Unsupervised Monocular Depth Estimation

Baladhurgesh Balagurusamy Paramasivan [1] **Denim Patel** [1] **Adhitya Athyur Ganesh** [1] **Adithya Gupte** [1]

Abstract

Estimating depth from single images is an ill posed problem as it does not contain enough information required to generate such depth map. Learning based methods have been effective in solving this problem in a supervised fashion by training on depth maps and image pairs. Here, we have implemented self-supervised learning based method called MonoDepth which provides a novel training objective and training method to generate dense depth map. Model tries to predict the disparity map and then using this disparity map, depth map is generated. Instead of depth maps which are harder to obtain in a reliable manner this method uses stereo images for loss generation. The novel loss function exploits epipolar geometrical constraints as well as enforces left right image disparity consistency. This approach produced state of the art results on the KITTI dataset and is an foundational paper in the field of what now is known as self- supervised monocular depth estimation.

Keywords: Monocular Depth Estimation, Self supervised learning, KITTI, MonoDepth, DispNet

1. Introduction

Estimating depth is a crucial step in scene reconstruction, 3D object recognition, segmentation, and detection and thus has seen a lot of interest. This problem was previously addressed by using the methods like structure from motion and by using multi view geometry. There has been a lot of work [12] in classical computer vision addressing this problem. A lot of supervised learning methods based on stereo images have also been developed. All these methods assume that reliable multi-view data is available for the scenes under consideration. Humans are very effective at monocular depth estimation by taking cues from the environment and have inspired many researchers to find ways to emulate it. A lot of research interest is generated in the field of monocular depth estimation to overcome this limiting assumption of the need of multiple views of the scene. The learning based methods for depth estimation provide a way to estimate depth without making any geometrical assumptions about the scene, further this paper also relaxes the need for multiple views of the scene and also does this in an unsupervised manner. The paper we are implementing poses the depth estimation problem as an image reconstruction problem. The depth is estimated as an intermediate quantity while generation of the image. The fully convolutional model is trained on stereo images instead of traditionally training on depth images like RGBD or point clouds. The model used is fully differential as opposed to other similar approaches used. Some earlier works like Deep3D by [4] in essence also have similar methodology as the paper we are implementing but have non differentiable model and loss. Hence they perform Taylor approximation that makes optimization more difficult.
The main contributions of this paper are:

1. Solving the monocular depth estimation problem in an unsupervised form.
2. Using stereo images for training instead of depth maps which are harder to obtain.
3. Having a completely differentiable loss function while having above mentioned advantages.

2. Related Work

The problem of depth estimation is a old one and has been addressed numerous times by using stereo images [12] or by using multiple views [3]. These approaches are typically only applicable when there is more than one input image available of the scene of interest. With the advent of deep learning, there has been rise of methods that use neural networks to perform depth estimation. Networks outperformed hand engineered feature s and got far superior results on function matching. Early methods in monocular depth estimation were majorly supervised approach based. In Depth estimation problem, supervised learning

[1]Worcester Polytechnic Institute. Correspondence to: Baladhurgesh <bbalagurusamypar@wpi.edu>, Denim <dmpatel@wpi.edu>, Adithya Athyur Ganesh <aathyurganesh@wpi.edu>, Aditya Gupte <agupte@wpi.edu>.

CS/DS 541 Deep Learning Final Project, Worcester Polytechnic Institute.

methods needs either RGBD or point clouds as the ground truth for training. The early attempts to solve the problem of depth estimation using single image was done by Saxena et al. [11]. They proposed a learning based approach that uses Markoff random field and over samples the input to generate depth maps. Other methods which utilised handcrafted representations for depth estimation then followed by some probabilistic enhancement model like MRF(Markoff Random Fields) were also developed. Another successful method was proposed by Eigen & Fergus [1]. A lot of progress has happened in MDE(monocular depth estimation)[2; 13; 10; 16] since.

SOTA in monocular depth estimation:The current state of the art method for monocular depth estimation that uses KITTI dataset as the benchmark is the method by Lee et al. [8]. The architecture consists of an encoder-decoder network, and it uses Local Planar Guidance layers as opposed to skip connections and nearest neighbour upsampling in the decoder. The earlier state of the art before BTS was

