Unsupervised Monocular Depth Estimation with Left-Right Consistency

Xiaojing Hu
Department of Computer Science
University at Buffalo
xhu7@buffalo.edu

Sili Liu
Department of Computer Science
University at Buffalo
sililiu@buffalo.edu

Abstract

This article is an implementation report of the paper "*Unsupervised Monocular Depth Estimation with Left-Right Consistency*" [1].

1 Introduction

Nowadays, with the development of electronic devices, the use of cameras is ubiquitous. We've got cameras in our phones, in the monitors, in the robots and etc. We want these cameras to be our surrogate "eyes". As for a human being, besides sensing objects with one's eyes, he is also able to estimate the distances from the objects. This is not only because humans have two eyes, but also because humans have collected plenty of experience at distance measuring. For example, if you are shown a 2D picture instead of a 3D scene, it's still probable that you can tell the distances of the points in the image from the camera. To learn to infer the distances from the objects to a camera, this task is called depth estimation. Depth estimation has a lot of real-life applications. For example, we need a robot to be able to estimate the distances from the objects to avoid running into them or to act on them. Another example is the VR (visual reality) system, where the device needs to construct a 3D scene from a sequence of images. Of course we can use sensors to detect the distance, but if depth estimation algorithms can achieve high accuracy, then the cost will be relatively low. Depth estimation is a rather big field. A traditional approach is to use the disparity of multiple images to estimate the depth information. Usually, the images can be obtained from multiple cameras or a single camera under different conditions. In this case, we can obtain the absolute depth estimation. Another approach is to estimate the depth from a single monocular RGB image. The application of the later approach is more broad, however it is also much more difficult. And usually monocular depth estimation problem is regarded as a classification or regression problem learnt in a supervised way. However, though these methods are successful in application, they all need large amount of ground truth depthmaps, which is not feasible since in real life the scanners are usually expensive. And even if we can get the depthmaps using scanners, the obtained depthmaps probably are inaccurate because of occlusion and other noise. So, it is very useful if we can predict the depth information without any ground truth depthmaps. And in the paper [1], the authors proposed an unsupervised learning method to do monocular depth estimation. The basic idea of this method is to use the relationship between depthmaps and disparity maps. At training time, pairs of images from binocular cameras are used. At testing time, only a single image from a monocular camera is used for depth estimation.

2 Method

2.1 Depth Estimation as Image Reconstruction

Since there is no ground truth depth maps, the authors regarded this problem as an image reconstruction problem during training. The intuition behind is that given a pair of images from binocular cameras, if we can construct one image from the other, then we must have learnt some information regarding the 3D scene.

At training time, we have access to 2 images I^l and I^r, corresponding to the left and right images from a pair of cameras. d^r is the dense correspondence field when applied on the left image, can help to reconstruct the right image. We will refer to the reconstructed image $I^l(d^r)$ as $\widetilde{I}^r$. Similarly, we can also estimate the left image given the right one . Assuming that the images are rectified, then d corresponds to the image disparity.

The basic idea of this method is to use the relationship between disparity maps and depth maps.

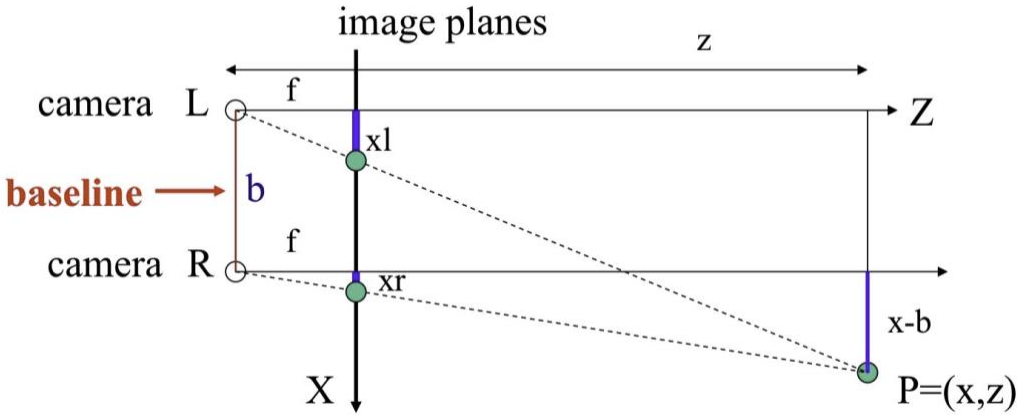

Figure 1. This is a graphical illustration of the relationship between the disparity maps and the depth value. *L* is the left camera and *R* is the right camera. *b* is the distance between them. f is the focal length. And *P* is the position of a point on an object.

