

A Two-Streamed Network for Estimating Fine-Scaled Depth Maps from Single RGB Images

Jun Li^{1,2}, Reinhard Klein¹ and Angela Yao¹

¹University of Bonn, ²National University of Defense Technology

lij, rk, yao@cs.uni-bonn.de

Abstract

Estimating 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 2D 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 two-streamed CNN that predicts depth and depth gradients, which are then fused together into an accurate and detailed depth map. We also define a novel set loss over multiple images; by regularizing the estimation between a common set of images, the network is less prone to over-fitting and achieves better accuracy than competing methods. Experiments on the NYU Depth v2 dataset shows that our depth predictions are competitive with state-of-the-art and lead to faithful 3D projections.

1. Introduction

Estimating depth for common indoor scenes from monocular RGB images has widespread applications in scene understanding, depth-aware image editing or re-rendering, 3D modelling, 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.

The use of convolutional neural networks (CNNs) has greatly improved the accuracy of depth estimation techniques [7, 8, 14, 16, 17, 19, 23, 29]. Rather than coarsely approximating the depth of large structures such as walls and ceilings, state-of-the-art networks [7, 16] benefit from using pre-trained CNNs and can capture fine-scaled 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. Faithful reconstructions should be rich with

local structure; detailing becomes especially important in applications derived from the reconstructions such as object recognition and depth-aware image re-rendering and editing. Despite the impressive evaluation scores of recent works [7, 16] however, the estimated depth maps still suffer from artifacts at finer scales and have unsatisfactory alignments between surfaces. These distortions are especially prominent when projected into 3D (see Figure 1).

Other CNN-based end-to-end applications such as semantic segmentation [4, 21] and normal estimation [1, 7] face similar challenges of preserving local details. The repeated convolution and pooling operations are critical for capturing the entire image extent, but simultaneously shrink resolution and degrade the detailing. While up-convolution and feature map concatenation strategies [6, 16, 21, 22] have been proposed to improve resolution, output map boundaries often still fail to align with image boundaries. As such, optimization measures like bilateral filtering [2] or CRFs [4] yield further improvements.

It is with the goal of preserving detailing that we motivate our work on depth estimation. We want to benefit from the accuracy of CNNs, but avoid the degradation of resolution and detailing. First, we ensure network accuracy and generalization capability by introducing a novel set image loss. This loss is defined jointly over multiple images, where each image is a transformed version of an original image via standard data 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. Adding this regularizer greatly improves the depth accuracy and reduces RMS error by approximately 5%. As similar data augmentation approaches are also used in other end-to-end frameworks, *e.g.* for semantic segmentation and normal estimation, we believe that the benefits of the set loss will carry over to these applications as well.

We capture scene detailing by considering information contained in depth gradients. We postulate that local structure can be better encoded with first-order derivative terms than the absolute depth values. Perceptually, it is sharp

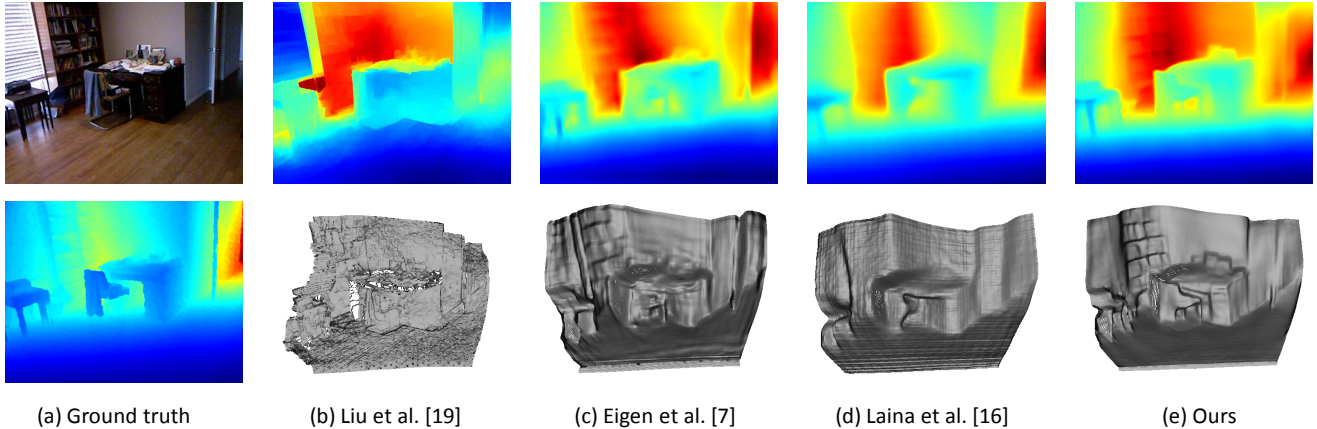


Figure 1. 3D projects from estimated depth maps of state-of-the-art methods (b-d), along with ground truth (a). Our estimated depth maps(e) are more accurate than state-of-the-art methods with fine-scaled details. Note that colours values of each depth map are individually scaled.

