Learning Single Camera Depth Estimation using Dual-Pixels Supplement

1. Derivation of Equation 2

Suppose a point light source is located at depth $Z(x, y)$ where the center of the camera lens is at position $(0, 0, 0)$. Light from this point light source is focused by the lens to another point on the opposite side of the lens. Let Z_i be the distance from the lens to this other point. Also, let g be the focus distance and g_i be the position of the sensor. By the paraxial and thin-lens approximations,

$$\frac{1}{g_i} = \frac{1}{f} - \frac{1}{g} \quad \text{and} \quad \frac{1}{Z_i} = \frac{1}{f} - \frac{1}{Z}. \tag{1}$$

By similar triangles, the blur size b is

$$b = \frac{L(g_i - Z_i)}{Z_i} \tag{2}$$

Substituting Eqn. 1 into Eqn. 2, we get Eqn. 2 in the main paper

$$b = \frac{Lg}{1 - f/g}\left(\frac{1}{g} - \frac{1}{Z}\right) \tag{3}$$

2. Data Processing

2.1. Depth from Multi View Stereo

We use two different stereo algorithms for computing the "ground truth" depth maps we use for training and evaluation. These depth maps are computed at a resolution of 756×1008, i.e., one quarter the resolution of RGB images.

We use the COLMAP multi-view stereo algorithm [1, 9] that computes per pixel depth and filters based on geometric consistency. We sample depth using inverse perspective sampling in the range [0.2m, 100m] to yield D^*. Confidence C is set to zero wherever COLMAP does not provide depth due to geometric inconsistency, and is set to 1 elsewhere.

Because the depth maps from COLMAP tend to have edge fattening artifacts on our data, we implemented our own plane-sweep multi-view stereo algorithm. We plane sweep along 256 planes sampled using inverse perspective sampling in the range [0.2m, 100m] and take the minimum of a filtered cost volume as each pixel's depth. To compute the cost volume, for each pixel, we compute the sum of absolute differences for each of the warped neighbors and then bilaterally filter the cost volume using the grayscale reference image as the guide image. This ensures that we are aggregating costs over similar pixels in a local window thus avoiding edge fattening artifacts [8]. We use a spatial sigma of 3 pixels and a range sigma of 12.5 for the bilateral filter. Finally, we normalize the plane indices to [0, 1] range so that they are in the same domain as COLMAP depth. To compute confidence, we check for depth coherence across views by checking for left / right consistency [2]. We first compute consistency with each of the 4 neighboring images:

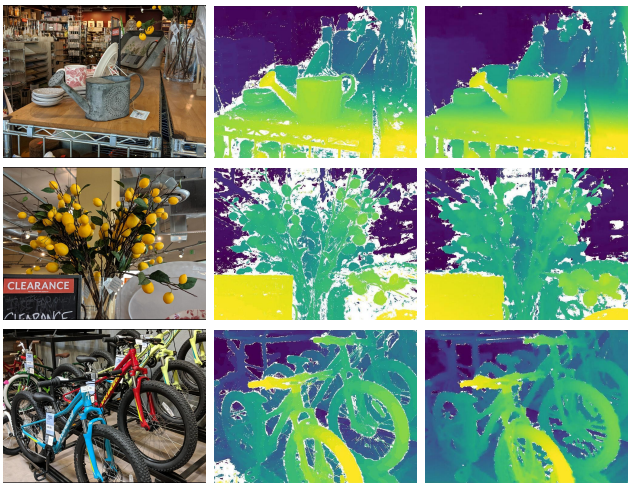
$$C_j(x, y) = \exp\left(-\frac{\left\|D_0^*(x, y) - D_j^*\left(M(x, y; D_0^*)\right)\right\|^2}{2\sigma^2}\right) \tag{4}$$

where $\sigma = 1/256$, and j is the index of the neighboring image. Then, under the assumption that a pixel must be visible in at least two other cameras for its depth to be reliable, we take the product of the largest two $C_j(x, y)$ values for each pixel to compute our final confidence $C(x, y)$.