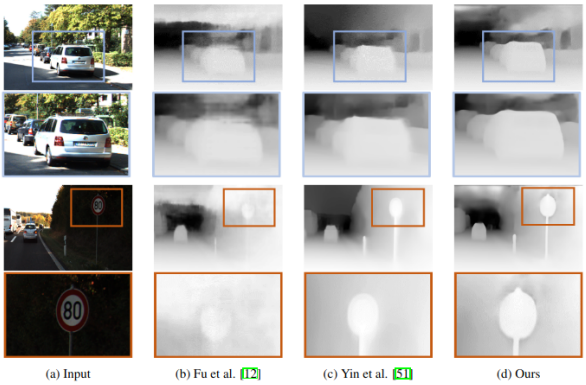

Figure 1. Results of BTS

a method by Fu et al. [2] called DORN which casted the problem as classification problem and used a Spacing Increasing Discretization strategy to simplify the complicated decoder network. The MonoDepth paper we implement as

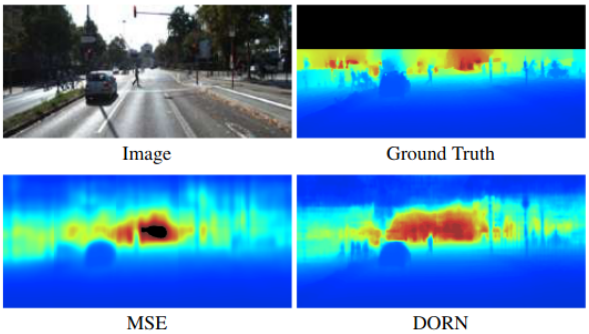

Figure 2. Results of DORN

the part of our project is the foundational paper(2017) of the field of self supervised monocular vision.

3. Method

3.1. Problem statement

The problem at hand is of dense per pixel depth estimation but if directly addressed would require depth data for training which as stated is harder to obtain, hence the problem is restated as an image reconstruction problem.So, given an image from the left camera of a stereo camera pair we need to reconstruct the other image. Training utilizes both images I^l (the left image) and I^r (the right image). The method in the paper tries to find the dense correspondence field d^t that transforms the left image into right image and vice versa. This can be mathematically represented as

$$\bar{I}^l = I^r\left(d^r\right) \tag{1}$$

$$\bar{I}^r = I^l\left(d^l\right) \tag{2}$$

If the images are rectified the d^l and d^r corresponds to the disparity d. Once we have the disparity between the image pair, the depth can be recovered trivially by using the equation $\hat{d} = bf/d$. Here b represents the baseline distance between the left and right camera centers and f is the camera focal length.

3.2. Depth Estimation Network architecture and concept

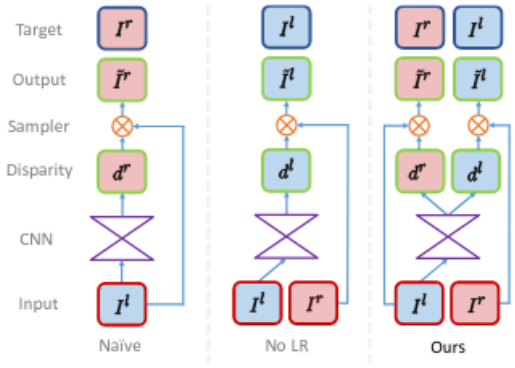

Figure 3. sampling strategies for backward mapping of output disparity maps. Naive approach aligns the disparity with output, the No LR approach produces the artifacts, the third approach by the authors gives better consistency in disparity outputs

If the model is used to generate right image $\bar{I}^r$ from the left image I^l it will naively generate disparity mapping aligned to the target image I^r, However the output disparity should be aligned with input image I^l and the sampling should be done from the right image I^r. This approach alone should suffice as a complete method to get depth maps but

this method produces unwanted artifacts at edges where the depth is discontinuous. Thus the method where both disparity maps are estimated using the same image and sampling is done from opposite input images is used by the authors. This method along with the left right consistency loss term help to get better and smooth depth maps. Figure 3 shows the approach used for producing disparity maps and sampling as discussed above. The sampling used to generate the image is bilinear sampling, similar to what is used by Spatial Transformation Network(STN) by Jaderberg et al. [7]. The image and the disparity is provided to the sampler, the sampler in the case of generation of the right image samples the points from the original left image according to the disparity. The disparity when added to the image may correspond to non integer number that do not represent pixels, in that case bilinear sampling is done. BS considers the 4 diagonal nearest neighbours of points and then the respective pixel values are generated. This process is differentiable and is advantageous for producing a completely differentiable and and end to end trainable model.