edges and corners which define an object and make it recognizable, rather than (correct) depth values (compare in Figure 4). As such, we think it’s better to represent a scene with both depth and depth gradients, and propose a fast-to-train two-streamed CNN to regress the depth and depth gradients (see Figure 2). In addition, we propose two possibilities for fusing the depth and depth gradients, one via a CNN, to allow for end-to-end training, and one via direct optimization. We summarize our contributions as follows:

- A novel set image loss with a regularizer that minimizes differences in estimated depth of related images; this loss makes better use of augmented data and promotes stronger network generalization, resulting in higher estimation accuracy.
- A joint representation of the 2.5D scene with depth and depth gradients; this representation captures local structures and fine detailing and is learned with a two-streamed network.
- Two methods for fusing depth and depth gradients into a final depth output, one via CNNs for end-to-end training and one via direct optimization; both methods yield depth maps which, when projected into 3D, have less distortion and are richer with structure and object detailing than competing state-of-the-art.

Representing the scene with with both depth and depth gradients is redundant, as one can be derived from the other. We show, however, that this redundancy offers explicit consideration for local detailing that is otherwise lost in the standard Euclidean loss on depth alone and/or with a simple consistency constraint in the loss. Our final depth output is accurate and clean with local detailing, with fewer artifacts than competing methods when projected into 3D.

2. Related Work

Depth estimation is a rich field of study and we discuss only the monocular methods. A key strategy in early works for handling depth ambiguity was to use strong assumptions and prior knowledge. For example, Saxena *et al.* [24, 25] devised a multi-scale MRF, but assumed that all scenes were horizontally aligned with the ground plane. Hoiem *et al.* [11], instead of predicting depth explicitly, estimated geometric structures for major image regions and composed simple 3D models to represent the scene.

Once RGB-D data could be collected from laser or depth cameras on a large scale, it became feasible to apply data-driven learning-based approaches [13, 20, 24, 25, 30]. Karsch *et al.* [13] 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.* [20] modelled image regions as superpixels and used discrete-continuous optimization for depth estimation and later integrated mid-level region features and global scene layout [30]. Others tried to improve depth estimations by exploiting semantic labels [9, 15, 18]. With hand-crafted features, however, the inferred depth maps are coarse and only approximate the global layout of a scene. Furthermore, they lack the finer details necessary for many applications in computer vision and graphics.

Deep learning has proven to be highly effective for depth estimation [7, 8, 14, 17, 19, 29, 23]. Liu *et al.* [19] combined CNNs and CRFs in a unified framework to 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.

More recent methods [3, 7, 16] have the harnessed the power of pre-trained CNNs in the form of fully convolutional networks [21]. The convolutional layers from net-

