

# Unsupervised Monocular Depth Estimation with Left-Right Consistency

Clément Godard

Oisín Mac Aodha

Gabriel J. Brostow

University College London

<http://visual.cs.ucl.ac.uk/pubs/monoDepth/>

## Abstract

*Learning based methods have shown very promising results for the task of depth estimation in single images. However, most existing approaches treat depth prediction as a supervised regression problem and as a result, require vast quantities of corresponding ground truth depth data for training. Just recording quality depth data in a range of environments is a challenging problem. In this paper, we innovate beyond existing approaches, replacing the use of explicit depth data during training with easier-to-obtain binocular stereo footage.*

*We propose a novel training objective that enables our convolutional neural network to learn to perform single image depth estimation, despite the absence of ground truth depth data. Exploiting epipolar geometry constraints, we generate disparity images by training our network with an image reconstruction loss. We show that solving for image reconstruction alone results in poor quality depth images. To overcome this problem, we propose a novel training loss that enforces consistency between the disparities produced relative to both the left and right images, leading to improved performance and robustness compared to existing approaches. Our method produces state of the art results for monocular depth estimation on the KITTI driving dataset, even outperforming supervised methods that have been trained with ground truth depth.*

## 1. Introduction

Depth estimation from images has a long history in computer vision. Fruitful approaches have relied on structure from motion, shape-from-X, binocular, and multi-view stereo. However, most of these techniques rely on the assumption that multiple observations of the scene of interest are available. These can come in the form of multiple viewpoints, or observations of the scene under different lighting conditions. To overcome this limitation, there has recently been a surge in the number of works that pose the task of monocular depth estimation as a supervised learning problem [32, 10, 36]. These methods attempt to directly predict the depth of each pixel in an image using models that have been trained offline on large collections of ground truth depth data. While these methods have enjoyed great success, to date they

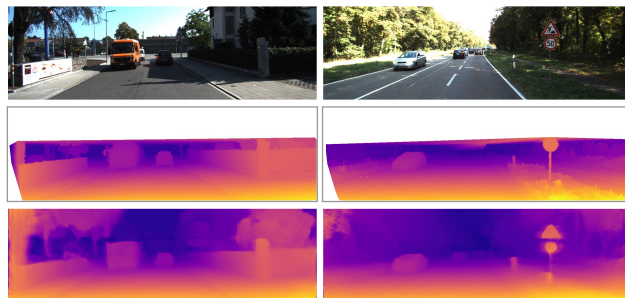


Figure 1. Our depth prediction results on KITTI 15. Top to bottom: input image, ground truth disparities, and our result. Our method is able to estimate depth for thin structures such as street signs and poles.

have been restricted to scenes where large image collections and their corresponding pixel depths are available.

Understanding the shape of a scene from a single image, independent of its appearance, is a fundamental problem in machine perception. There are many applications such as synthetic object insertion in computer graphics [29], synthetic depth of field in computational photography [3], grasping in robotics [34], using depth as a cue in human body pose estimation [48], robot assisted surgery [49], and automatic 2D to 3D conversion in film [53]. Accurate depth data from one or more cameras is also crucial for self-driving cars, where expensive laser-based systems are often used.

Humans perform well at monocular depth estimation by exploiting cues such as perspective, scaling relative to the known size of familiar objects, appearance in the form of lighting and shading and occlusion [24]. This combination of both top-down and bottom-up cues appears to link full scene understanding with our ability to accurately estimate depth. In this work, we take an alternative approach and treat automatic depth estimation as an image reconstruction problem during training. Our fully convolutional model does not require any depth data, and is instead trained to synthesize depth as an intermediate. It learns to predict the pixel-level correspondence between pairs of rectified stereo images that have a known camera baseline. There are some existing methods that also address the same problem, but with several limitations. For example they are not fully differentiable, making training suboptimal [16], or have image formation models that do

not scale to large output resolutions [53]. We improve upon these methods with a novel training objective and enhanced network architecture that significantly increases the quality of our final results. An example result from our algorithm is illustrated in Fig. 1. Our method is fast and only takes on the order of 35 milliseconds to predict a dense depth map for a  $512 \times 256$  image on a modern GPU. Specifically, we propose the following contributions:

- 1) A network architecture that performs end-to-end unsupervised monocular depth estimation with a novel training loss that enforces left-right depth consistency inside the network.
- 2) An evaluation of several training losses and image formation models highlighting the effectiveness of our approach.
- 3) In addition to showing state of the art results on a challenging driving dataset, we also show that our model generalizes to three different datasets, including a new outdoor urban dataset that we have collected ourselves, which we make openly available.

## 2. Related Work

There is a large body of work that focuses on depth estimation from images, either using pairs [46], several overlapping images captured from different viewpoints [14], temporal sequences [44], or assuming a fixed camera, static scene, and changing lighting [52, 2]. These approaches are typically only applicable when there is more than one input image available of the scene of interest. Here we focus on works related to monocular depth estimation, where there is only a single input image, and no assumptions about the scene geometry or types of objects present are made.

### Learning-Based Stereo

The vast majority of stereo estimation algorithms have a data term which computes the similarity between each pixel in the first image and every other pixel in the second image. Typically the stereo pair is rectified and thus the problem of disparity (*i.e.* scaled inverse depth) estimation can be posed as a 1D search problem for each pixel. Recently, it has been shown that instead of using hand defined similarity measures, treating the matching as a supervised learning problem and training a function to predict the correspondences produces far superior results [54, 31]. It has also been shown that posing this binocular correspondence search as a multi-class classification problem has advantages both in terms of quality of results and speed [38]. Instead of just learning the matching function, Mayer et al. [39] introduced a fully convolutional [47] deep network called DispNet that directly computes the correspondence field between two images. At training time, they attempt to directly predict the disparity for each pixel by minimizing a regression training loss. DispNet has a similar architecture to their previous end-to-end deep optical flow network [12].

The above methods rely on having large amounts of accurate ground truth disparity data and stereo image pairs at training time. This type of data can be difficult to obtain for real world

scenes, so these approaches typically use synthetic data for training. Synthetic data is becoming more realistic, *e.g.* [15], but still requires the manual creation of new content for every new application scenario.

### Supervised Single Image Depth Estimation

Single-view, or monocular, depth estimation refers to the problem setup where only a single image is available at test time. Saxena et al. [45] proposed a patch-based model known as Make3D that first over-segments the input image into patches and then estimates the 3D location and orientation of local planes to explain each patch. The predictions of the plane parameters are made using a linear model trained offline on a dataset of laser scans, and the predictions are then combined together using an MRF. The disadvantage of this method, and other planar based approximations, *e.g.* [22], is that they can have difficulty modeling thin structures and, as predictions are made locally, lack the global context required to generate realistic outputs. Instead of hand-tuning the unary and pairwise terms, Liu et al. [36] use a convolutional neural network (CNN) to learn them. In another local approach, Ladicky et al. [32] incorporate semantics into their model to improve their per pixel depth estimation. Karsch et al. [28] attempt to produce more consistent image level predictions by copying whole depth images from a training set. A drawback of this approach is that it requires the entire training set to be available at test time.

Eigen et al. [10, 9] showed that it was possible to produce dense pixel depth estimates using a two scale deep network trained on images and their corresponding depth values. Unlike most other previous work in single image depth estimation, they do not rely on hand crafted features or an initial over-segmentation and instead learn a representation directly from the raw pixel values. Several works have built upon the success of this approach using techniques such as CRFs to improve accuracy [35], changing the loss from regression to classification [5], using other more robust loss functions [33], and incorporating strong scene priors in the case of the related problem of surface normal estimation [50]. Again, like the previous stereo methods, these approaches rely on having high quality, pixel aligned, ground truth depth at training time. We too perform single depth image estimation, but train with an added binocular color image, instead of requiring ground truth depth.