Through some simple math induction, if we denote the distance from to the baseline in Figure 1 as $\hat{d}$ and the disparity between left eye and right eye as d, then we have $\hat{d} = bf/d$. This means that if we can predict the disparity maps from the images, we can recover the depth maps using this relationship.

2.2 Depth Estimation Network

Instead of using a pair of images to calculate the disparity map, the authors proposed a method that uses a single left image to infer both the left-right and right-left disparity maps, and enforce them to be consistent.

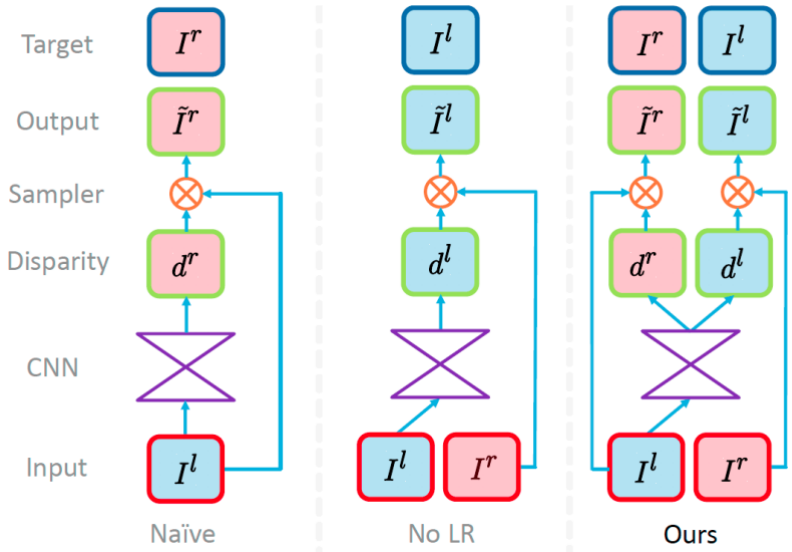

Figure 3. Sampling strategies for backward mapping. With naïve sampling the CNN produces a disparity map aligned with the target instead of the input. No LR corrects for this, but suffers from artifacts. Our approach uses the left image to produce disparities for both images, improving quality by enforcing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

The left model in Figure 3 uses a single left image as input, it predicts the right-left disparity map and uses a bilinear sampler to output a synthesized right image. However, through this procedure we will only get the disparity map aligned with the right image while actually what we want is the disparity map aligned with the left image. The middle model inputs both the left and right image. It uses only the left image to infer the left-right disparity map while uses the right image for sampling to get the synthesized left image. But this model alone will cause "texture-copy" artifacts and and errors at depth discontinuities. The authors solved this problem using left-right consistency. In this method, the left image is used to generate both the left-right and right-left disparity maps, and both the left and right images are used for sampling to generate both the left and right image. Experiment shows that enforcing left-right consistency will help to get more accurate results.

The architecture of this model is based on DispNet, but a few modifications are made. The network has 2 main parts, the encoder and the decoder. The predicted disparity maps are outputted at four different scales, doubled in spatial resolution.

3 Training loss

As mentioned before, The predicted disparity maps are outputted at four different scales, doubled in spatial resolution. For each scale s , the authors defined a loss C_s. So the total loss is $C = \sum_{s=1}^{4} C_s$. Every loss C_s is a combination of 3 terms: appearance matching loss, disparity smoothness loss, and left-right disparity consistency loss,

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

3.1 Appearance matching loss

After we get the disparity maps, we need to reconstruct the left image with the right image and the

right-to-left disparity map and reconstruct the right image with the left image and the left-to-right disparity map. The disparity map can be viewed as an affine matrix. For example, suppose we've already know the left image I^l and the disparity map d^r, we need to reconstruct $I^{\tilde{r}}$. For each pixel in $I^{\tilde{r}}$, we can get its position in the original left image using the disparity map. One problem is that since the disparity map is calculated from the CNNs, the values are not likely to be integers. So, in this method we use bilinear sampling. Each pixel in the reconstructed image is a weighted sum of 4 pixels from the original opposite image. And we use the image sampler from the spatial transformer network (STN) [2]. This makes the whole model fully differentiable.