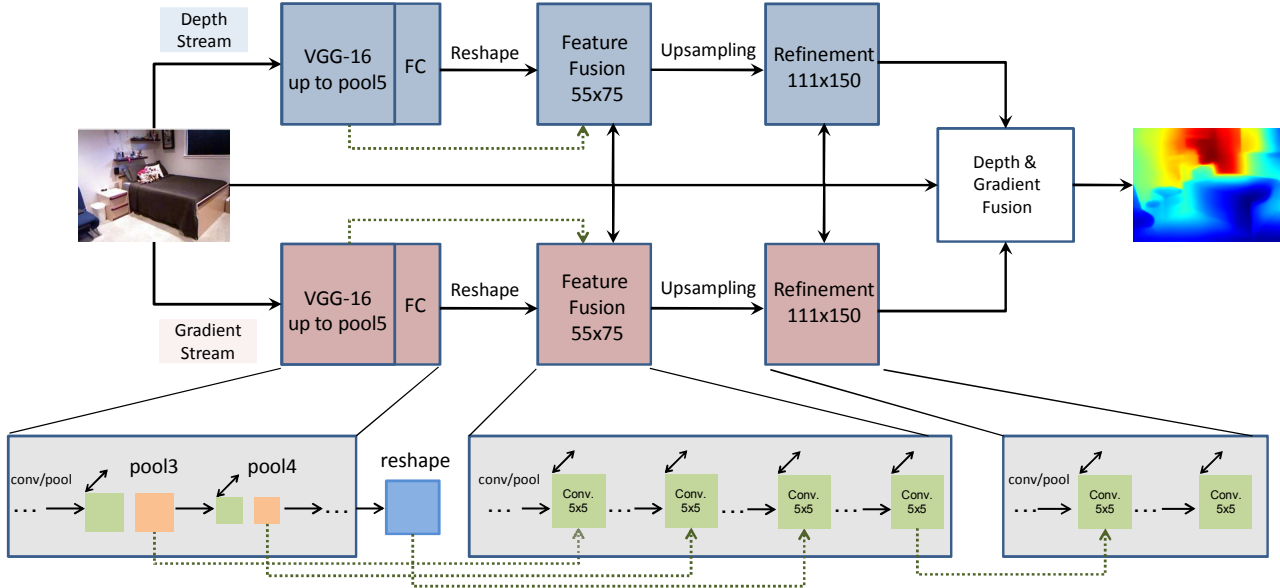


Figure 2. Our two-streamed depth estimation network architecture; the top stream (blue) estimates depth while the bottom (pink) estimates depth gradients. The dotted lines represent features fused from the VGG convolutional layers (see Section 3.1). The depth and depth gradients are then combined either via further convolution layers or directly with an optimization enforcing consistency between the depth and depth gradients. Figure is best viewed in colour.

works such as VGG [28] and ResNet [10] are fine-tuned, while the fully connected layers are re-learned from scratch to encode a spatial feature mapping of the scene. The learned map, however, is at a much lower resolution than the original input. To recover a high-resolution depth image, the feature mapping is then up-sampled [3, 7] or passed through up-convolution blocks [16]. Our network architecture follows a similar fully convolutional approach, and increases resolution via up-sampling. In addition, we add skip connections between the up-sampling blocks to better leverage intermediate outputs.

3. Learning

3.1. Network Architecture

Our network architecture, shown in Figure 2, follows a two-stream model; one stream regresses depth and the other depth gradients, both from an RGB input image. The two streams follow the same format: an image parsing block, flowed by a feature fusion block and finally a refinement block. The image parsing block consists of the convolutional layers of VGG-16 (up to *pool5*) and two fully connected layers. The output from the second fully connected layer is then reshaped into a $55 \times 75 \times D$ feature map to be passed onto the feature fusion block, where $D = 1$ for the depth stream and $D = 2$ for the gradient stream. In place of VGG-16, other pre-trained networks can be used as well for the image parsing block, *e.g.* VGG-19 or ResNet.

The feature fusion block consists of one 9×9 convolution

and pooling, followed by eight successive 5×5 convolutions without pooling. It takes as input a down-sampled RGB image and then fuses together features from the VGG convolutional layers and the image parsing block output. Specifically, the features maps from VGG *pool3* and *pool4* are fused at the input to the second and fourth convolutional layers respectively, while the output of the image parsing block is fused at the input to the sixth convolutional layer, all with skip layer connections. The skip connections for the VGG features have a 5×5 convolution and a 2x or 4x up-sampling to match the working 55×75 feature map size; the skip connection from the image parsing block is a simple concatenation. As noted by other image-to-image mapping works [12, 21, 22], the skip connections provide a convenient way to share hierarchical information and we find that this also results in much faster network training convergence. The output of the feature fusion block is a coarse $55 \times 75 \times D$ depth or depth gradient map.

The refinement block, similar to the feature fusion block, consists of one 9×9 convolution and pooling and five 5×5 convolutions without pooling. It takes as input a down-sampled RGB image and then fuses together a bilinearly up-sampled output of the feature fusion block via a skip connection (concatenation) to the third convolutional layer. The working map size in this block is 111×150 , with the output being depth or gradient maps at this higher resolution.

The depth and gradient fusion block brings together the

depth and depth gradient estimates from the two separate streams into a single coherent depth estimate. We propose two possibilities, one with convolutional processing in an end-to-end network, and one via a numerical optimization. The two methods are explained in detail in Section 3.3. We refer the reader to the supplementary material for specifics on the layers, filter sizes, and learning rates.