### Unsupervised Depth Estimation

Recently, a small number of deep network based methods for novel view synthesis and depth estimation have been proposed, which do not require ground truth depth at training time. Flynn et al. [13] introduced a novel image synthesis network called DeepStereo that generates new views by selecting pixels from nearby images. During training, the relative pose of multiple cameras is used to predict the appearance of a held-out nearby image. Then the most appropriate depths are selected to sample colors from the neighboring images, based on plane sweep volumes.

At test time, image synthesis is performed on small overlapping patches. As it requires several nearby posed images at test time DeepStereo is not suitable for monocular depth estimation.

The Deep3D network of Xie et al. [53] also addresses the problem of novel view synthesis, where their goal is to generate the corresponding right view from an input left image (*i.e.* the source image) in the context of binocular pairs. Again using an image reconstruction loss, their method produces a distribution over all the possible disparities for each pixel. The resulting synthesized right image pixel values are a combination of the pixels on the same scan line from the left image, weighted by the probability of each disparity. The disadvantage of their image formation model is that increasing the number of candidate disparity values greatly increases the memory consumption of the algorithm, making it difficult to scale their approach to bigger output resolutions. In this work, we perform a comparison to the Deep3D image formation model, and show that our algorithm produces superior results.

Closest to our model in spirit is the concurrent work of Garg et al. [16]. Like Deep3D and our method, they train a network for monocular depth estimation using an image reconstruction loss. However, their image formation model is not fully differentiable. To compensate, they perform a Taylor approximation to linearize their loss resulting in an objective that is more challenging to optimize. Similar to other recent work, *e.g.* [43, 56, 57], our model overcomes this problem by using bilinear sampling [27] to generate images, resulting in a fully (sub-)differentiable training loss.

We propose a fully convolutional deep neural network loosely inspired by the supervised DispNet architecture of Mayer et al. [39]. By posing monocular depth estimation as an image reconstruction problem, we can solve for the disparity field without requiring ground truth depth. However, only minimizing a photometric loss can result in good quality image reconstructions but poor quality depth. Among other terms, our fully differentiable training loss includes a left-right consistency check to improve the quality of our synthesized depth images. This type of consistency check is commonly used as a post-processing step in many stereo methods, *e.g.* [54], but we incorporate it directly into our network.

### 3. Method

This section describes our single image depth prediction network. We introduce a novel depth estimation training loss, featuring an inbuilt left-right consistency check, which enables us to train on image pairs without requiring supervision in the form of ground truth depth.

#### 3.1. Depth Estimation as Image Reconstruction

Given a single image  $I$  at test time, our goal is to learn a function  $f$  that can predict the per-pixel scene depth,  $\hat{d} = f(I)$ . Most existing learning based approaches treat this as a supervised learning problem, where they have color input

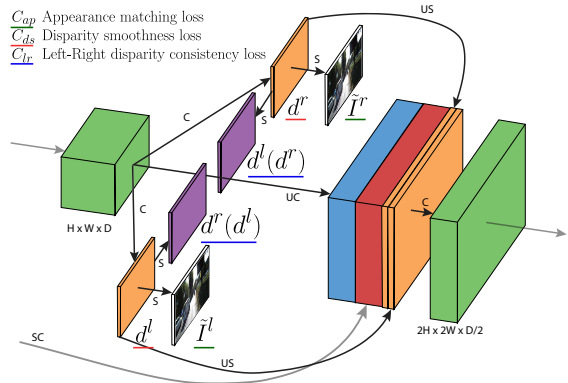


Figure 2. Our loss module outputs left and right disparity maps,  $d^l$  and  $d^r$ . The loss combines smoothness, reconstruction, and left-right disparity consistency terms. This same module is repeated at each of the four different output scales. C: Convolution, UC: Up-Convolution, S: Bilinear Sampling, US: Up-Sampling, SC: Skip Connection.

images and their corresponding target depth values at training. It is presently not practical to acquire such ground truth depth data for a large variety of scenes. Even expensive hardware, such as laser scanners, can be imprecise in natural scenes featuring movement and reflections. As an alternative, we instead pose depth estimation as an image reconstruction problem during training. The intuition here is that, given a calibrated pair of binocular cameras, if we can learn a function that is able to reconstruct one image from the other, then we have learned something about the 3D shape of the scene that is being imaged.

Specifically, at training time, we have access to two images  $I^l$  and  $I^r$ , corresponding to the left and right color images from a calibrated stereo pair, captured at the same moment in time. Instead of trying to directly predict the depth, we attempt to find the dense correspondence field  $d^r$  that, when applied to the left image, would enable us to reconstruct the right image. We will refer to the reconstructed image  $I^l(d^r)$  as  $\tilde{I}^r$ . Similarly, we can also estimate the left image given the right one,  $\tilde{I}^l = I^r(d^l)$ . Assuming that the images are rectified [19],  $d$  corresponds to the image disparity - a scalar value per pixel that our model will learn to predict. Given the baseline distance  $b$  between the cameras and the camera focal length  $f$ , we can then trivially recover the depth  $\hat{d}$  from the predicted disparity,  $\hat{d} = bf/d$ .

#### 3.2. Depth Estimation Network

At a high level, our network estimates depth by inferring the disparities that warp the left image to match the right one. The key insight of our method is that we can simultaneously infer both disparities (left-to-right and right-to-left), using only the left input image, and obtain better depths by enforcing them to be consistent with each other.

Our network generates the predicted image with backward mapping using a bilinear sampler, resulting in a fully differentiable image formation model. As illustrated in Fig. 3, naively learning to generate the right image by sampling from the left

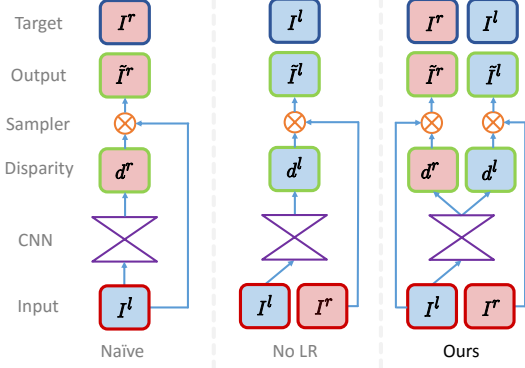


Figure 3. Sampling strategies for backward mapping. With naïve sampling the CNN produces a disparity map aligned with the target instead of the input. No LR corrects for this, but suffers from artifacts. Our approach uses the left image to produce disparities for both images, improving quality by enforcing mutual consistency.

one will produce disparities aligned with the right image (target). However, we want the output disparity map to align with the input left image, meaning the network has to sample from the right image. We could instead train the network to generate the left view by sampling from the right image, thus creating a left view aligned disparity map (**No LR** in Fig. 3). While this alone works, the inferred disparities exhibit ‘texture-copy’ artifacts and errors at depth discontinuities as seen in Fig. 5. We solve this by training the network to predict the disparity maps for both views by sampling from the opposite input images. This still only requires a single left image as input to the convolutional layers and the right image is only used during training (**Ours** in Fig. 3). Enforcing consistency between both disparity maps using this novel left-right consistency cost leads to more accurate results.

Our fully convolutional architecture is inspired by DispNet [39], but features several important modifications that enable us to train without requiring ground truth depth. Our network, is composed of two main parts - an encoder (from cnv1 to cnv7b) and decoder (from upcnv7), please see the supplementary material for a detailed description. The decoder uses skip connections [47] from the encoder’s activation blocks, enabling it to resolve higher resolution details. We output disparity predictions at four different scales (disp4 to disp1), which double in spatial resolution at each of the subsequent scales. Even though it only takes a single image as input, our network predicts two disparity maps at each output scale - left-to-right and right-to-left.