The whole appearance matching loss is comprised of 2 terms. One is a single scale structural similarity index (SSIM) [3] term and the other is an L1 loss. So, for the left image,

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

where we use a simplified SSIM with a 33 block filter. α is a coefficient and N is the number of pixels.

3.2 Disparity smoothness loss

This term is to enforce the disparity maps to be smooth with an L1 penalty on the disparity gradient. And because the discontinuities usually occur at image gradients, we weighted the cost using an edge-aware term.

$$C_{ds}{}^{l} = \frac{1}{N}\sum_{ij} \left| \partial_x d_{ij}{}^{l} \right| e^{-\left\| \partial_x I_{ij}{}^{l} \right\|} + \left| \partial_y d_{ij}{}^{l} \right| e^{-\left\| \partial_y I_{ij}{}^{l} \right\|}$$

3.3 Left-right disparity consistency loss

The third term is a left-right disparity consistency term.

$$C_{lr}{}^{l} = \frac{1}{N}\sum_{i,j} \left| d_{ij}{}^{l} - d^{r}{}_{ij + d_{ji}{}^{l}} \right|.$$

For each pixel (i,j) in the left image, we can computer its corresponding pixel in the right image using the disparity map $d^l{}_{ij}$, and then we require that the corresponding pixel in $d^r{}_{ij}$ is the same as the original pixel in $d^l{}_{ij}$.

At training, the model takes both the left and right images as input and outputs both the left-to-right and right-to-left disparity maps. And at test time, we discard the right image and only inputs the left image, and outputs the right-to-left disparity map. Then use the relationship between the depth and the disparity value we can recover the depth map.

4 Results

We train this model on rectified stereo image pairs, and do not require any supervision in the form

of ground truth depth. We evaluate this approach using the popular KITTI [4].

4.1 Model architecture

The network architecture shown in the Table 1 below, where **k** is the kernel size, **s** the stride, **chns** the number of input and output channels for each layer, **in** and **out** 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× upsampling of the layer`.

"Encoder"						
layer	**k**	**s**	**chns**	**in**	**out**	**input**
conv1	7	2	3/32	1	2	left
conv1b	7	1	32/32	2	2	conv1
conv2	5	2	32/64	2	4	conv1b
conv2b	5	1	64/64	4	4	conv2
conv3	3	2	64/128	4	8	conv2b
conv3b	3	1	128/128	8	8	conv3
conv4	3	2	128/256	8	16	conv3b
conv4b	3	1	256/256	16	16	conv4
conv5	3	2	256/512	16	32	conv4b
conv5b	3	1	512/512	32	32	conv5
conv6	3	2	512/512	32	64	conv5b
conv6b	3	1	512/512	64	64	conv6
conv7	3	2	512/512	64	128	conv6b
conv7b	3	1	512/512	128	128	conv7

"Decoder"						
layer	**k**	**s**	**chns**	**in**	**out**	**input**
upconv7	3	2	512/512	128	64	conv7b
iconv7	3	1	1024/512	64	64	upconv7+conv6b
upconv6	3	2	512/512	64	32	iconv7
iconv6	3	1	1024/512	32	32	upconv6+conv5b
upconv5	3	2	512/256	32	16	iconv6
iconv5	3	1	512/256	16	16	upconv5+conv4b
upconv4	3	2	256/128	16	8	iconv5
iconv4	3	1	128/128	8	8	upconv4+conv3b
disp4	3	1	128/2	8	8	iconv4
upconv3	3	2	128/64	8	4	iconv4
iconv3	3	1	130/64	4	4	upconv3+conv2b+disp4*
disp3	3	1	64/2	4	4	iconv3
upconv2	3	2	64/32	4	2	iconv3
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: model architecture

4.2 Implementation

We implemented this network in Tensorflow [5]. It contains 31 million trainable parameters, and takes on the order of 17 hours to train using a single NVIDIA GTX 2080 GPU on a dataset of 30 thousand images for 50 epochs.