3.2. Set Image Loss

For many machine learning problems, it has become 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 loss is applied 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 formulate the set loss as follows. We start by defining the pixel-wise l_2 difference between two depth maps D_1 and D_2 :

$$l_2(D_1, D_2) = \frac{1}{n} \sum_p (D_1^p - D_2^p)^2, \quad (1)$$

where p is a pixel index, to be summed over n valid depth pixels¹. A simple error measure comparing an estimated depth map D_i and its corresponding ground truth D_{gt_i} can then be defined as $l_2(D_i, D_{gt_i})$.

Now consider an image set $\{I, f_1(I), f_2(I) \dots f_{N-1}(I)\}$, of size N , where I is the original image and f are the data augmentation transformations such as colour adjustment, flipping, rotation, skew, *etc.* For a set of images, the set loss L_{set} is given by

$$L_{set} = L_{single} + \lambda \cdot \Omega_{set}, \quad (2)$$

which considers the estimation error of each image in the set as independent samples in L_{single} , along with a regularization term Ω_{set} based on the entire set of transformed images, with λ as a weighting parameter between the two.

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

$$L_{single} = \frac{1}{N} \sum_{i=1}^N l_2(D_i, D_{gt_i}). \quad (3)$$

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

The regularization term Ω_{set} is defined as the l_2 difference between estimates within an image set:

$$\Omega_{set} = \frac{2}{N(N-1)} \sum_{i=1}^N \sum_{j>i}^N l_2(D_i, g_{ij}(D_j)). \quad (4)$$

Here, D_i and D_j are the depth estimates of any two images I_i and I_j in the image set. g is a mapping between their 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 and f^{-1} are simply identity functions. Ω_{set} acts as a consistency measure which encourages all the depth maps of an image set to be the same after accounting for the transformations. This regularization term has loose connections to tangent propagation [27], which also has the aim of encouraging invariance to input transformations. Instead of explicit computing the tangent vectors, we transform augmented samples back into a common reference before doing standard back-propagation.

3.3. Depth and Depth Gradient Estimation

To learn the network in the depth stream, we use our proposed set loss as defined in Equations 2 to 4. For learning the network in the depth gradient stream, we use the same formulation, but modify the pixel-wise difference for two gradient maps G_1 and G_2 , substituting l_{2g} for l_2 :

$$l_{2g}(G_1, G_2) = \frac{1}{n} \sum_p (G_{1x}^p - G_{2x}^p)^2 + (G_{1y}^p - G_{2y}^p)^2, \quad (5)$$

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.

Fusion in an End-to-end Network We propose two possibilities for fusing the outputs of the depth and gradient streams into a final depth output. The first is via a combination block, with the same architecture as the refinement block. It takes as input the RGB image and fuses together the depth estimates and gradient estimates via skip connections (concatenations) as inputs to the third convolutional layer. We use the following combined loss L_{comb} that maintains depth accuracy and gradient consistency:

$$L_{comb} = L_{set} + \frac{1}{N} \sum_{i=1}^N l_{2g}(\nabla D_i, G_{est_i}), \quad (6)$$

where ∇D_i indicates application of the gradient operator on depth map D_i . In this combined loss, the first term L_{set} is based only on depth, while the second term enforces consistency between the gradient of the final depth and estimated gradients with the same l_{2g} pixel-wise difference from Equation 5.

Fusion via Optimization Alternatively, as optimization measures have also shown to be highly effective in improving output map detailing [2, 4], we directly estimate an optimal depth D^* based on the following minimization:

$$D^* = \underset{D}{\operatorname{argmin}} \sum_{p=1}^n \phi(D^p - D_{est}^p) + \omega \sum_p^n [\phi(\nabla_x D^p - G_x^p) + \phi(\nabla_y D^p - G_y^p)], \quad (7)$$

where D_{est} is the estimated depth and G_x and G_y are the estimated gradients in x and y . $\phi(x)$ acts as a robust L_1 measure, *i.e.* $\phi(x) = \sqrt{x^2 + \epsilon}$, $\epsilon = 10^{-4}$; ∇_x , ∇_y are x , y gradient operators on depth D (we use filters $[-1, 0, 1]$, $[-1, 0, 1]^T$) and p is the pixel index summed over the n valid pixels. We solve for D with iteratively re-weighted least squares using the implementation provided by Karsch [13].

3.4. Training Strategy

We apply the same implementation for the networks in both the depth and the gradient streams. With the exception of the VGG convolutional layers, the fully connected layers and all layers in the feature fusion, refinement and combination block are initialized randomly.