### 3.3. Training Loss

We define a loss  $C_s$  at each output scale  $s$ , forming the total loss as the sum  $C = \sum_{s=1}^4 C_s$ . Our loss module (Fig. 2) computes  $C_s$  as a combination of three main terms,

$$C_s = \alpha_{ap}(C_{ap}^l + C_{ap}^r) + \alpha_{ds}(C_{ds}^l + C_{ds}^r) + \alpha_{lr}(C_{lr}^l + C_{lr}^r), \quad (1)$$

where  $C_{ap}$  encourages the reconstructed image to appear similar to the corresponding training input,  $C_{ds}$  enforces

smooth disparities, and  $C_{lr}$  prefers the predicted left and right disparities to be consistent. Each of the main terms contains both a left and a right image variant, but only the left image is fed through the convolutional layers.

Next, we present each component of our loss in terms of the left image (e.g.  $C_{ap}^l$ ). The right image versions, e.g.  $C_{ap}^r$ , require to swap left for right and to sample in the opposite direction.

**Appearance Matching Loss** During training, the network learns to generate an image by sampling pixels from the opposite stereo image. Our image formation model uses the image sampler from the spatial transformer network (STN) [27] to sample the input image using a disparity map. The STN uses bilinear sampling where the output pixel is the weighted sum of four input pixels. In contrast to alternative approaches [16, 53], the bilinear sampler used is locally fully differentiable and integrates seamlessly into our fully convolutional architecture. This means that we do not require any simplification or approximation of our cost function.

Inspired by [55], we use a combination of an  $L1$  and single scale SSIM [51] term as our photometric image reconstruction cost  $C_{ap}$ , which compares the input image  $I_{ij}^l$  and its reconstruction  $\tilde{I}_{ij}^l$ , where  $N$  is the number of pixels,

$$C_{ap}^l = \frac{1}{N} \sum_{i,j} \alpha \frac{1 - \text{SSIM}(I_{ij}^l, \tilde{I}_{ij}^l)}{2} + (1 - \alpha) \|I_{ij}^l - \tilde{I}_{ij}^l\|. \quad (2)$$

Here, we use a simplified SSIM with a  $3 \times 3$  block filter instead of a Gaussian, and set  $\alpha = 0.85$ .

**Disparity Smoothness Loss** We encourage disparities to be locally smooth with an  $L1$  penalty on the disparity gradients  $\partial d$ . As depth discontinuities often occur at image gradients, similar to [21], we weight this cost with an edge-aware term using the image gradients  $\partial I$ ,

$$C_{ds}^l = \frac{1}{N} \sum_{i,j} |\partial_x d_{ij}^l| e^{-\|\partial_x I_{ij}^l\|} + |\partial_y d_{ij}^l| e^{-\|\partial_y I_{ij}^l\|}. \quad (3)$$

**Left-Right Disparity Consistency Loss** To produce more accurate disparity maps, we train our network to predict both the left and right image disparities, while only being given the left view as input to the convolutional part of the network. To ensure coherence, we introduce an  $L1$  left-right disparity consistency penalty as part of our model. This cost attempts to make the left-view disparity map be equal to the *projected* right-view disparity map,

$$C_{lr}^l = \frac{1}{N} \sum_{i,j} |d_{ij}^l - d_{ij+d_{ij}^l}^r|. \quad (4)$$

Like all the other terms, this cost is mirrored for the right-view disparity map and is evaluated at all of the output scales.

| Method                 | Dataset | Abs Rel      | Sq Rel       | RMSE         | RMSE log     | $Dl$ -all     | $\delta < 1.25$ | $\delta < 1.25^2$ | $\delta < 1.25^3$ |
|------------------------|---------|--------------|--------------|--------------|--------------|---------------|-----------------|-------------------|-------------------|
| Ours with Deep3D [53]  | K       | 0.412        | 16.37        | 13.693       | 0.512        | 66.85         | 0.690           | 0.833             | 0.891             |
| Ours with Deep3Ds [53] | K       | 0.151        | 1.312        | 6.344        | 0.239        | 59.64         | 0.781           | 0.931             | 0.976             |
| Ours No LR             | K       | 0.123        | 1.417        | 6.315        | 0.220        | 30.318        | 0.841           | 0.937             | 0.973             |
| Ours                   | K       | 0.124        | 1.388        | 6.125        | 0.217        | 30.272        | 0.841           | 0.936             | 0.975             |
| Ours                   | CS      | 0.699        | 10.060       | 14.445       | 0.542        | 94.757        | 0.053           | 0.326             | 0.862             |
| Ours                   | CS + K  | 0.104        | 1.070        | 5.417        | 0.188        | 25.523        | 0.875           | 0.956             | 0.983             |
| Ours pp                | CS + K  | 0.100        | 0.934        | 5.141        | 0.178        | 25.077        | 0.878           | 0.961             | <b>0.986</b>      |
| Ours resnet pp         | CS + K  | <b>0.097</b> | <b>0.896</b> | <b>5.093</b> | <b>0.176</b> | <b>23.811</b> | <b>0.879</b>    | <b>0.962</b>      | <b>0.986</b>      |
| Ours Stereo            | K       | 0.068        | 0.835        | 4.392        | 0.146        | 9.194         | 0.942           | 0.978             | 0.989             |

Lower is better  
Higher is better

Table 1. Comparison of different image formation models. Results on the KITTI 2015 stereo 200 training set disparity images [17]. For training, K is the KITTI dataset [17] and CS is Cityscapes [8]. Our model with left-right consistency performs the best, and is further improved with the addition of the Cityscapes data. The last row shows the result of our model trained *and tested* with two input images instead of one (see Sec. 4.3).

At test time, our network predicts the disparity at the finest scale level for the left image  $d^l$ , which has the same resolution as the input image. Using the known camera baseline and focal length from the training set, we then convert from the disparity map to a depth map. While we also estimate the right disparity  $d^r$  during training, it is not used at test time.

## 4. Results

Here we compare the performance of our approach to both supervised and unsupervised single view depth estimation methods. We train on rectified stereo image pairs, and do not require any supervision in the form of ground truth depth. Existing single image datasets, such as [41, 45], that lack stereo pairs, are not suitable for evaluation. Instead we evaluate our approach using the popular KITTI 2015 [17] dataset. To evaluate our image formation model, we compare to a variant of our algorithm that uses the original Deep3D [53] image formation model and a modified one, Deep3Ds, with an added smoothness constraint. We also evaluate our approach with and without the left-right consistency constraint.

### 4.1. Implementation Details

The network which is implemented in TensorFlow [1] contains 31 million trainable parameters, and takes on the order of 25 hours to train using a single Titan X GPU on a dataset of 30 thousand images for 50 epochs. Inference is fast and takes less than 35 ms, or more than 28 frames per second, for a  $512 \times 256$  image, including transfer times to and from the GPU. Please see the supplementary material and our code<sup>1</sup> for more details.

During optimization, we set the weighting of the different loss components to  $\alpha_{ap} = 1$  and  $\alpha_{lr} = 1$ . The possible output disparities are constrained to be between 0 and  $d_{max}$  using a scaled sigmoid non-linearity, where  $d_{max} = 0.3 \times$  the image width at a given output scale. As a result of our multi-scale output, the typical disparity of neighboring pixels will differ by a factor of two between each scale (as we are upsampling the output by a factor of two). To correct for this, we scale the disparity smoothness term  $\alpha_{ds}$  with  $r$  for each scale to get equivalent smoothing at each level. Thus  $\alpha_{ds} = 0.1/r$ , where  $r$  is the

<sup>1</sup>Available at <https://github.com/mrharicot/monodepth>

downscaling factor of the corresponding layer with respect to the resolution of the input image that is passed into the network.

