructions that are realistic and rich with detail.

1. Introduction

Estimating depth for common indoor scenes from monocular RGB images is a challenging task in computer vision with widespread applications in scene understanding, depth-aware image editing or re-rendering, 3D modeling, robotics, etc. Given a single RGB image as input, the goal is to predict a dense depth map for each pixel. Inferring the underlying depth is an ill-posed and inherently ambiguous problem. In particular, indoor scenes have large texture and structural variations, heavy object occlusions and rich geometric detailing, all of which contributes to the difficulty of accurate depth estimation.

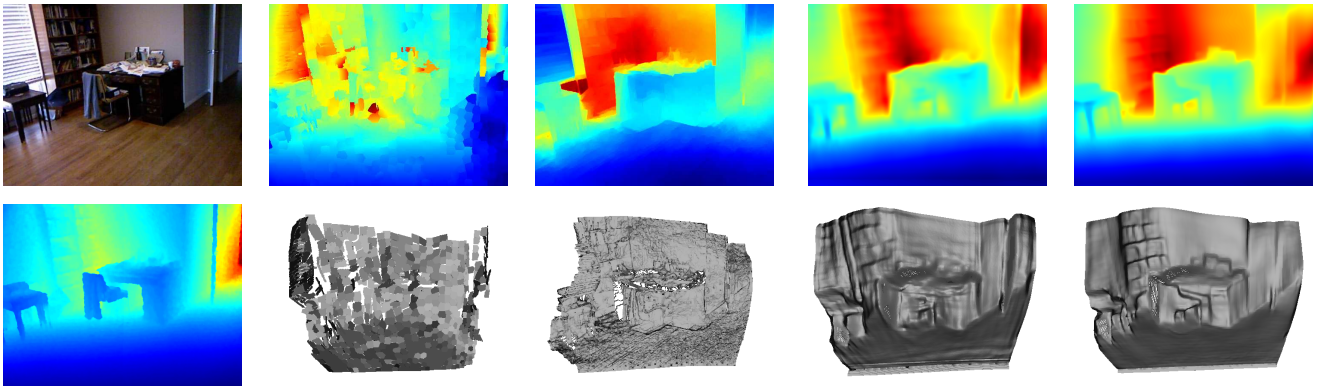
To deal with the ambiguity of estimating depth from a single image, previous works have relied on strong priors, based on either geometry or scene knowledge. Before RGB-D images were readily available, it was possible to construct simple 3D models from the image labels [6] or estimate mean depth values from image structure [20] but not predict dense depth maps. Once RGB-D data could be collected from laser or depth cameras on a large scale, it became feasible to apply learning-based approaches for dense depth estimation [7, 13, 16, 17, 23]. Nevertheless, with only hand-crafted features, the inferred depth maps can only approximate the global layout of an image and not depth at finer scales (see example in Fig. 1(b)).

In recent years, deep learning methods have demonstrated their strength in a multitude of vision tasks, including monocular depth estimation. Depth accuracy on a per-pixel basis has improved significantly with elegantly designed convolutional neural networks (CNNs) [2, 4, 8, 10, 15, 12, 21]. Rather than yielding approximations of the coarse depth of large structures such as walls and ceilings, state-of-the-art multi-scale networks [1] can capture finer-scale items such as furniture and home accessories.

The pinnacle of success for depth estimation is the ability to generate realistic and accurate 3D scene reconstructions from the estimated depths. Despite the impressive evaluation scores of recent works [1, 12], however, the estimated depth maps still suffer from artifacts at finer scales and have unsatisfactory alignments between surfaces. These distortions become especially prominent when projected into 3D, for example in Fig. 1(c,d). It is with the goal of 3D reconstruction and preserving local structure and clean detailing that we motivate our work. We want to benefit from the increased accuracy of deep learning, but with standard CNN architectures, pooling after convolution repeatedly shrinks image resolution and loses image detail. Other end-to-end estimation tasks such as optical flow and semantic segmentation face similar challenges and have introduced up-convolution strategies [14], data sub-sampling [22], and concatenated feature maps or predictions [3] from previous parts of the network to maintain resolution.

Inspired by these works and by the success of multi-scale CNNs [2, 1], we introduce a three-stage, coarse-to-fine CNN that predicts accurate depth maps. We apply four layer fusions to pass features and accumulate image understanding across the scales.In particular, the two skip layer fusions from the scale one to two offers significant speedup for network convergence during training.

A key improvement that we make over previous state-of-the-art [1] is the use of a novel set image loss. This loss is defined jointly over multiple images, where each image is a transformed version of an original via standard data



(a) Ground truth (b) Liu et al. [13] (c) Liu et al. [12] (d) Eigen et al. [1] (e) Ours

Figure 1. State-of-the-art 3D reconstructions from estimated depth maps (b-d), along with ground truth (a). Our estimated depth maps(e) are more accurate than state-of-the-art methods with fine-scaled details. Note the colour values of each depth map is individually scaled.

augmentation techniques. The set loss considers not only the accuracy of each transformed image's output depth, but also has a regularization term to minimize prediction differences within the set. We find that the added regularization term greatly improves the accuracy of the estimated depth (reducing RMS error by 5%).