The two streams are initially trained individually, each with a two-step procedure. First, the image parsing and feature fusion blocks are trained with a loss on the depth and depth gradients. These blocks are then fixed, while in the refinement blocks are trained in a second step. For each step, the same set loss with the appropriate pixel differences (see Equations 2, 1, 5) is used, albeit with different map resolutions (55×75 after feature fusion, 111×75 after refinement). Training ends here if the optimization-based fusion is applied. Otherwise, for the end-to-end network fusion, the image parsing, feature fusion and refinement blocks are fixed while the fusion block is trained using the combined loss (Equation 6). A final fine-tuning is applied to all the blocks of both streams jointly based on this combined loss.

The network is fast to train; only 2 to 3 epochs are required for convergence (see Figure 3). During the first 20K iterations, we stabilize training with gradient clipping. For fast convergence, we use a batch size of 1; 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.

The regularization constants λ and ω are set to 1 and 10 respectively. Preliminary experiments showed that different values of λ which controls the extent of the set image regularizer does not affect the resulting accuracy, although a larger λ does slow down network convergence. ω , controlling the extent of the gradients in the gradient optimization,

was set by a validation set; a larger ω over-emphasizes artifacts in the gradient estimates and leads to less accurate depth maps.

4. Experimentation

4.1. Dataset & Evaluation

We use the NYU Depth v2 dataset [26] with the standard scene splits; from the 249 training scenes, we extract $\sim 220k$ training images. RGB images are down-sampled by half and then cropped to 232×310 to remove 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 the two fusion methods on the 654 NYU Depth v2 [26] test images. Since our depth output is 111×150 and a lower resolution than the original NYU Depth images, we bilinearly up-sample our depth map (4x) and fill in missing borders with a cross-bilateral filter, similar to previous methods [7, 17, 19, 20]. We 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): $\sqrt{\frac{1}{T} \sum_i^T (d_i^{gt} - d_i)^2}$,
- thresholded accuracy: percentage of d_i such that $\max(d_i^{gt}/d_i, d_i/d_i^{gt}) = \delta < \text{threshold}$.

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(%) $\delta < \text{threshold}$ are better.

We make a qualitative comparison by projecting the estimated 2D depth maps into a 3D point cloud. The 3D projections are computed using the Kinect camera projection matrix and the resulting point cloud is rendered with lighting. Representative samples are shown in Figure 4; the reader is referred to the Supplementary Material for more results over the test set.

4.2. Depth Estimation Baselines

The accuracy of our depth estimates is compared with other methods in Table 1. We consider as our baseline the accuracy of only the depth stream with a VGG-16 base network, without adding gradients (L_{single} , depth only). This baseline already outperforms [16] (VGG-16) and is comparable to [7] (VGG-16). With the set loss, however (L_{set} , depth only), we surpass [7] with more accurate depth estimates, especially in terms of rms-error and thresholded accuracy with $\delta < 1.25$.

Method	Base Network	rel	log10	rms	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Karsch <i>et al.</i> [13]	-	0.35	0.131	1.2	-	-	-
Liu <i>et al.</i> [20]	-	0.335	0.127	1.06	-	-	-
Ladicky <i>et al.</i> [15]	-	-	-	-	0.542	0.829	0.941
Li <i>et al.</i> [17]	-	0.232	0.094	0.821	0.621	0.886	0.968
Liu <i>et al.</i> [19]	-	0.213	0.087	0.759	0.650	0.906	0.976
Eigen <i>et al.</i> [7]	VGG-16	0.158	-	0.641	0.769	0.950	0.988
Laina <i>et al.</i> [16]	VGG-16	0.194	0.083	0.790	0.629	0.889	0.971
Chakrabarti <i>et al.</i> [3]	VGG-19	0.149	-	0.620	0.806	0.958	0.987
Laina <i>et al.</i> [16]	ResNet-50	0.127	0.055	0.573	0.811	0.953	0.988
L_{single} , depth only	VGG-16	0.161	0.068	0.640	0.765	0.950	0.988
L_{set} , depth only	VGG-16	0.153	0.065	0.617	0.786	0.954	0.988
L_{set} , depth + bilateral filtering	VGG-16	0.152	0.065	0.621	0.785	0.954	0.988
L_{set} , depth + gradients, end-to-end	VGG-16	0.153	0.064	0.615	0.788	0.954	0.988
L_{set} , depth + gradients, optimization	VGG-16	0.152	0.064	0.611	0.789	0.955	0.988
L_{set} , depth + gradients, optimization	VGG-19	0.146	0.063	0.617	0.795	0.958	0.991
L_{set} , depth + gradients, optimization	ResNet-50	0.143	0.063	0.635	0.788	0.958	0.991

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 $\delta < \text{threshold}$ are better.