Table 1 shows the architecture of the model. it has two main parts the encoder and decoder. This architecture is inspired from DispNet [9]. The CNN encoder has standard architecture and has ELU activations instead of the standard ReLU as a non linearity. The decoder instead of using transpose convolutions/upconvolution uses neighbourhood sampling followed by convolutions. The decoder also has skip connections from the encoders activation blocks. The decoder produces 4 output disparity maps at different spatial resolutions. The network outputs two disparity maps at each spatial resolution, hence the two output channels at each spatial resolution can be seen in the decoder architecture. To generate loss at all the spatial resolutions we need to scale down the input image to those respective resolution. This is done by down sampling the input image by half at each level. Hence the 4 images have spatial resolutions scaled by the factor of 1/8,1/4,1/2,1.

3.3. Training Loss

Loss at each scale C_s composed of three parts:Appearance Matching Loss C_{ap}, Disparity Smoothness Loss C_{ds}, Left-Right Disparity Consistency Loss C_{lr}.

$$C_s = \alpha_{ap}\left(C^l_{ap} + C^r_{ap}\right) + \alpha_{ds}\left(C^l_{ds} + C^r_{ds}\right) + \alpha_{lr}\left(C^l_{lr} + C^r_{lr}\right) \tag{3}$$

The loss C_s is calculated at all output scales s we get the total loss as the sum $C = \sum_{s=1}^{4} C_s$. The loss is calculated for four spatial resolutions and the loss is weighted and added together to give total loss. The weightage given to the loss at the lowest spatial resolution is the highest.

Appearance Matching Loss : The appearance matching loss enforces that the output image be similar to the input image [7]. The Apperance matching loss has a L1 norm combined with SSIM [14]. The SSIM (structural similarity) is a metric which is used to measure the similarity between to uncompressed images. It is given by

$$SSIM(x,y) = \frac{\left(2\mu_x\mu_y + c_1\right)\left(2\sigma_{xy} + c_2\right)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)} \tag{4}$$

This operation is performed on a windows of size $3X3$. The μ_x μ_y are the averages over the $3X3$ input windows x and y. σ_x^2 and σ_y^2 are the variances and the σ_{xy} is the covariance. The structural similarity is subtracted from 1 , thus higher the similarity lower is the loss. The L1 penalty is also applied on the difference between the generated image and the original image. This combination of SSIM and L1 penalty are weighted and together form the appearance matching loss as given. A simplified version of the SSIM loss where a 3x3 block filter is used instead of a Gaussian filter.

$$C^l_{ap} = \frac{1}{N}\sum_{i,j} \alpha \frac{1 - SSIM\left(I^l_{ij}, \tilde{I}^l_{ij}\right)}{2} + (1-\alpha)\left\|I^l_{ij} - \tilde{I}^l_{ij}\right\| \tag{5}$$

Disparity Smoothness Loss: To have smooth depth maps $L1$ penalty is enforced on the disparity gradient. This term is weighted with the image gradient making it edge aware, as depth discontinuities occur mostly at edges.

$$C^l_{ds} = \frac{1}{N}\sum_{i,j} \left|\partial_x d^l_{ij}\right| e^{-\left\|\partial_x I^l_{ij}\right\|} + \left|\partial_y d^l_{ij}\right| e^{-\left\|\partial_y I^l_{ij}\right\|} \tag{6}$$

Here ∂d is the depth gradient and ∂I is image gradient.

Left-Right Disparity Consistency Loss: The model produces two disparity maps, the left and right image disparities. To ensure coherence between both the disparity maps a $L1$ disparity consistency penalty is imposed. This cost attempts to make the left-view disparity map be equal to the projected right-view disparity map. The projection uses the same bilinear sampler for projection.

$$C^l_{lr} = \frac{1}{N}\sum_{i,j} \left|d^l_{ij} - d^r_{ij + d^l_{ij}}\right| \tag{7}$$

. **Temporal Disparity Smoothness Loss**: Temporal smoothness loss ensures the disparity maps that are produced are smooth even wrt. two consecutive video frames. This ensures smooth disparity maps while dealing with videos. The $\partial_t d$ is the gradient of the disparity map wrt time. This term is weighted with the a term that takes image gradients into consideration.

$$C^l_{t_d s} = \frac{1}{N}\sum_{i,j} \left|\partial_t d^l_{ij}\right| e^{-\left\|\partial_t I^l_{ij}\right\|} \tag{8}$$

"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: Our network architecture, where **k** is the kernel size, **s** the stride, **chns** the number of input and output channels for each layer, **in**put and **out**put 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.

.

All the mathematical equations for individual losses are given only for left view, the formulas have to be mirrored to the get the corresponding loss values for the right view, both the left and right view values as given in equation 3 are used for the Loss C_s.

4. Experiments

We carried out certain experiments by varying the hyper parameters and taking combinations of losses as explained in the up coming section.