For the non-linearities in the network, we used exponential linear units [7] instead of the commonly used rectified linear units (ReLU) [40]. We found that ReLUs tended to prematurely fix the predicted disparities at intermediate scales to a single value, making subsequent improvement difficult. Following [42], we replaced the usual deconvolutions with a nearest neighbor upsampling followed by a convolutions. We trained our model from scratch for 50 epochs, with a batch size of 8 using Adam [30], where  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ , and  $\epsilon = 10^{-8}$ . We used an initial learning rate of  $\lambda = 10^{-4}$  which we kept constant for the first 30 epochs before halving it every 10 epochs until the end. We initially experimented with progressive update schedules, as in [39], where lower resolution image scales were optimized first. However, we found that optimizing all four scales at once led to more stable convergence. Similarly, we use an identical weighting for the loss of each scale as we found that weighting them differently led to unstable convergence. We experimented with batch normalization [26], but found that it did not produce a significant improvement, and ultimately excluded it.

Data augmentation is performed on the fly. We flip the input images horizontally with a 50% chance, taking care to also swap both images so they are in the correct position relative to each other. We also added color augmentations, with a 50% chance, where we performed random gamma, brightness, and color shifts by sampling from uniform distributions in the ranges [0.8, 1.2] for gamma, [0.5, 2.0] for brightness, and [0.8, 1.2] for each color channel separately.

**Resnet50** For the sake of completeness, and similar to [33], we also show a variant of our model using Resnet50 [20] as the encoder, the rest of the architecture, parameters and training procedure staying identical. This variant contains 48 million trainable parameters and is indicated by **resnet** in result tables.

**Post-processing** In order to reduce the effect of stereo disocclusions which create disparity ramps on both the left side of the image and of the occluders, a final post-processing step is performed on the output. For an input image  $I$  at test time, we also

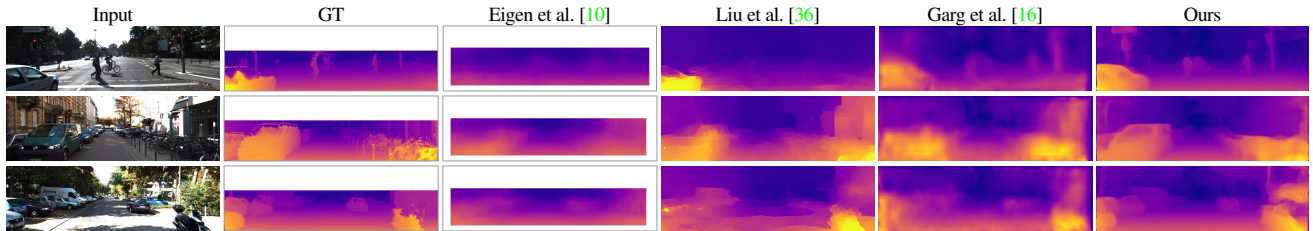


Figure 4. Qualitative results on the KITTI Eigen Split. The ground truth velodyne depth being very sparse, we interpolate it for visualization purposes. Our method does better at resolving small objects such as the pedestrians and poles.

compute the disparity map  $d'_l$  for its horizontally flipped image  $I'$ . By flipping back this disparity map we obtain a disparity map  $d''_l$ , which aligns with  $d_l$  but where the disparity ramps are located on the right of occluders as well as on the right side of the image. We combine both disparity maps to form the final result by assigning the first 5% on the left of the image using  $d''_l$  and the last 5% on the right to the disparities from  $d_l$ . The central part of the final disparity map is the average of  $d_l$  and  $d''_l$ . This final post-processing step leads to both better accuracy and less visual artifacts at the expense of doubling the amount of test time computation. We indicate such results using **pp** in result tables.

## 4.2. KITTI

We present results for the KITTI dataset [17] using two different test splits, to enable comparison to existing works. In its raw form, the dataset contains 42,382 rectified stereo pairs from 61 scenes, with a typical image being  $1242 \times 375$  pixels in size.

**KITTI Split** First we compare different variants of our method including different image formation models and different training sets. We evaluate on the 200 high quality disparity images provided as part of the official KITTI training set, which covers a total of 28 scenes. The remaining 33 scenes contain 30,159 images from which we keep 29,000 for training and the rest for evaluation. While these disparity images are much better quality than the reprojected velodyne laser depth values, they have CAD models inserted in place of moving cars. These CAD models result in ambiguous disparity values on transparent surfaces such as car windows, and issues at object boundaries where the CAD models do not perfectly align with the images. In addition, the maximum depth present in the KITTI dataset is on the order of 80 meters, and we cap the maximum predictions of all networks to this value. Results are computed using the depth metrics from [10] along with the *D1-all* disparity error from KITTI [17]. The metrics from [10] measure error in both meters from the ground truth and the percentage of depths that are within some threshold from the correct value. It is important to note that measuring the error in depth space while the ground truth is given in disparities leads to precision issues. In particular, the non-thresholded measures can be sensitive to the large errors in depth caused by prediction errors at small disparity values.

In Table 1, we see that in addition to having poor scaling prop-

erties (in terms of both resolution and the number of disparities it can represent), when trained from scratch with the same network architecture as ours, the Deep3D [53] image formation model performs poorly. From Fig. 6 we can see that Deep3D produces plausible image reconstructions but the output disparities are inferior to ours. Our loss outperforms both the Deep3D baselines and the addition of the left-right consistency check increases performance in all measures. In Fig. 5 we illustrate some zoomed in comparisons, clearly showing that the inclusion of the left-right check improves the visual quality of the results. Our results are further improved by first pre-training our model with additional training data from the Cityscapes dataset [8] containing 22,973 training stereo pairs captured in various cities across Germany. This dataset brings higher resolution, image quality, and variety compared to KITTI, while having a similar setting. We cropped the input images to only keep the top 80% of the image, removing the very reflective car hoods from the input. Interestingly, our model trained on Cityscapes alone does not perform very well numerically. This is likely due to the difference in camera calibration between the two datasets, but there is a clear advantage to fine-tuning on data that is related to the test set.

**Eigen Split** To be able to compare to existing work, we also use the test split of 697 images as proposed by [10] which covers a total of 29 scenes. The remaining 32 scenes contain 23,488 images from which we keep 22,600 for training and the rest for evaluation, similarly to [16]. To generate the ground truth depth images, we reproject the 3D points viewed from the velodyne laser into the left input color camera. Aside from only producing depth values for less than 5% of the pixels in the input image, errors are also introduced because of the rotation

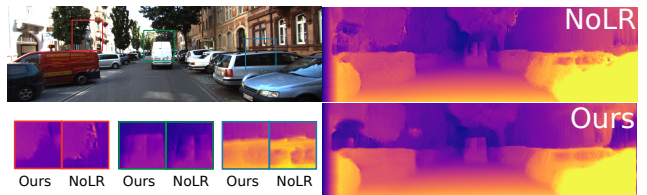


Figure 5. Comparison between our method with and without the left-right consistency. Our consistency term produces superior results on the object boundaries. Both results are shown without post-processing.