Current state-of-the-art results are achieved with fully convolutional approaches [3, 7, 16]. The general trend is that deeper base networks (VGG-19, ResNet-50 vs. VGG-16) leads to higher depth accuracy. We observe a similar trend in our results, though improvements are not always consistent. We achieve some gains with VGG-19 over VGG-16. However, unlike [16], we found little gains with ResNet-50.

4.3. Fused Depth and Depth Gradients

Fusing depth estimates together with depth gradients achieves similar results as the optimization, both quantitatively and qualitatively. When the depth maps are projected into 3D (see Figure 4), there is little difference between the two fusion methods. When comparing with [16]’s ResNet-50 results, which are significantly more accurate, one sees that [16]’s 3D projections are more distorted. In fact, many structures, such as the shelves in Figure 4(a), the sofa in (b) or the pillows in (b,d) are unidentifiable. Furthermore, the entire projected 3D surface seems to suffer from grid-like artifacts, possibly due to their up-projection methodology. On the other hand, the projections of [3] are much cleaner and detailed, even though their method also reports lower accuracy than [16]. As such, it is our conclusion that the current numerical evaluation measures are poor indicators of detail preservation. This is expected, since the gains from detailing have little impact on the numerical accuracy measures. Instead, differences are more salient qualitatively, especially in the 3D projection.

Compared to [7], our 3D projections 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 Figure 4(b,c), [7] has heavy artifacts in highly textured areas, *e.g.* the picture on the wall of (b), the windows in (c) and in regions with strong reflections such as the cabinet in (c). Our results are robust to these difficulties and give a more faithful 3D projections of the underlying objects in the scene.

At a first glance, one may think that jointly representing the scene with both depth and depth gradients simply has a smoothing effect. While the fused results are definitely smoother, no smoothing operations, 2D or 3D, can recover non-existent detail. When applying a 2D bilateral filter with 0.1m range and 10×10 spatial Gaussian kernel (L_{set} depth + bilateral filtering), we find little difference in the numerical measures and still some losses in detailing (see Fig.4)

4.4. Set Loss & Data Augmentation

Our proposed set image loss has a large impact in improving the estimated depth accuracy. For the reported results in Table 1, we used three images in the image set: I , $\text{flip}(I)$ and $\text{colour}(I)$. 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.

4.5. Training Convergence and Timing

We show the convergence behaviour of our network for the joint training of the image parsing and feature fusion blocks Figure 3. Errors decrease faster with a batch size of

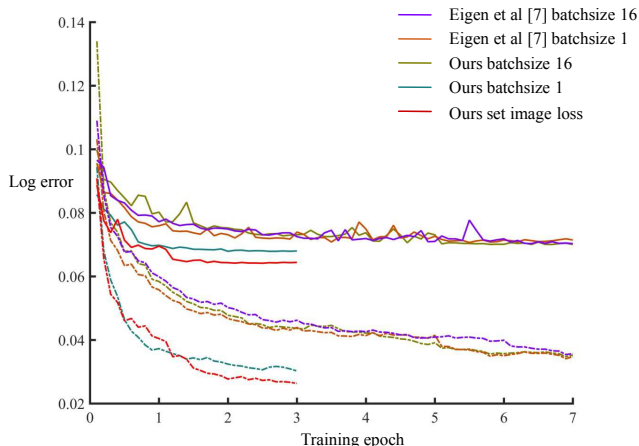


Figure 3. A comparison of the \log_{10} training and test errors between our proposed network and [7]. For clarity, we plot the \log error at every 0.1 epochs, and show only the first 7 epochs, even though the method of [7] has not yet converged. The dashed lines denote training error, and solid lines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

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 4.4 and observe that errors 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 connection. In comparison, the network architecture of [7] 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. 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.

5. Discussion and Conclusion

We have proposed a fast-to-train multi-streamed CNN architecture for accurate depth estimation. To predict accurate and detailed depth maps, we introduced three 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 represent a scene with a joint depth and depth gradient representation, for which we learn with a two-streamed network, to preserve the fine detailing in a scene. Finally, we propose two methods, one CNN-based and one optimization-based, for fusing the depth and gradient estimates into a final depth output. Experiments on the NYU Depth v2 dataset shows that our depth predictions are

not only competitive with state-of-the-art but also lead to 3D projections that are more accurate and richer with details.