To capture the finer details of a scene, we leverage the information contained in depth gradients. We learn a CNN, using the same architecture that we use for depth estimation, to estimate depth gradients and then solve for a refined depth map which miminizes the difference in estimated depth and estimated gradients. The final refined depth is accurate, clean with local detailing and has fewer artifacts.

Our main contributions can be summarized as follows

- We introduce a novel set image loss which tries to minimize the differences in estimated depth of related images. This form of regularization makes better use of augmented data and promotes stronger network generalization, as demonstrated by our accurate depth estimates which surpass state-of-the-art methods [1].
- We propose an enhancement method that optimizes depth with respect to depth gradient estimates. The enhanced depth map is more accurate and when projected into 3D, is less distorted and richer with detail.
- We propose a fast-to-train multi-scale CNN architecture. By adding skip layers between the scales, we better leverages intermediate outputs and simplify the learning at finer scales, resulting in a fast-converging network with a low memory footprint.

2. Related Work

Depth estimation and 3D reconstruction is a rich field of study and we focus on methods using single images as input. A key strategy in early works for handling depth ambiguity given a single input image was to use strong assumptions and prior knowledge. For example, Saxena *et al*. [16, 17] devised a multi-scale MRF, but assumed that all scenes were horizontally aligned with the ground plane. Hoiem *et al*. [6], instead of predicting depth explicitly, estimated geometric structures for major image regions and composed simple 3D models to represent the scene.

Once the capture of RGBD images became readily available, data-driven learning-based approaches gained popularity. Karsch *et al*. [7] proposed a non-parametric method to transfer depth from aligned exemplars and formulated depth estimation as an optimization problem with smoothness constraints. Liu *et al*. [13] modelled image regions as super-pixels and used discrete-continuous optimization for depth estimation; this was further improved by integrating mid-level region features and global scene layout in [23]. Others tried to improve depth estimations by exploiting semantic labels [5, 9, 11]. All of the above mentioned methods use hand-crafted features; resulting depth estimates are coarse and lack the finer details necessary for many applications in computer vision and graphics.

Deep learning has been proven to be highly effective for depth estimation [1, 2, 4, 8, 10, 12, 21, 15]. Liu *et al*. [12] combined CNNs and conditional random fields in a unified framework to jointly learn unary and pairwise potentials with CNNs. They predicted depth at a superpixel level which works well for preserving edges, but when projected to 3D, suffers from distortions and artifacts, as each superpixel region retains the same or very similar depth after an in-painting post-processing. Roy *et al*. [15] formulated a neural regression forest for the problem of monocular depth estimation. More recently, Garg *et al*. [4] proposed a unsupervised approach for predicting depth maps by training an autoencoder with small but known camera motions on image pairs. Kim *et al*. [8] jointly estimated a depth map and intrinsic images from single image input.

Eigen *et al*. [1, 2] proposed the use of a multi-scaled

deep network for predicting depth. Their network estimates high-resolution depth maps and is able to capture structural detailing that had not been seen before in works previously. Our system architecture is based on the network of [1]; however, our network has key differences in layer fusion and reshaping, which leads to fast convergence during training. We discuss these differences in detail in Section 3.

3. Network Architecture

3.1. Architecture of Depth Estimation Network

The architecture, kernel types and sizes for each layer and the learning rates are shown in Figure 2. At each scale, we use as input a re-sized and cropped 232×310 version of the original RGB image. The first scale accumulates global information through a downsampling net with two fully connected layers after five convolutional layers. This scale provides a global view and is important for accurate depth estimation as it captures the underlying structure of a scene. This was also shown in Li *et al.*'s [10] work based on multi-scale image patches and they found that the largest scale patches (407×407 pixels) had the biggest contribution in the depth estimation. Our convolutional layers in Scale 1 are initialized with the VGG parameters [19], while the two fully-connected layers are initialized randomly.

The second scale predicts a coarse 55×75 depth map based on the RGB input and three layer fusions (two skip layer connections from layers 1.3 to 2.2, 1.4 to 2.4 and a concatenation from the Scale 1 output 1.7 to 2.6; see Figure 2). At Scale 2, the first two layers are both a convolution and pooling layer, with no padding and stride two. This reduces the input resolution to 1/4 of the original. We do not use zero-padding at the initial layers and let the input shrink to avoid artifacts at the boundaries and corners. Other layers in Scale 2 are convolutional layers with normal zero-padding. We jointly train Scale 1 and Scale 2 using the loss described in Section 4. The third scale refines the estimate to a higher resolution 111×150 depth map based on the RGB input and concatenation of the Scale 2 depth output 2.9 to 3.3. Again, we do not use zero-padding for the first convolution and pooling layer at this scale.