During optimization, we set the parameters shown in the Table 2 below. As a result of the multi-scale output, the typical disparity of neighboring pixels will differ by a factor of two between each scale (as upsampling the output by a factor of two). To correct for this, we scale the disparity smoothness term α_{ds} with r for each scale to get equivalent smoothing at each level.

Loss Function			
α_{ap}	α_{lr}	α_{ds}	d_{max}
1	1	$0.1/r$	0.3×image width
Optimization (Adam) [3]			
β_1	β_2	ε	λ (initial)
0.9	0.999	10^{-8}	10^{-4}

Table 2: some training parameters, where r is the 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 [6] instead of the commonly used rectified linear units [7]. Because ReLUs tended to prematurely fix the predicted disparities at intermediate scales to a single value, making subsequent improvement difficult. And we replaced

the usual deconvolutions with a nearest neighbor upsampling followed by a convolutions. We trained this model from scratch for 50 epochs, with a batch size of 8. We kept the leaning rate constant for first 30 epochs before halving it every 10 epochs until the end.

In order to reduce the effect 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 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 l final post-processing step leads to both better accuracy and less visual artifacts at the expense of doubling the amount of test time computation.

4.3 Performance

We present results for the KITTI dataset with a typical image being 1242×375 pixels in size. We evaluate 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. 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. The results are shown in the Tab.3 with some example outputs shown in Fig.4.

Method	Supervised	Dataset	RMSE	log RMSE
Eigen [9] Coarse	Yes	K	6.563	0.292
Eigen [9] Fine	Yes	K	6.307	0.282
Author	No	K	5.927	0.247
Ours	No	K	6.031	0.261
Author	No	K + CS	5.311	0.219
Ours	No	K + CS	5.413	0.224

Table 3. Results of Eigen [9], Original author and our method. For training, K is the KITTI dataset and CS is Cityscapes.

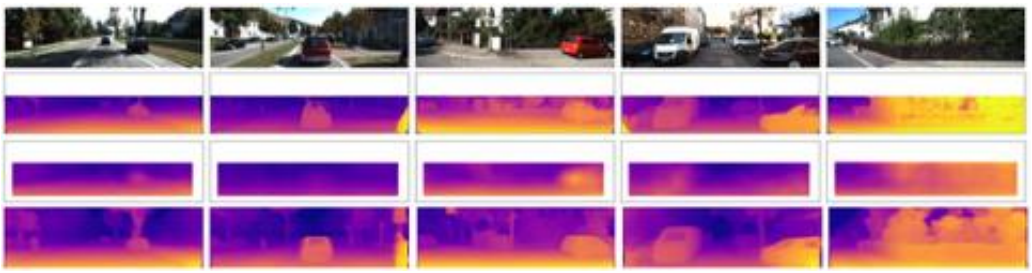

Figure 4. Qualitative results on the KITTI dataset. The 1st row is image, 2nd row is ground truth, 3rd row is output of Eigen and 4th row is our method output.

We also test our model which trained on KITTI and Cityscapes on NYU Depth dataset [10] which is comprised of video sequences from a variety of indoor scenes as recorded by both the RGB and Depth cameras from the Microsoft Kinect.

Figure 5. Qualitative results on the NYU Depth dataset. The 1st row is image, 2nd row is ground truth, 3rd row is output of our method.

Finally, we illustrate some further examples of our model generalizing to other datasets. As we can see in Fig.5, our model doesn't generalize to NYU Depth dataset well.

5 Limitations

Even though left-right consistency check and postprocessing 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. And 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. Moreover, our model does not have a good generalization to other datasets. Finally, our method mainly relies on the image reconstruction term, meaning that

specular and transparent surfaces will produce inconsistent depths.

6 Conclusion

We implement this 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.

7 Division of this project

Xiaojing Hu takes responsibility for theory part, doing data preprocessing (transfer KITTI velodyne data to depth ground truth) and analysing the results of our project.

Sili Liu takes responsibility for reconstructing the model architecture using tensorflow, training and testing the model and writing a live demo program.

References

[1] Cl´ement Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency.CoRR,abs/1609.03677, 2016.

[2] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In NIPS, 2015.

[3] 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] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In CVPR, 2012.

[5] 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.

[6] 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.

[7] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In ICML, 2010.

[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.

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

[10] N. Silberman, P. Kohli, D. Hoiem, R. Fergus. Indoor Segmentation and Support Inference from RGBD Images. In ECCV 2012