4.1. Datatset

: The dataset we chose is the Kitti dataset.The dataset contains $42,382$ rectified stereo pairs from 61 scenes, with a typical image being 1242 x 375 pixels in size. 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.This dataset is refered to as KITTi split. Ground truth depth images were obtained from the KITTI dataset.

4.2. Training

The network was developed using PyTorch. We trained for a total of 50 epochs which takes around 40 min per epoch. The hardware we used to train the network and evaluate the results was Google Cloud Platform and PC with NVIDIA RTX 2060, NVIDIA .

Experiments were conducted in four steps. Each experiment was conducted with different combinations of loss functions and with slightly different hyper parameters. The loss function used for different experiments are shown in Table 2.

We used the dataloader to load the data and trained the network for 50 epochs. The output provides two disparity maps, which are passed into a Bilinear sampler to reconstruct the images. These images were used for calculation of Loss values and update weights. The disparity maps were obtained as intermediate values from the network, which then were converted into depth maps.

In the first experiment, we used only Appearance matching loss to optimize the output. In the second experiment, Disparity smoothness loss was added along with Appearance matching loss to test the change in the quality and smoothness of output. We then added L-R disparity consistency loss in the third experiment since it was the novel approach used in the paper.

We observed that the paper MonoDepth2 [5] uses temporal appearance matching loss which makes the results more accurate. We couldn't add temporal appearance loss directly as we were short of time to implement PoseNet. Hence we added a temporal smoothness loss function in our final experiment. Our intuition was to add temporal information

and that might solve the infinite depth problem, as well as ensure that the disparity is consistent with respect to time.

Learning rate scheduling is used in training procedure. Learning rate was set as 10^{-4} for the first 30 epochs. It was then decayed to $10-5$ for the next 20 epochs.

5. Results

We observe from $Fig.8$ that as we added the components to the loss functions, the system gained accuracy and smoothness. We see that the disparity map with Disparity smoothness loss seems more smoother in comparison with just appearance matching loss. Similarly, the disparity map with with temporal smoothness only gives importance to objects and not insignificant objects like the shadows on the roads. We see this as an improvement over the disparity map with L-R disparity consistency loss as described in our base paper.

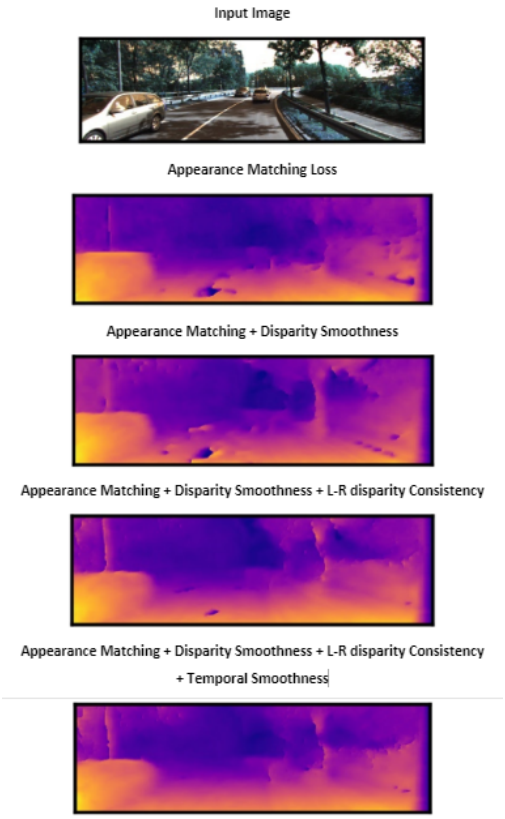

Fig 8. Disparity maps corresponding to each experiment with loss functions specified above the map given a single input image

6. Conclusions and Future Work

From the results and values, we conclude that the Appearance matching loss contributes the most during optimization, since it is a direct loss. We also observed the effect of the new edge aware temporal smoothness loss which reduces the sudden obvious false depth detects of the road. This keeps the network from getting confused with obvious depth defects. Prioritizing loss function like disparity smoothness, Temporal Smoothness loss by giving them more weight reduces the sharpness of the image to make it more smoother. One more finding was that the L-R consistency loss wasn't contributing as much as the other loss functions during training.

We propose the following changes to improve the efficiency and effectiveness in terms of memory footprint of the network.

1. Instead of using a concatenation operator in the decoder, we can reduce the latency by a greater amount by replacing it with a summing operator. This method reduces the number of channels in the layers by a significant margin yet the result shows little to no reduction in results.