At an initial glance, our network is similar in structure to the three-scaled network of Eigen *et al.* [1]. The key difference lies in the way we propagate information across the scales. First, we add two skip layers to fuse intermediate outputs from Scale 1 to Scale 2. Each skip layer connection has a 5×5 convolution and ReLU layer along with up-sampling. These two skip layers provide a hierarchical propagation of information between the scales and results in much faster network convergence during training. To accommodate three layer fusions from Scale 1 to Scale 2 and still have sufficient downstream processing, we extend the depth of Scale 2 to nine layers. A second difference of our design is the last fully connected layer at Scale 1 and reshaping of the output. In [1], the last fully connected layer outputs a large feature vector of 17024 dimensions, which is then reshaped to a 2D 19×14 map with 64 channels. In comparison, our last fully connected layer has only 4125 dimensions and is reshaped into a 2D 55×75 map with only a single channel which can already be considered a coarse depth prediction. This modification greatly reduces computational expense, without any degradation on accuracy of estimated depth.

3.2. Architecture of Gradient Estimation Network

We try to capture local structure and detail with gradients. We found that adding gradient constraints into the depth estimation loss, like in [1] did not have much impact; rather we learn a CNN to estimate gradients, using the same architecture as our depth estimation network. The only difference is that the output dimensions of the layers 1.7, 2.9, and 3.6 is now two (for X and Y gradients) instead of one (for depth estimation). The gradient estimates are then combined with the depth estimate into a refined depth (see Section 4.3).

4. Learning

4.1. Depth Training Loss

We define a difference between depth maps D_1 and D_2 as

$$E(D_1, D_2) = \frac{1}{n} \sum_{p=1}^{n} (D_1^p - D_2^p)^2, \qquad (1)$$

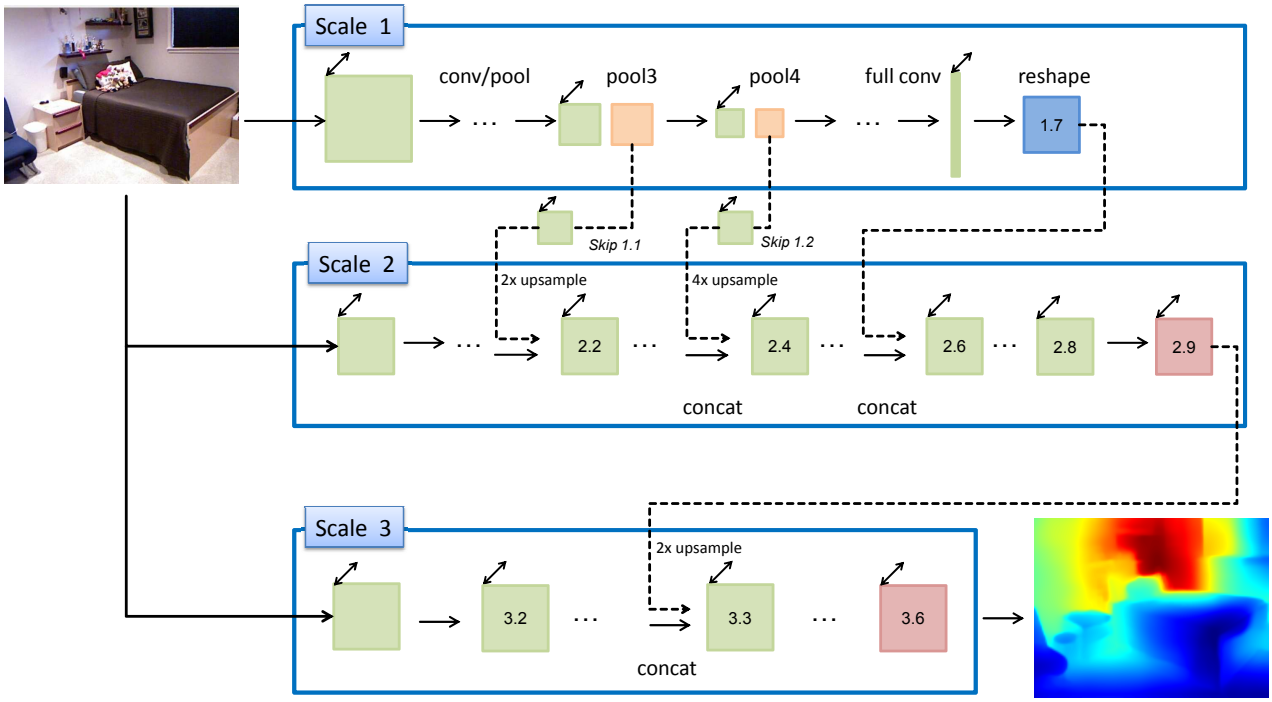
where p is a pixel index and n is the number of valid depth pixels[1]. A simple error measure comparing an estimated depth map D_i and its corresponding ground truth D_{gt_i} can then be defined as $E(D_i, D_{\text{gt}_i})$.