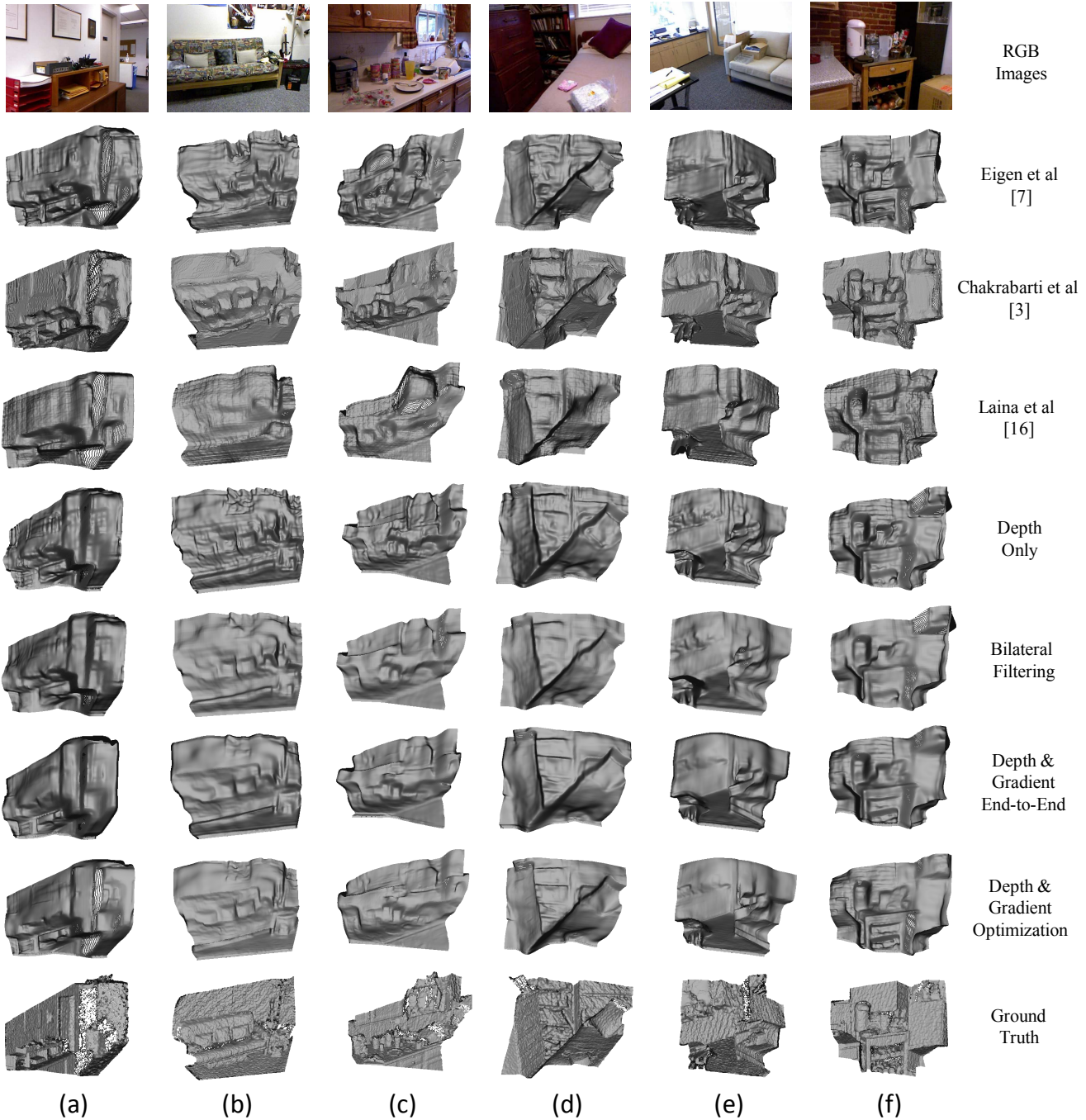
Looking at our experimental results as well as the results of state-of-the-art methods [3, 7, 16], it becomes clear that the current numerical metrics used for evaluating estimated depth are not always consistent. Such inconsistencies become even more prominent when the depth maps are projected into 3D. Unfortunately, the richness of a scene is often qualified by clean structural detailing which are difficult to capture numerically and which makes it in turn difficult to design appropriate loss or objective functions. Alternatively, one may look to applications using image-estimated depths as input, such as 3D model retrieval or scene-based relighting, though such an indirect evaluation may also introduce other confounding factors.