| Method                                | Supervised | Dataset | Abs Rel      | Sq Rel       | RMSE         | RMSE log     | $\delta < 1.25$ | $\delta < 1.25^2$ | $\delta < 1.25^3$ |
|---------------------------------------|------------|---------|--------------|--------------|--------------|--------------|-----------------|-------------------|-------------------|
| Train set mean                        | No         | K       | 0.361        | 4.826        | 8.102        | 0.377        | 0.638           | 0.804             | 0.894             |
| Eigen et al. [10] Coarse <sup>o</sup> | Yes        | K       | 0.214        | 1.605        | 6.563        | 0.292        | 0.673           | 0.884             | 0.957             |
| Eigen et al. [10] Fine <sup>o</sup>   | Yes        | K       | 0.203        | 1.548        | 6.307        | 0.282        | 0.702           | 0.890             | 0.958             |
| Liu et al. [36] DCNF-FCSP FT *        | Yes        | K       | 0.201        | 1.584        | 6.471        | 0.273        | 0.68            | 0.898             | 0.967             |
| <b>Ours No LR</b>                     | No         | K       | 0.152        | 1.528        | 6.098        | 0.252        | 0.801           | 0.922             | 0.963             |
| <b>Ours</b>                           | No         | K       | 0.148        | 1.344        | 5.927        | 0.247        | 0.803           | 0.922             | 0.964             |
| <b>Ours</b>                           | No         | CS + K  | 0.124        | 1.076        | 5.311        | 0.219        | 0.847           | 0.942             | 0.973             |
| <b>Ours pp</b>                        | No         | CS + K  | 0.118        | 0.923        | 5.015        | 0.210        | 0.854           | 0.947             | <b>0.976</b>      |
| <b>Ours resnet pp</b>                 | No         | CS + K  | <b>0.114</b> | <b>0.898</b> | <b>4.935</b> | <b>0.206</b> | <b>0.861</b>    | <b>0.949</b>      | <b>0.976</b>      |
| Garg et al. [16] L12 Aug 8x cap 50m   | No         | K       | 0.169        | 1.080        | 5.104        | 0.273        | 0.740           | 0.904             | 0.962             |
| <b>Ours cap 50m</b>                   | No         | K       | 0.140        | 0.976        | 4.471        | 0.232        | 0.818           | 0.931             | 0.969             |
| <b>Ours cap 50m</b>                   | No         | CS + K  | 0.117        | 0.762        | 3.972        | 0.206        | 0.860           | 0.948             | 0.976             |
| <b>Ours pp cap 50m</b>                | No         | CS + K  | 0.112        | 0.680        | 3.810        | 0.198        | 0.866           | 0.953             | <b>0.979</b>      |
| <b>Ours resnet pp cap 50m</b>         | No         | CS + K  | <b>0.108</b> | <b>0.657</b> | <b>3.729</b> | <b>0.194</b> | <b>0.873</b>    | <b>0.954</b>      | <b>0.979</b>      |
| <b>Ours pp uncropped</b>              | No         | CS + K  | 0.134        | 1.261        | 5.336        | 0.230        | 0.835           | 0.938             | 0.971             |
| <b>Ours resnet pp uncropped</b>       | No         | CS + K  | 0.130        | 1.197        | 5.222        | 0.226        | 0.843           | 0.940             | 0.971             |

|                  |
|------------------|
| Lower is better  |
| Higher is better |

Table 2. Results on KITTI 2015 [17] using the split of Eigen et al. [10]. For training, K is the KITTI dataset [17] and CS is Cityscapes [8]. The predictions of Liu et al. [36]\* are generated on a mix of the left and right images instead of just the left input images. For a fair comparison, we compute their results relative to the correct image. As in the provided source code, Eigen et al. [10]<sup>o</sup> results are computed relative to the velodyne instead of the camera. Garg et al. [16] results are taken directly from their paper. All results, except [10], use the crop from [16]. We also show our results with the same crop and maximum evaluation distance. The last two rows are computed on the uncropped ground truth.

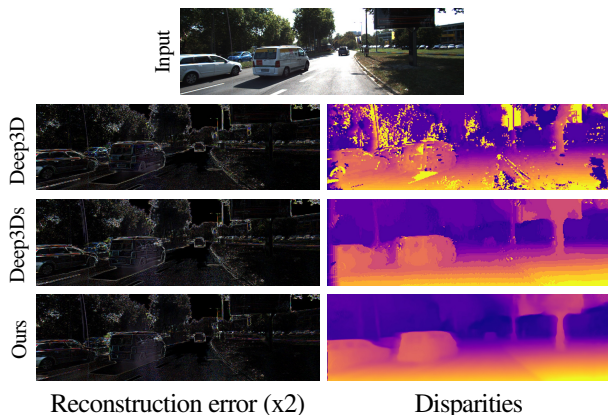


Figure 6. Image reconstruction error on KITTI. While all methods output plausible right views, the Deep3D image formation model without smoothness constraints does not produce valid disparities.

of the Velodyne, the motion of the vehicle and surrounding objects, and also incorrect depth readings due to occlusion at object boundaries. To be fair to all methods, we use the same crop as [10] and evaluate at the input image resolution. With the exception of Garg et al.’s [16] results, the results of the baseline methods are recomputed by us given the authors’s original predictions to ensure that all the scores are directly comparable. This produces slightly different numbers than the previously published ones, *e.g.* in the case of [10], their predictions were evaluated on much smaller depth images (1/4 the original size). For all baseline methods we use bilinear interpolation to resize the predictions to the correct input image size.

Table 2 shows quantitative results with some example outputs shown in Fig. 4. We see that our algorithm outperforms all other existing methods, including those that are trained with ground truth depth data. We again see that pre-training on the Cityscapes dataset improves the results over using KITTI alone.

### 4.3. Stereo

We also implemented a stereo version of our model, see Fig. 8, where the network’s input is the concatenation of both left and right views. Perhaps unsurprisingly, the stereo models outperforms our monocular network on every single metric, especially on the *D1-all* disparity measure, as can be seen in Table 1. This model was only trained for 12 epochs as it becomes unstable if trained for longer.

### 4.4. Make3D

To illustrate that our method can generalize to other datasets, here we compare to several fully supervised methods on the Make3D test set of [45]. Make3D consists of only RGB/Depth pairs and no stereo images, thus our method cannot train on this data. We use our network trained only on the Cityscapes dataset and despite the dissimilarities in the datasets, both in content and camera parameters, we still achieve reasonable results, even beating [28] on one metric and [37] on three. Due to the different aspect ratio of the Make3D dataset we evaluate on a central crop of the images. In Table 3, we compare our output to the similarly cropped results of the other methods. As in the case of the KITTI dataset, these results would likely be improved with more relevant training data. A qualitative comparison to some of the related methods is shown in Fig. 7. While our numerical results are not as good as the baselines, qualitatively, we compare favorably to the supervised competition.

### 4.5. Generalizing to Other Datasets

Finally, we illustrate some further examples of our model generalizing to other datasets in Figure 9. Using the model only trained on Cityscapes [8], we tested on the CamVid driving dataset [4]. In the accompanying video and the supplementary material we can see that despite the differences in location, image characteristics, and camera calibration, our model still

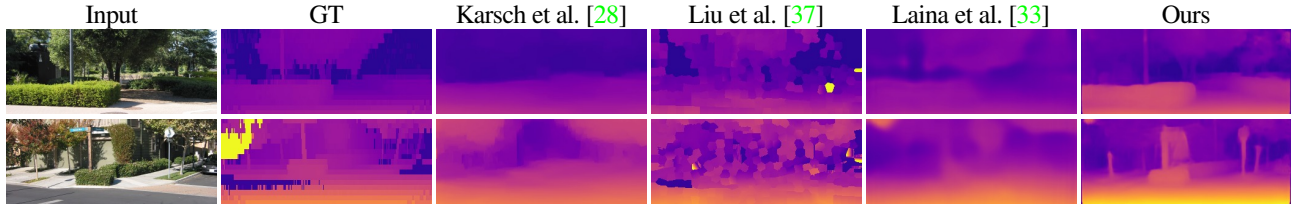


Figure 7. Our method achieves superior qualitative results on Make3D despite being trained on a different dataset (Cityscapes).