A sample of images from our test set with corresponding depth from our method and COLMAP is shown in Fig. 1.

2.2. Preprocessing of RGB and Dual-Pixel Data

RGB images are 3024×4032, but we always downsample them to 1512×2016. The green pixels in each Bayer quad on the camera sensor are split in half (Fig. 2 in main paper). In the RGB image, the sensor sums adjacent green half-pixels to form the green channel. In the DP data, the sensor bins four green half-pixels in a 4×2 pattern to yield DP data of size $756 \times 2016 \times 2$. This data is 10-bit raw data and we apply a square root to the raw data and quantize the result to 8-bits. We also upsample the first dimension to 1512, so that its dimensions match those of the downsampled RGB image. When feeding RGB and DP images as input to the model, they are simply concatenated along the channel dimension to form a 5 channel input.



(a) RGB Image (b) Our True Depth (c) COLMAP's Depth

Figure 1. Images (a) from our test set, ground truth depth (b) computed using our multi-view stereo pipeline, and ground truth depth (c) computed using COLMAP [1]. Low confidence regions are shown in white. Our depth tends to be conservative in labelling a depth sample confident and avoids edge fattening artifacts.

3. Evaluation Metrics

3.1. Affine Invariant Weighted Error

One metric we use to evaluate models against our ground-truth depths is the minimum of a L^p-norm between an estimated inverse depth $\hat{D}$ and the true inverse depth D^* (weighted by the true depth's confidence C and scaled by the number of pixels) under all possible affine transformations. Let us define $\mathrm{AIWE}(p)$ as:

$$\min_{a,b} \left(\frac{\sum_{(x,y)} C(x,y) \left| D^*(x,y) - \left(a\hat{D}(x,y) + b \right) \right|^p}{\sum_{(x,y)} C(x,y)} \right)^{1/p} \tag{5}$$

Because root mean squared error and mean absolute error are standard choices when comparing depth maps, in the paper we present $\mathrm{AIWE}(2)$ (affine-invariant weighted RMSE) and $\mathrm{AIWE}(1)$ (affine-invariant weighted MAE). $\mathrm{AIWE}(2)$ can be evaluated straightforwardly by solving a least-squares problem, and $\mathrm{AIWE}(1)$ can be computed using iteratively reweighted least squares (in our experiments, we use 5 iterations).

3.2. Spearman's Rank Correlation Coefficient

We also use Spearman's rank correlation coefficient ρ_s for evaluation, which evaluates the ordinal correctness of the estimated depth. Because ρ_s is a function of the rank of each pixel's depth, it is invariant to any monotonic transformation of the depth which, naturally, includes affine transformations (with positive scales). Because our ground-truth depths D^* may contain repeated elements, ρ_s is computed by first computing the ranks of all elements in D^* and $\hat{D}$ and then computing the Pearson correlation of those ranks. We use the ground-truth depth confidences C when computing Pearson correlation (using it to weight the expectations used to compute the variances and covariance of the ranks) thereby resulting in a weighted variant of Spearman's ρ. To handle cases when the affine scaling is negative, we take the absolute value of ρ_s, and we report $1 - |\rho_s|$ to maintain consistency with $\mathrm{AIWE}(\cdot)$, in terms of lower values being better.

4. Model Architecture

Our DPNet architecture is composed of two key building blocks, an encoder block $E(i, o, s)$ and a decoder block $D(i, o)$ where i denotes number of intermediate features, o denotes number of output features, and s denotes the stride which controls the downsampling done by the encoder block. Each encoder block takes as input the output from the previous encoder block. Each decoder block takes as input the output of the previous decoder block and the output of an encoder block as a skip connection. Unless otherwise mentioned, we use Batch Normalization [6] before each convolution layer and PReLu [5] as an activation function for each output with initial leakiness a_i set to be 0.05.

Each encoder block $E_a(i, o, s)$ consists of a series of 3 convolutional layers, the first of which has i filters with size 3×3 and stride s, the second of which is a depthwise separable 3×3 convolutional layer with i filters, and the third of which is a 1×1 convolutional layer with o filters whose output is added to the max-pooled input (with pool size and stride both s) before applying a PReLu activation.