Our method generates accurate and rich 3D projections, but the outputs are still only 111×150 , whereas the original input is 427×561 . Like many end-to-end applications, we work at lower-than-original resolutions to trade off the number of network parameters versus the amount of training data. While depth estimation requires no labels, the main bottleneck is the variation in scenes. The training images of NYU Depth v2 are derived from videos of only 249 scenes. The small training dataset size may explain why our set loss with its regularization term has such a strong impact. As larger datasets are introduced [5], it may become feasible to work at higher resolutions. Finally, 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 further improve the depth estimates.

Acknowledgements: The authors would like to thank NVidia for their GPU donation to enable this research.

References

- [1] A. Bansal, B. Russell, and A. Gupta. Marr revisited: 2D-3D alignment via surface normal prediction. In *CVPR*, 2016. 1
- [2] J. T. Barron and B. Poole. The fast bilateral solver. *ECCV*, 2016. 1, 5
- [3] A. Chakrabarti, J. Shao, and G. Shakhnarovich. Depth from a single image by harmonizing overcomplete local network predictions. In *NIPS*, 2016. 2, 3, 6, 7, 8
- [4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *arXiv preprint arXiv:1606.00915*, 2016. 1, 5
- [5] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *CVPR*, 2017. 7
- [6] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox.



RMS error	a	b	c	d	e	f
Eigen <i>et al.</i> [7]	0.795	0.154	0.377	0.204	0.452	0.253
Chakrabarti <i>et al.</i> [3]	0.558	0.236	0.223	0.208	0.400	0.304
Laina <i>et al.</i> [16]	0.824	0.237	0.306	0.132	0.344	0.159
Depth Only	0.715	0.259	0.246	0.127	0.335	0.202
Bilateral Filtering	0.715	0.259	0.245	0.126	0.334	0.201
Depth & Gradient (End-to-End)	0.719	0.261	0.249	0.130	0.336	0.192
Depth & Gradient (Optimization)	0.715	0.265	0.248	0.124	0.338	0.185

Figure 4. Comparison of example 3D projections with corresponding RMS errors using VGG-16 as the base network. Please see supplementary materials for more results. The four variations of our proposed method all report similar RMS values, though differences are more noticeable in the 3D projection. From examples (b) and (f), where [16] reports the lower RMS, one can see that a numerical measure such as RMS is not always indicative of estimated depth quality.

- FlowNet: learning optical flow with CNNs. In *ICCV*, 2015. 1
- [7] 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, 8
- [8] R. Garg, B. V. Kumar, G. Carneiro, and I. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *ECCV*, 2016. 1, 2
- [9] C. Hane, L. Ladicky, and M. Pollefeys. Direction matters: Depth estimation with a surface normal classifier. In *CVPR*, 2015. 2
- [10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3
- [11] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. In *SIGGRAPH*, 2005. 2
- [12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2016. 3
- [13] K. Karsch, C. Liu, and S. Kang. Depth extraction from video using non-parametric sampling. In *ECCV*, 2012. 2, 5, 6
- [14] 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
- [15] L. Ladick, J. Shi, and M. Pollefeys. Pulling things out of perspective. In *CVPR*, 2014. 2, 6
- [16] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *3DV*, 2016. 1, 2, 3, 5, 6, 7, 8
- [17] B. Li, C. Shen, Y. Dai, A. Hengel, and M. He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs. In *CVPR*, 2015. 1, 2, 5, 6
- [18] B. Liu, S. Gould, and D. Koller. Single image depth estimation from predicted semantic labels. In *CVPR*, 2010. 2
- [19] 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
- [20] M. Liu, M. Salzmann, and X. He. Discrete-continuous depth estimation from a single image. In *CVPR*, 2014. 2, 5, 6
- [21] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1, 2, 3
- [22] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 1, 3
- [23] A. Roy and S. Todorovic. Monocular depth estimation using neural regression forest. In *ICCV*, 2015. 1, 2
- [24] A. Saxena, S. Chung, and A. Ng. Learning depth from single monocular images. In *NIPS*, 2005. 2
- [25] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3-d scene structure from a single still image. *TPAMI*, 2008. 2
- [26] N. Silberman, P. Kohli, D. Hoiem, and R. Fergus. Indoor segmentation and support inference from rgb-d images. In *ECCV*, 2012. 5
- [27] P. Simard and J. Denker. Tangent prop-a formalism for specifying selected invariances in an adaptive net. In *NIPS*, 1992. 4
- [28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 3
- [29] 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
- [30] W. Zhuo, M. Salzmann, X. He, and M. Liu. Indoor scene structure analysis for single image depth estimation. In *CVPR*, 2015. 2