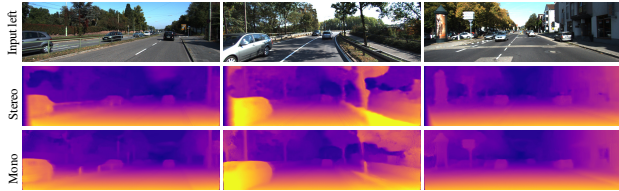


Figure 8. Our stereo results. While the stereo disparity maps contains more detail, our monocular results are comparable.

| Method                       | Sq Rel       | Abs Rel      | RMSE         | $\log_{10}$  |
|------------------------------|--------------|--------------|--------------|--------------|
| Train set mean*              | 15.517       | 0.893        | 11.542       | 0.223        |
| Karsch et al. [28]*          | 4.894        | 0.417        | 8.172        | 0.144        |
| Liu et al. [37]*             | 6.625        | 0.462        | 9.972        | 0.161        |
| Laina et al. [33] berHu*     | <b>1.665</b> | <b>0.198</b> | <b>5.461</b> | <b>0.082</b> |
| <b>Ours with Deep3D [53]</b> | 17.18        | 1.000        | 19.11        | 2.527        |
| <b>Ours</b>                  | 11.990       | 0.535        | 11.513       | 0.156        |
| <b>Ours pp</b>               | 7.112        | 0.443        | 8.860        | 0.142        |

Table 3. Results on the Make3D dataset [45]. All methods marked with an \* are supervised and use ground truth depth data from the Make3D training set. Using the standard C1 metric, errors are only computed where depth is less than 70 meters in a central image crop.

produces visually plausible depths. We also captured a 60,000 frame dataset, at 10 frames per second, taken in an urban environment with a wide angle consumer 1080p stereo camera. Finetuning the Cityscapes pre-trained model on this dataset produces visually convincing depth images for a test set that was captured with the same camera on a different day, please see the video in the supplementary material for more results.



Figure 9. Qualitative results on Cityscapes, CamVid, and our own urban dataset captured on foot. For more results please see our video.

## 4.6. Limitations

Even though both our left-right consistency check and post-processing improve the quality of the results, there are still some artifacts visible at occlusion boundaries due to the pixels in the occlusion region not being visible in both images. Explicitly reasoning about occlusion during training [23, 25] could improve these issues. It is worth noting that depending how large the baseline between the camera and the depth sensor, fully supervised approaches also do not always have valid depth for all pixels.

Our method requires rectified and temporally aligned stereo pairs during training, which means that it is currently not possible to use existing single-view datasets for training purposes *e.g.* [41]. However, it is possible to fine-tune our model on application specific ground truth depth data.

Finally, our method mainly relies on the image reconstruction term, meaning that specular [18] and transparent surfaces will produce inconsistent depths. This could be improved with more sophisticated similarity measures [54].

## 5. Conclusion

We have presented an unsupervised deep neural network for single image depth estimation. Instead of using aligned ground truth depth data, which is both rare and costly, we exploit the ease with which binocular stereo data can be captured. Our novel loss function enforces consistency between the predicted depth maps from each camera view during training, improving predictions. Our results are superior to fully supervised baselines, which is encouraging for future research that does not require expensive to capture ground truth depth. We have also shown that our model can generalize to unseen datasets and still produce visually plausible depth maps.

In future work, we would like to extend our model to videos. While our current depth estimates are performed independently per frame, adding temporal consistency [28] would likely improve results. It would also be interesting to investigate sparse input as an alternative training signal [58, 6]. Finally, while our model estimates per pixel depth, it would be interesting to also predict the full occupancy of the scene [11].

**Acknowledgments** We would like to thank David Eigen, Ravi Garg, Iro Laina and Fayao Liu for providing data and code to recreate the baseline algorithms. We also thank Stephan Garbin for his lua skills and Peter Hedman for his L<sup>A</sup>T<sub>E</sub>X magic. We are grateful for EPSRC funding for the EngD Centre EP/G037159/1, and for projects EP/K015664/1 and EP/K023578/1.



## References

- [1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. 5
- [2] A. Abrams, C. Hawley, and R. Pless. Heliometric stereo: Shape from sun position. In *ECCV*, 2012. 2
- [3] J. T. Barron, A. Adams, Y. Shih, and C. Hernández. Fast bilateral-space stereo for synthetic defocus. *CVPR*, 2015. 1
- [4] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 2009. 7
- [5] Y. Cao, Z. Wu, and C. Shen. Estimating depth from monocular images as classification using deep fully convolutional residual networks. *arXiv preprint arXiv:1605.02305*, 2016. 2
- [6] W. Chen, Z. Fu, D. Yang, and J. Deng. Single-image depth perception in the wild. In *NIPS*, 2016. 8
- [7] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. 5
- [8] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 5, 6, 7
- [9] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 2
- [10] 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, 6, 7
- [11] M. Firman, O. Mac Aodha, S. Julier, and G. J. Brostow. Structured Prediction of Unobserved Voxels From a Single Depth Image. In *CVPR*, 2016. 8
- [12] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 2
- [13] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. Deepstereo: Learning to predict new views from the world’s imagery. In *CVPR*, 2016. 2
- [14] Y. Furukawa and C. Hernández. Multi-view stereo: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2015. 2
- [15] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016. 2
- [16] R. Garg, V. Kumar BG, and I. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *ECCV*, 2016. 1, 3, 4, 6, 7
- [17] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 5, 6, 7
- [18] C. Godard, P. Hedman, W. Li, and G. J. Brostow. Multi-view reconstruction of highly specular surfaces in uncontrolled environments. In *3DV*, 2015. 8
- [19] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 3
- [20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5
- [21] P. Heise, S. Klose, B. Jensen, and A. Knoll. Pm-huber: Patchmatch with huber regularization for stereo matching. In *ICCV*, 2013. 4
- [22] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. *TOG*, 2005. 2
- [23] D. Hoiem, A. N. Stein, A. A. Efros, and M. Hebert. Recovering occlusion boundaries from a single image. In *ICCV*, 2007. 8
- [24] I. P. Howard. *Perceiving in depth, volume 1: basic mechanisms*. Oxford University Press, 2012. 1
- [25] A. Humayun, O. Mac Aodha, and G. J. Brostow. Learning to Find Occlusion Regions. In *CVPR*, 2011. 8
- [26] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 5
- [27] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *NIPS*, 2015. 3, 4
- [28] K. Karsch, C. Liu, and S. B. Kang. Depth transfer: Depth extraction from video using non-parametric sampling. *PAMI*, 2014. 2, 7, 8
- [29] K. Karsch, K. Sunkavalli, S. Hadap, N. Carr, H. Jin, R. Fonte, M. Sittig, and D. Forsyth. Automatic scene inference for 3d object compositing. *TOG*, 2014. 1
- [30] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5
- [31] L. Ladický, C. Häne, and M. Pollefeys. Learning the matching function. *arXiv preprint arXiv:1502.00652*, 2015. 2
- [32] L. Ladický, J. Shi, and M. Pollefeys. Pulling things out of perspective. In *CVPR*, 2014. 1, 2
- [33] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *3DV*, 2016. 2, 5, 8
- [34] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 2015. 1
- [35] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *CVPR*, 2015. 2
- [36] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *PAMI*, 2015. 1, 2, 6, 7
- [37] M. Liu, M. Salzmann, and X. He. Discrete-continuous depth estimation from a single image. In *CVPR*, 2014. 7, 8
- [38] W. Luo, A. Schwing, and R. Urtasun. Efficient deep learning for stereo matching. In *CVPR*, 2016. 2
- [39] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016. 2, 3, 4, 5
- [40] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 5
- [41] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 5, 8
- [42] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. 5

- [43] V. Patraucean, A. Handa, and R. Cipolla. Spatio-temporal video autoencoder with differentiable memory. *arXiv preprint arXiv:1511.06309*, 2015. 3
- [44] R. Ranftl, V. Vineet, Q. Chen, and V. Koltun. Dense monocular depth estimation in complex dynamic scenes. In *CVPR*, 2016. 2
- [45] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3d scene structure from a single still image. *PAMI*, 2009. 2, 5, 7, 8
- [46] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 2002. 2
- [47] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *PAMI*, 2016. 2, 4
- [48] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 2013. 1
- [49] D. Stoyanov, M. V. Scanzanella, P. Pratt, and G.-Z. Yang. Real-time stereo reconstruction in robotically assisted minimally invasive surgery. In *MICCAI*, 2010. 1
- [50] X. Wang, D. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *CVPR*, 2015. 2
- [51] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *Transactions on Image Processing*, 2004. 4
- [52] R. J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical engineering*, 1980. 2
- [53] J. Xie, R. Girshick, and A. Farhadi. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In *ECCV*, 2016. 1, 2, 3, 4, 5, 6, 8
- [54] J. Žbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *JMLR*, 2016. 2, 3, 8
- [55] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Is l2 a good loss function for neural networks for image processing? *arXiv preprint arXiv:1511.08861*, 2015. 4
- [56] T. Zhou, P. Krähenbühl, M. Aubry, Q. Huang, and A. A. Efros. Learning dense correspondence via 3d-guided cycle consistency. *CVPR*, 2016. 3
- [57] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *ECCV*, 2016. 3
- [58] D. Zoran, P. Isola, D. Krishnan, and W. T. Freeman. Learning ordinal relationships for mid-level vision. In *ICCV*, 2015. 8