In many machine learning problems, it is now standard practice to augment the training set with transformed versions of the original training samples. By learning with the augmented set, the resulting classifier or regressor should be more robust to these variations. The standard loss is then computed over some batch of the augmented set, where transformed samples are treated as standard training samples. However, there are strong relations between original and transformed samples that can be further leveraged during training. For example, a sample image that is re-coloured to approximate different lighting conditions should have exactly the same depth estimate as the original. A flipped sample will also have the same depth estimate as the original after un-flipping the output and so on.

Based on this observation, we define the a loss based on the estimation errors of single images L_{single} and a con-

[1] Invalid pixels are parts of the image with missing ground truth depth; such holes are typical of Kinect images and not considered in the loss.

232 x 310



	layers	1.1	1.2	1.3	1.4	1.5	1.6	1.7	skip 1.1	skip 1.2
Scale 1	size	113x152	56x76	28x38	14x19	7x9	1x1	55x75	28x38	14x19
	#convs	2	2	3	3	3	2	-	1	1
	#chan	64	128	256	512	512	4125×D	D	64	64
	ker.sz	3x3	3x3	3x3	3x3	3x3	-	1x1	5x5	5x5
	l.rate	0.0001	0.0001	0.0001	0.0001	0.0001	0.1	0.1	0.01	0.01

	layers	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	2.9
Scale 2	size	55x75	55x75	55x75	55x75	55x75	55x75	55x75	55x75	55x75
	#chan	96+64	64	64+64	64	64+D	64	64	64	D
	ker.sz	9x9	5x5	5x5	5x5	5x5	5x5	5x5	5x5	5x5
	l.rate	0.001	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.001

	layers	3.1	3.2	3.3	3.4	3.5	3.6
Scale 3	size	111x150	111x150	111x150	111x150	111x150	111x150
	#chan	96	64	64+D	64	64	D
	ker.sz	9x9	5x5	5x5	5x5	5x5	5x5
	l.rate	0.001	0.01	0.01	0.01	0.01	0.001

Figure 2. Our depth estimation network architecture and layer details. D is the number of output channels in the prediction map; $D = 1$ for depth estimation and $D = 2$ for gradient estimation. *size* is the output map resolution of each layer or layer block. *#convs* is the number of convolutional layers in each layer block. *#chan* is the channel size of output feature map. *ker.sz* is the filter size. *l.rate* is the learning rate. Specific layers with concatenated feature maps are denoted by their layer index. The reshaped layer is blue while prediction layers are pink. Some pooling layers and ReLU layers are omitted from the figure for clarity.

straint term L_{set}, based on a set of transformed images, with λ as a weighting parameter between the two:

$$L = L_{\text{single}} + \lambda \cdot L_{\text{set}}. \tag{2}$$

L_{single} is simply the the estimation error of each (augmented) image considered individually, *i.e.*

$$L_{\text{single}} = \frac{1}{N} \sum_{i=1}^{N} E(D_i, D_{\text{gt}_i}), \tag{3}$$

where N is the number of images in image set $(I, f_1(I), f_2(I)...f_{N-1}(I))$, I the original image, while f are data augmentation transformations such as colour adjustment, flipping, rotation, skew, *etc*. L_{set} is defined as the difference between estimates in an image set:

$$L_{\text{set}} = \frac{1}{N-1} \sum_{i} \sum_{j, j \neq i} E(D_i, g_{ij}(D_j)). \tag{4}$$

Here, D_i and D_j are the depth estimates of (transformed) images I_i and I_j and g is a mapping between the two transformations, *i.e.* $g_{ij}(\cdot) = f_i(f_j^{-1}(\cdot))$. For all f which are

spatial transformations, g should realign the two estimated depth maps into a common reference frame for comparison. For colour transformations, f^{-1} is simply an identity function. L_{set} can be considered a regularization term which encourages all the depth maps of an image set to be the same, after accounting for the transformations.

4.2. Gradient Loss

Loss for the gradient estimation network is defined by the same set loss as in Eqs. 2, 3 and 4, with differences for two gradient maps G_1 and G_2 given as

$$E(G_1, G_2) = \frac{1}{n}\sum_{p=1}^{n}[(G_{1_x}^p - G_{2_x}^p)^2 + (G_{1_y}^p - G_{2_y}^p)^2], \quad (5)$$

where n is the number of valid depth pixels and G_x^p and G_y^p the X and Y gradients at pixel p respectively.