2. One more way to make the computations faster is to use a lighter encoder such as MobileNet[6]. MobileNet[6] can also be used in the decoder side and depthwise convolution can be performed.

3. Perform model pruning, which significantly drops the convolution kernels from the model with little degradation in results

4. Perform the convolution before upsampling.

5. Incorporate suggestions of FastDepth [15] to make the model lightweight.

References

[1] Eigen, David and Fergus, Rob. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pp. 2650–2658, 2015.

[2] Fu, Huan, Gong, Mingming, Wang, Chaohui, Batmanghelich, Kayhan, and Tao, Dacheng. Deep ordinal regression network for monocular depth estimation, 2018.

[3] Furukawa, Yasutaka, Hernández, Carlos, et al. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015.

Experiment #	Appearance matching Loss	Disparity Smoothness Loss	L-R Disparity Consistency Loss	Temporal smoothness matching loss
1	Yes	No	No	No
2	Yes	Yes	No	No
3	Yes	Yes	Yes	No

Table 1. Loss functions used in ever experiment.

	Higher is better
	Lower is better

Method (Loss type)	Supervised	DataSet	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
AP	No	KITTI	0.2329	3.7105	8.566	0.301	0.741	0.906	0.956
AP + DS	No	KITTI	0.2538	4.3579	9.105	0.352	0.685	0.862	0.931
AP + DS +LR	No	KITTI	0.2246	3.3004	8.099	0.312	0.720	0.890	0.948
AP + DS +LR + TS	No	KITTI	0.2360	3.7233	8.319	0.323	0.708	0.883	0.943
Monodepth 1	No	KITTI	0.148	1.344	5.927	0.247	0.803	0.922	0.964
Monodepth 2	No	KITTI	0.132	1.044	5.142	0.210	0.845	0.948	0.977
Semantically guided	No	KITTI	0.100	0.761	4.270	0.175	0.902	0.965	0.982
SOTA (BTS)	No	KITTI	0.059	0.245	2.756	0.096	0.956	0.993	0.998

Table 2. Comparison of our results with other paper results

[4] Garg, Ravi, G, Vijay Kumar B., and Reid, Ian D. Unsupervised CNN for single view depth estimation: Geometry to the rescue. *CoRR*, abs/1603.04992, 2016. URL http://arxiv.org/abs/1603.04992.

[5] Godard, Clément, Aodha, Oisin Mac, and Brostow, Gabriel J. Digging into self-supervised monocular depth estimation. *CoRR*, abs/1806.01260, 2018. URL http://arxiv.org/abs/1806.01260.

[6] Howard, Andrew G., Zhu, Menglong, Chen, Bo, Kalenichenko, Dmitry, Wang, Weijun, Weyand, Tobias, Andreetto, Marco, and Adam, Hartwig. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL http://arxiv.org/abs/1704.04861.

[7] Jaderberg, Max, Simonyan, Karen, Zisserman, Andrew, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pp. 2017–2025, 2015.

[8] Lee, Jin Han, Han, Myung-Kyu, Ko, Dong Wook, and Suh, Il Hong. From big to small: Multi-scale local planar guidance for monocular depth estimation, 2019.

[9] Mayer, Nikolaus, Ilg, Eddy, Hausser, Philip, Fischer, Philipp, Cremers, Daniel, Dosovitskiy, Alexey, and Brox, Thomas. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4040–4048, 2016.

[10] Roy, Anirban and Todorovic, Sinisa. Monocular depth estimation using neural regression forest. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5506–5514, 2016.

[11] Saxena, Ashutosh, Chung, Sung H, and Ng, Andrew Y. Learning depth from single monocular images. In *Advances in neural information processing systems*, pp. 1161–1168, 2006.

[12] Scharstein, Daniel and Szeliski, Richard. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002.

[13] Wang, Peng, Shen, Xiaohui, Lin, Zhe, Cohen, Scott, Price, Brian, and Yuille, Alan L. Towards unified depth and semantic prediction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2800–2809, 2015.

[14] Wang, Zhou, Bovik, Alan C, Sheikh, Hamid R, and Simoncelli, Eero P. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[15] Wofk, Diana, Ma, Fangchang, Yang, Tien-Ju, Karaman, Sertac, and Sze, Vivienne. Fastdepth: Fast monocular depth estimation on embedded systems. *CoRR*, abs/1903.03273, 2019. URL http://arxiv.org/abs/1903.03273.

[16] Zhang, Ziyu, Schwing, Alexander G, Fidler, Sanja, and Urtasun, Raquel. Monocular object instance segmentation and depth ordering with cnns. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2614–2622, 2015.