# Unsupervised Monocular Depth Estimation with Left-Right Consistency

## Supplementary Material

### 1. Model architecture

| “Encoder” |   |   |         |     |     |        | “Decoder” |   |   |          |     |     |                       |
|-----------|---|---|---------|-----|-----|--------|-----------|---|---|----------|-----|-----|-----------------------|
| layer     | k | s | chns    | in  | out | input  | layer     | k | s | chns     | in  | out | input                 |
| conv1     | 7 | 2 | 3/32    | 1   | 2   | left   | upconv7   | 3 | 2 | 512/512  | 128 | 64  | conv7b                |
| conv1b    | 7 | 1 | 32/32   | 2   | 2   | conv1  | iconv7    | 3 | 1 | 1024/512 | 64  | 64  | upconv7+conv6b        |
| conv2     | 5 | 2 | 32/64   | 2   | 4   | conv1b | upconv6   | 3 | 2 | 512/512  | 64  | 32  | iconv7                |
| conv2b    | 5 | 1 | 64/64   | 4   | 4   | conv2  | iconv6    | 3 | 1 | 1024/512 | 32  | 32  | upconv6+conv5b        |
| conv3     | 3 | 2 | 64/128  | 4   | 8   | conv2b | upconv5   | 3 | 2 | 512/256  | 32  | 16  | iconv6                |
| conv3b    | 3 | 1 | 128/128 | 8   | 8   | conv3  | iconv5    | 3 | 1 | 512/256  | 16  | 16  | upconv5+conv4b        |
| conv4     | 3 | 2 | 128/256 | 8   | 16  | conv3b | upconv4   | 3 | 2 | 256/128  | 16  | 8   | iconv5                |
| conv4b    | 3 | 1 | 256/256 | 16  | 16  | conv4  | iconv4    | 3 | 1 | 128/128  | 8   | 8   | upconv4+conv3b        |
| conv5     | 3 | 2 | 256/512 | 16  | 32  | conv4b | disp4     | 3 | 1 | 128/2    | 8   | 8   | iconv4                |
| conv5b    | 3 | 1 | 512/512 | 32  | 32  | conv5  | upconv3   | 3 | 2 | 128/64   | 8   | 4   | iconv4                |
| conv6     | 3 | 2 | 512/512 | 32  | 64  | conv5b | iconv3    | 3 | 1 | 130/64   | 4   | 4   | upconv3+conv2b+disp4* |
| conv6b    | 3 | 1 | 512/512 | 64  | 64  | conv6  | disp3     | 3 | 1 | 64/2     | 4   | 4   | iconv3                |
| conv7     | 3 | 2 | 512/512 | 64  | 128 | conv6b | upconv2   | 3 | 2 | 64/32    | 4   | 2   | iconv3                |
| conv7b    | 3 | 1 | 512/512 | 128 | 128 | conv7  | iconv2    | 3 | 1 | 66/32    | 2   | 2   | upconv2+conv1b+disp3* |
|           |   |   |         |     |     |        | disp2     | 3 | 1 | 32/2     | 2   | 2   | iconv2                |
|           |   |   |         |     |     |        | upconv1   | 3 | 2 | 32/16    | 2   | 1   | iconv2                |
|           |   |   |         |     |     |        | iconv1    | 3 | 1 | 18/16    | 1   | 1   | upconv1+disp2*        |
|           |   |   |         |     |     |        | disp1     | 3 | 1 | 16/2     | 1   | 1   | iconv1                |

Table 1: Our network architecture, where **k** is the kernel size, **s** the stride, **chns** the number of input and output channels for each layer, **input** and **output** is the downscaling factor for each layer relative to the input image, and **input** corresponds to the input of each layer where + is a concatenation and \* is a  $2\times$  upsampling of the layer.

### 2. Post-Processing

The post-processed disparity map corresponds to the per-pixel weighted sum of two components:  $d^l$  the disparity of the input image,  $d^{fl}$  the flipped disparity of the flipped input image.

We define the per-pixel weight map  $w^l$  for  $d^l$  as

$$w^l(i,j) = \begin{cases} 1 & \text{if } j \leq 0.1 \\ 0.5 & \text{if } j > 0.2 \\ 5*(0.2-i)+0.5 & \text{else,} \end{cases}$$

where  $i, j$  are normalized pixel coordinates, and the weight map  $w^l$  for  $d^{fl}$  is obtained by horizontally flipping  $w^l$ .

The final disparity is calculated as,

$$d = d^l w^l + d^{fl} w^{fl}.$$

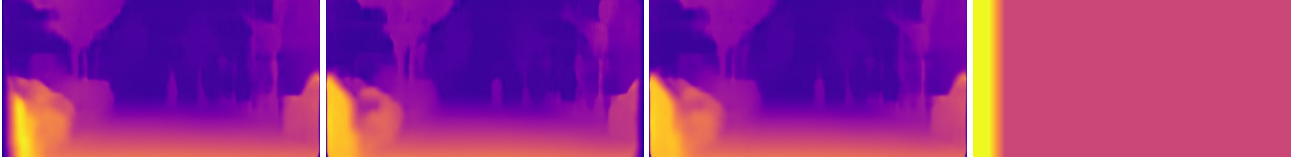


Figure 1: Example of a post-processed disparity map. From left to right: The disparities  $d^l$ ,  $d^m$ ,  $d$ , and the weight map  $w^l$ .

### 3. Deep3D Smoothness Loss

In the main paper we also compared to our enhanced version of the Deep3D [1] image formation model that includes smoothness constraints on the disparities. Deep3D outputs an intensity image as a weighted sum of offset copies of the input image. The weights  $w_i$ , seen in Fig. 2a, can be seen as a discrete probability distribution over the disparities for each pixel, as they sum to one. Thus, smoothness constraints cannot be applied directly onto these distributions. However, we see (in Fig. 2c) that if the probability mass is concentrated into one disparity, *i.e.*  $\max(w) \approx 1$ , then the sum of the cumulative sum of the weights is equal to the position of the maximum. To encourage the network to concentrate probability at single disparities, we added a cost,

$$C_{max} = \frac{1}{N} \|\max(w_i) - 1\|^2, \quad (1)$$

and its associated weight  $\alpha_{max} = 0.02$ . Assuming the maximum of each distribution is one, such as in Fig. 2b, meaning the network only picks one disparity per pixel, we can see that the (sum of the) cumulative sum  $cs(w_i)$  of the distribution (Fig. 2c) directly relates to the location of the maximum disparity:

$$d = \operatorname{argmax}_i(w_i) = n - \sum_i^n cs(w_i), \quad (2)$$

where  $n$  is the maximum number of disparities.

In the example presented in Fig. 2, we can see that the maximum is located at disparity 3, and that Equation 2 gives us  $d = 8 - 6 = 3$ . We use this observation to build our smoothness constraint for the Deep3D image formation model.

We can then directly apply smoothness constraints on the gradients of the cumulative sums of the weights at each pixel, so

$$C_{ds} = \frac{1}{N} \sum_{i,j} |\partial_x cs(w)_{ij}| + |\partial_y cs(w)_{ij}|, \quad (3)$$

and its associated weight  $\alpha_{ds} = 0.1$ .