4.3. Depth Refinement

We combine the estimated depth D_{est} and estimated gradients G_x and G_y into a refined depth map D_r with the following optimization:

$$\begin{aligned} D_r^* = \underset{D_r}{\operatorname{argmin}} \sum_{p=1}^{n} \phi(D_r^p - D_{\text{est}}^p) + \\ \omega \sum_{p=1}^{n} \left[\phi(\nabla_x D_r^p - G_x^p) + \phi(\nabla_y D_r^p - G_y^p)\right]. \end{aligned} \quad (6)$$

where $\phi(x) = \sqrt{x^2 + \epsilon}, \epsilon = 10^{-4}$ (robust $L1$). ∇_x, ∇_y are x, y gradient operators on D_r (we use filters $[-1, 0, 1]$, $[-1, 0, 1]^\intercal$). p is the pixel index and n the number of pixels in a depth map. We solve for D_r using an iteratively reweighted least squares, though one could alternatively combine the estimated depth and gradients in a pairwise MRF.

4.4. Implementation details

We apply the same implementation for both the depth estimation and the depth gradient estimation networks.

Initialization-wise, the convolutional layers of Scale 1 in both networks are initialized with weights from the VGG network [19]. The fully connected layers in Scale 1 and layers in Scale 2 and 3 are initialized randomly. We apply dropout only in layer 1.6, with a rate of 0.5.

Learning rates are specified in Fig. 2 and scaled by a factor of 0.1 at every epoch. Our network is fast to train; only 2 to 3 epochs are required for convergence (see Fig. 3). During the first 20K iterations, we stabilize training by applying gradient clipping. We clip the differences between the estimation and ground truth and estimates between two samples in the set to be within [-0.1, 0.1]. We apply two training phases; first, we jointly train Scales 1 and 2 to generate a coarse depth map Weights of these two scales are then fixed to train Scale 3 for higher resolution depth prediction. The same loss is used at each scale.

Regularization constants λ and ω are set to 1 and 10 respectively. Preliminary experiments showed that different values of λ does not affect the resulting accuracy but a larger λ does slow down network convergence. ω was set by a validation set; having a larger ω overemphasizes artifacts in the gradient estimates and leads to less accurate depth maps.

Batch Size plays a large role in the speed of network convergence and a batch size of 1 was the fastest. Note that because our loss is defined over an image set, a batch size of 1 is effectively a mini-batch of size N, depending on the number of image transformations used.

5. Experimentation

5.1. Dataset & Evaluation

We use the NYU Depth v2 dataset [18] according to the standard scene split. From the 249 scenes for training, we extract about 220k training images. Depth images and RGB images are synchronized according to the time stamp; corrupt images are filtered out according to file size. RGB images are down-sampled by half to 240×320 and then cropped to 232×310 to remove the blank boundaries after alignment with depth images. Depths are converted to log-scale while gradients are kept in a linear scale.

We evaluate our proposed network and refinement method on the 654 NYU Depthv2 [18] test images. Since our depth output is 111×150 and a lower resolution than the original NYUDepth images, we bilinearly upsample our depth map (4x) and fill in missing borders with a cross-bilateral filter, similar to previous methods [1, 2, 10, 12, 13]. We then evaluate our predictions in the valid Kinect depth projection area, using the same measures as previous work:

- mean relative error (rel): $\frac{1}{T}\sum_i^T \frac{|d_i^{gt} - d_i|}{d_i^{gt}}$,
- mean $\log_{10}$ error ($\log_{10}$): $\frac{1}{T}\sum_i^T |\log_{10} d_i^{gt} - \log_{10} d_i|$,
- root mean squared error (rms): $\sqrt{\frac{1}{T}\sum_i^T \left(d_i^{gt} - d_i\right)^2}$,
- thresholded accuracy: percentage of d_i s.t. $\max\left(d_i^{gt}/d_i, d_i/d_i^{gt}\right) = \delta < thr$.

In each measure, d_i^{gt} is the ground-truth depth, d_i the estimated depth, and T the total pixels in all evaluated images. Smaller values on rel, log_{10} and rms error are better and higher values on percentage(%) $\delta < thr$ are better.

We make a qualitative comparison by projecting the estimated 2D depth maps into a 3D point cloud[2]. Represen-

[2]The 3D projections are computed using the Kinect camera projection matrix and the resulting point cloud is rendered with lighting.

Method	rel	log10	rms	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Karsch *et al*. [7]	0.35	0.131	1.2	-	-	-
Liu *et al*. [13]	0.335	0.127	1.06	-	-	-
Ladicky *et al*. [9]	-	-	-	0.542	0.829	0.941
Li *et al*. [10]	0.232	0.094	0.821	0.621	0.886	0.968
Liu *et al*. [12]	0.213	0.087	0.759	0.650	0.906	0.976
Eigen *et al*. [1]	0.158	-	0.641	0.769	0.950	0.988
Single Image Loss	0.161	0.068	0.640	0.765	0.950	0.988
Set Image Loss	0.153	0.065	0.617	0.786	0.954	0.988
Single Image w/ Gradient Refinement	0.158	0.067	0.633	0.770	0.951	0.988
Set Image w/ Gradient Refinement	**0.151**	**0.064**	**0.611**	**0.789**	**0.955**	**0.988**

Table 1. Accuracy of depth estimates on the NYU Depth v2 dataset, as compared with state of the art. Smaller values on rel, log_{10} and rms error are better; higher values on percentage(%) $\delta < thr$ are better.

tative samples are shown in Fig. 5; the reader is referred to the Supplementary Material for results over the test set.

5.2. Depth Estimation Results

We tally the accuracy of our depth estimates and show the results in Table 1. Using only the single image loss, results are marginally worse than state-of-the-art [1] and a bit better once we introduce the refinement with the gradient estimates. With the set image loss, however, our depth estimates are much more accurate, especially in terms of root mean square (rms) error and thresholded accuracy with $\delta < 1.25$. We further improve the accuracy with the gradient estimates.

Fig. 5 compares our depth estimates with other state-of-the-art methods. When assessing only based on the 2D depth images, the differences between the methods are not so easy to see. However, when projected into 3D, the strength of our method becomes immediately apparent. For example, Liu *et al*. [12], which used superpixels for depth estimation, preserves edges in the 2D depth map but in 3D, suffers gross distortions and artifacts. The results of Eigen *et al*. [1] are much better than [12] but still worse than ours. Our 3D reconstructions are cleaner and smoother in the appropriate regions, *i.e*. walls and flat surfaces. Corners and edges are better preserved and the resulting scene is richer in detailing with finer local structures. For example, in the cluttered scenes of Fig. 5(c,d,e), [1] has heavy artifacts in highly textured areas, *e.g*. the picture on the wall of (d), the windows in (e) and in the regions with strong reflections, the kettle in (c) and the cabinet in (e). Our results are robust to these difficulties and give a more faithful reconstruction of the underlying objects in the scene.

5.3. Set Loss & Data Augmentation

Our proposed set image loss has a large impact in improving the accuracy of our estimated depths. For the reported results in Table 1, we used three images in the image set: I, flip(I) and colour(I). The flip is around the vertical axis, while the colour operation includes randomly increasing or decreasing brightness, contrast and multiplication with a random RGB value $\gamma \in [0.8, 1.2]^3$. Preliminary experiments showed that adding more operations like rotation and translation did not further improve the performance so we omit them from our training. We speculate that the flip and colouring operations bring the most global variations but leave pinpointing the exact cause for future work.

5.4. Multiple Scales & Refinement

The first two scales fully capture the layout of the scene; adding Scale 3 does not quantitatively impact the results and this was also observed in [1]. Qualitatively, however, there is a big difference between two versus three scales (see Fig. 4). Only with 3 scales can one capture the finer details. Similarly, our gradient refinement does not have significant quantitative improvement according to the evaluation measures in Table 1. Qualitatively, however, there is a large impact on the estimated depth maps, with the difference becoming especially prominent once projected into 3D (compare third and fourth rows of Fig. 4).

5.5. Training Strategy, Convergence and Timing

We show the convergence behaviour of our network for the joint training of Scale 1 and 2 in Fig. 3. Errors decrease faster with a batch size of 1 in comparison to a batch size of 16, and requires only 0.6M gradient steps or 2-3 epochs to converge. For the convergence experiment, we compare the single image loss (batch size 1, 16) with the set image loss as described in Section 5.3 and observe that errors, as expected are lower with the set loss but convergence still occurs quickly. Note that the fast convergence of our network is not due to the small batch size, but rather the improved architecture with the skip layers. The network architecture of [1] requires a total of 2.5M gradient steps to converge (more than 100 epochs) with batchsize 16, but even when trained with a batch size of 1, does not converge so quickly.

Our network architecture and selection of skip and con-

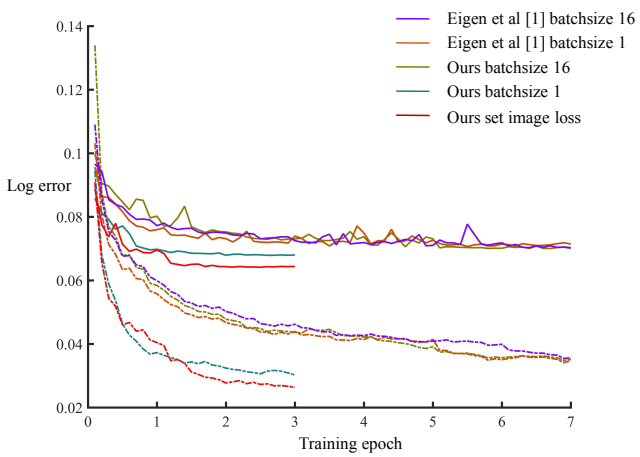


Figure 3. A comparison of the log_{10} training and test errors between our proposed network and [1]. For clarity, we plot the log error at every 0.1 epochs, and show only the first 7 epochs, even though the method of [1] has not yet converged. The dashed lines denote training error, and solid lines denote testing error. For our batch 1 and batch 16 results, we show the errors for the single image loss, and compare with our set image loss.

catenation layers propagates information across the scales, making the overall network easier to learn. Training for depth gradient estimates is even faster and converges within one epoch. Overall, our training time is about 70 hours (50 hours for depth learning and 20 hours for gradient learning) on a single GPU TITAN X.