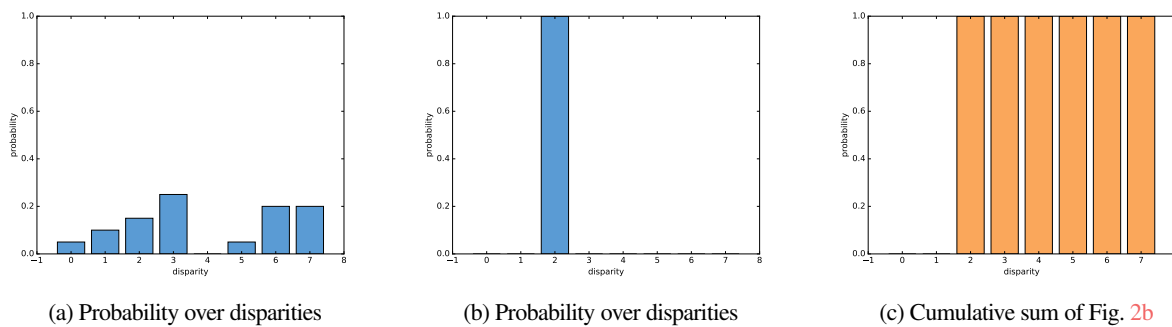


Figure 2: Deep3Ds per-pixel disparity probabilities.

### 4. More KITTI Qualitative Results

In Fig. 3 we show some additional qualitative comparisons for the KITTI dataset using the Eigen split.

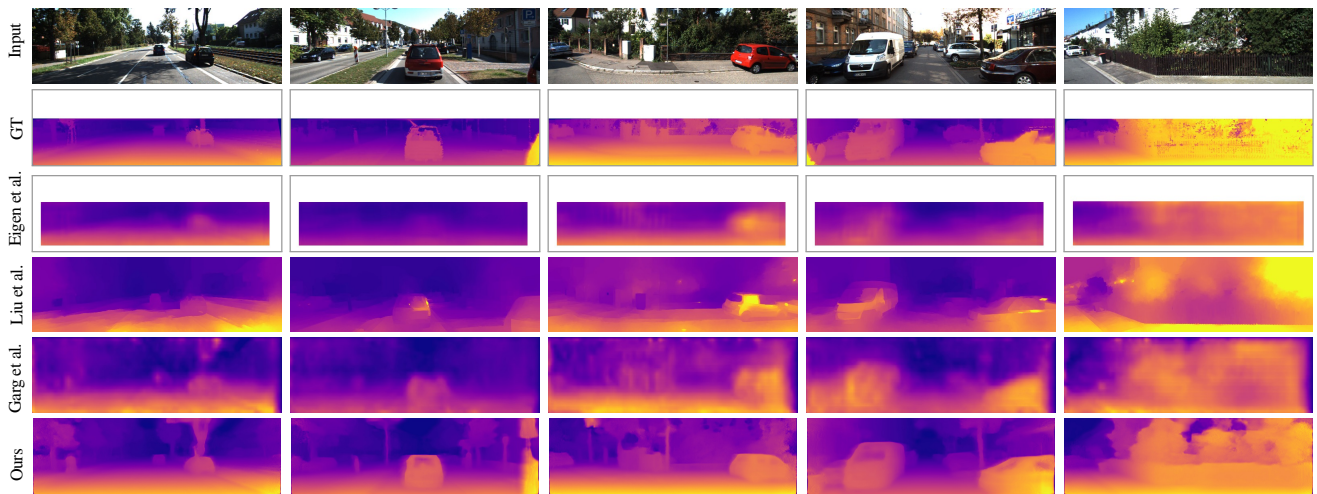
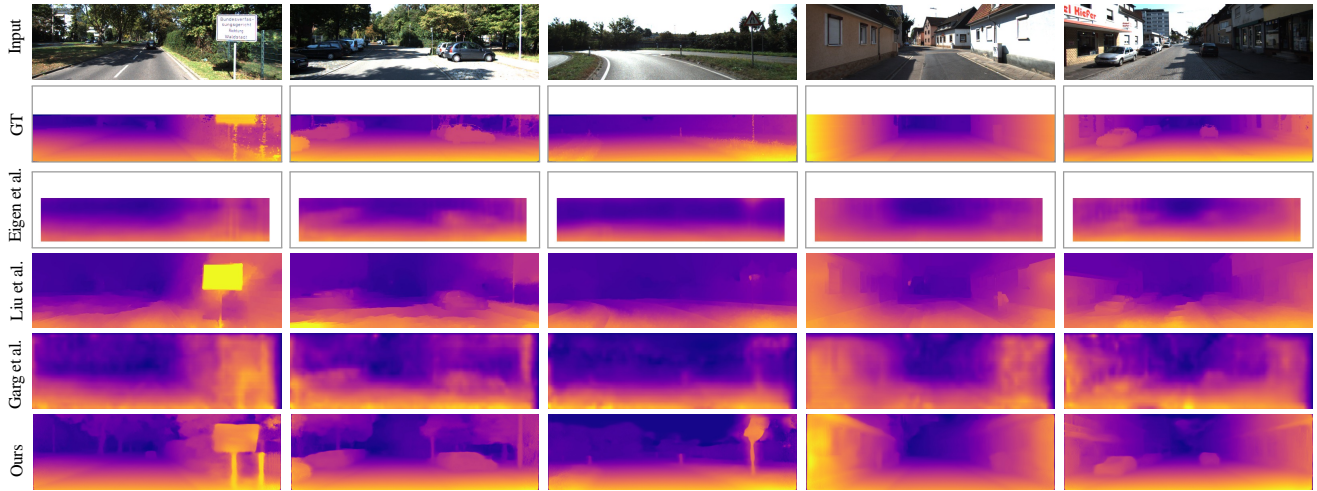


Figure 3: Additional qualitative results on the KITTI Eigen Split. As ground truth velodyne depth is very sparse, we interpolate it for visualization purposes.

## 5. Disparity Error Maps

As we train our model with color augmentations, we can apply the same principle at test time and analyze the distribution of the results. We applied 50 random augmentations to each test images and show the standard deviation of the disparities per pixel (see Figure 4). We can see that the network gets confused with close-by objects, texture-less regions, and occlusion boundaries. Interestingly, one test image was captured in a tunnel, resulting in a very dark image. Our network clearly shows high uncertainty for this sample as it is very different from the rest of the training set.

## References

- [1] J. Xie, R. Girshick, and A. Farhadi. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In *ECCV*, 2016. 2

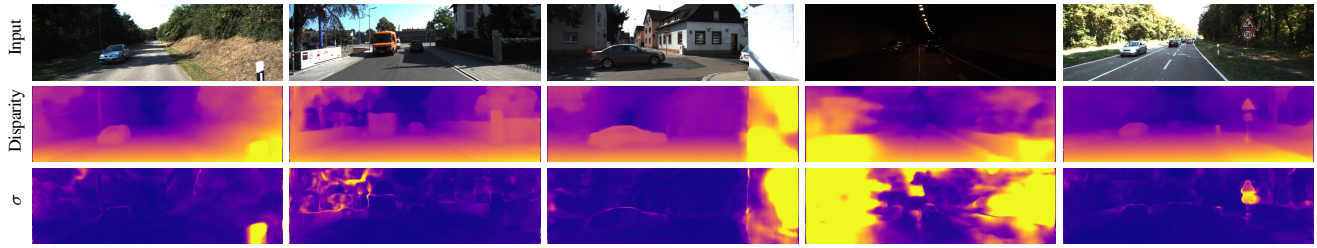


Figure 4: Uncertainty of our model on the KITTI dataset. From top to bottom: input image, predicted disparity, and standard deviation of multiple different augmentations. We can see that there is uncertainty in low texture regions and at occlusion boundaries.