6. Conclusion

We have proposed a fast-to-train multi-scale CNN architecture for accurate depth estimation. By adding skip layers between the scales, we can better leverage both intermediate and final outputs, thereby simplifying the learning task at finer scale and speed up the network convergence. To predict more accurate depth map with fine-scaled details, we introduce two novel contributions. First, we define a set loss jointly over multiple images. By regularizing the estimation between images in a common set, we achieve better accuracy than previous work. Second, we learn local depth gradients using the same network architecture. By optimizing the depth map with local gradient estimates, we further enhance our depth prediction and recover finer details. Experiments on the NYU Depth v2 dataset shows that our depth predictions not only outperform state-of-the-art but also lead to 3D reconstructions that are more accurate and rich with details. In the current work, we have addressed only the estimation of depth and depth gradients from an RGB source. It is likely that by combining the task with other estimates such as surface normals and semantic labels, one can improve the depth estimates, which we leave for future work.

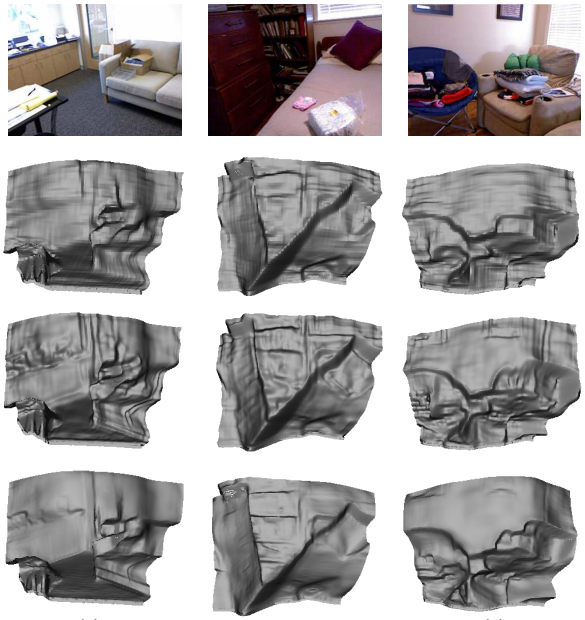


Figure 4. Comparisons of depth estimates from different stage. To better visualize the differences, we project the estimated depth map to 3D point clouds. The first row is the RGB image; the second the output from Scale 2; the third the output from Scale 3 and the last row is the output from depth optimization. Note the gradual improvement in detailing as one progresses downwards.

References

[1] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 1, 2, 3, 5, 6, 7

[2] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014. 1, 2, 5

[3] P. Fischer, A. Dosovitskiy, E. Ilg, P. Hausser, C. H. amd V. Golkov, P. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with CNNs. In *ICCV*, 2015. 1

[4] R. Garg, V. K. BG, G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *ECCV*, 2016. 1, 2

[5] C. Hane, L. Ladicky, and M. Pollefeys. Direction matters: Depth estimation with a surface normal classifier. In *CVPR*, 2015. 2

[6] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. In *SIGGRAPH*, 2005. 1, 2

[7] K. Karsch, C. Liu, and S. Kang. Depth extraction from video using non-parametric sampling. In *ECCV*, 2012. 1, 2, 6

[8] S. Kim, K. Park, K. Sohn, and S. Lin. Unified depth prediction and intrinsic image decomposition from a single image via joint convolutional neural fields. In *ECCV*, 2016. 1, 2

[9] L. Ladick, J. Shi, and M. Pollefeys. Pulling things out of perspective. In *CVPR*, 2014. 2, 6

[10] B. Li, C. Shen, Y. Dai, A. Hengel, and M. He. Depth and surface normal estimation from monocular images using re-

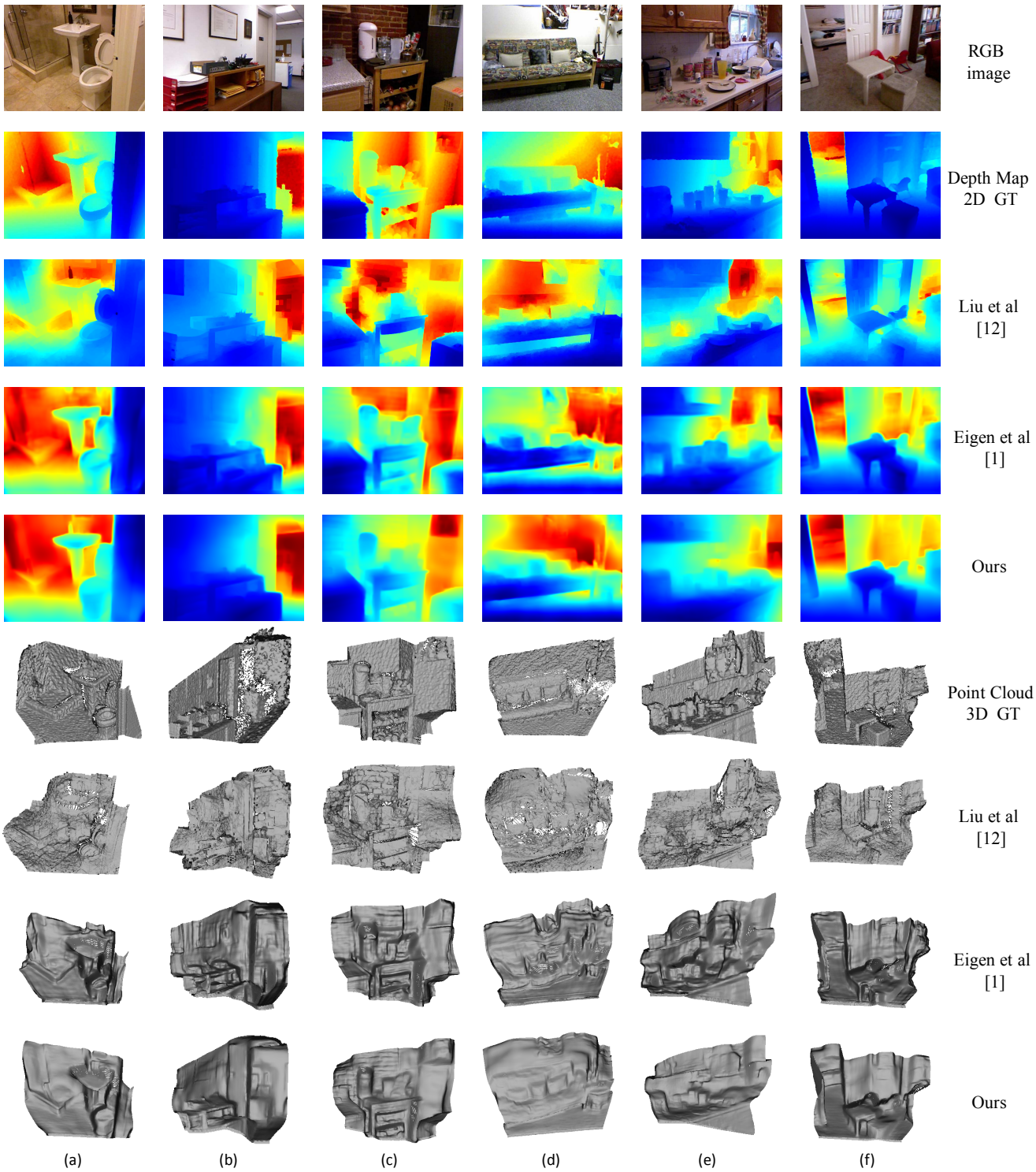


Figure 5. 2D depth map and 3D reconstruction comparisons. Note the colour values in depth map is individually scaled. Figure best viewed in colour.

gression on deep features and hierarchical CRFs. In *CVPR*, 2015. 1, 2, 3, 5, 6

[11] B. Liu, S. Gould, and D. Koller. Single image depth estimation from predicted semantic labels. In *CVPR*, 2010. 2

[12] F. Liu, C. Shen, and I. D. R. G. Lin. Learning depth from single monocular images using deep convolutional neural fields. In *TPAMI*, 2015. 1, 2, 5, 6

[13] M. Liu, M. Salzmann, and X. He. Discrete-continuous depth

estimation from a single image. In *CVPR*, 2014. 1, 2, 5, 6

[14] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

[15] A. Roy and S. Todorovic. Monocular depth estimation using neural regression forest. In *ICCV*, 2015. 1, 2

[16] A. Saxena, S. Chung, and A. Ng. Learning depth from single monocular images. In *NIPS*, 2005. 1, 2

[17] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3-d scene structure from a single still image. *TPAMI*, 2008. 1, 2

[18] N. Silberman, P. Kohli, D. Hoiem, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 5

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 3, 5

[20] A. Torralba and A. Oliva. Depth estimation from image structure. *TPAMI*, 2002. 1

[21] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. Yuille. Towards unified depth and semantic prediction from a single image. In *CVPR*, 2015. 1, 2

[22] Z. Zhang, A. G. Schwing, S. Fidler, and R. Urtasun. Monocular object instance segmentation and depth ordering with cnns. In *ICCV*, 2015. 1

[23] W. Zhuo, M. Salzmann, X. He, and M. Liu. Indoor scene structure analysis for single image depth estimation. In *CVPR*, 2015. 1, 2