We also use a different encoder block $E_b(o, s)$ that is directly applied to the input images, which is a convolutional layer with o filters of size 7×7 and stride s, whose output is concatenated with max-pooled input images with pool size and stride s.

For each decoder block $D(i, o)$, we first apply a 4×4 transposed convolutional layer with stride 2 and i filters to the output of the previous decoder layer, followed by a 3×3 depth separable convolutional layer and a 1×1 convolutional layer, each with i filters, followed by a 3×3 depth separable convolutional layer with i filters whose output is added to the filtered skip connections before which itself has been filtered via a 3×3 depth separable convolutional layer. PReLu activation is applied after summing the two. Finally, a 1×1 convolutional layer with o filters generates the output for the next decoder block.

The overall model consists of a series of encoders followed by a series of decoders:

$E_b(8,2)$ $E_a^1(11,11,1)$

$E_a(16,32,2)$ $E_a(16,32,1)$ $E_a^2(16,32,1)$

$E_a(16,64,2)$ $E_a(16,64,1)$ $E_a^3(16,64,1)$

$E_a(32,128,2)$ $E_a(32,128,1)$ $E_a^4(32,128,1)$

$E_a(32,128,2)$ $E_a(32,128,1)$ $E_a(32,128,1)$

$D^4(32,128)$

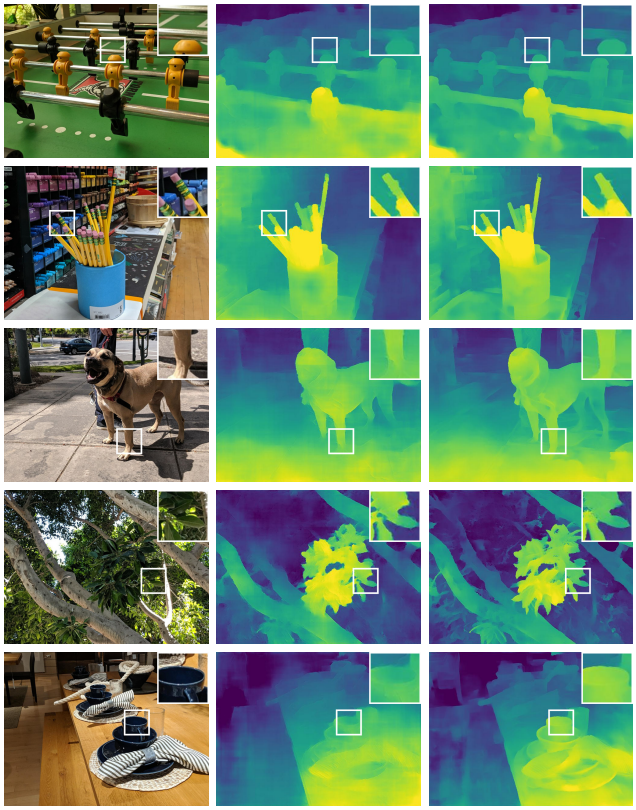
$D^3(16,64)$

$D^2(16,32)$

$D^1(8,8)$

where the outputs of each encoder marked with a superscript are connected by skip connections to a corresponding decoder with that superscript. The predictions at 5 different resolutions are obtained by applying a 3×3 convolution with a single filter and no activation and no Batch Normalization to the outputs of the decoders and the last encoder.

Our VGG model is same as that of [4] but we remove the last decoder block since our depth maps are at half the input resolution.



(a) Input Image (c) DPNet (Affine) RGB + DP (c) VGG (Affine) RGB + DP

Figure 2. Results of DPNet and VGG, both with RGB + DP input and trained with affine invariance.

	AIWE(1)	AIWE(2)	$1-\lvert\rho_s\rvert$
DPNet trained on Stereo	.0218	.0319	.180
DPNet trained on Multi-view	**.0175**	**.0264**	**.139**

Table 1. Comparison of stereo training data with multi-view training data. Accuracy is higher when using all the views for training vs just using the center-bottom camera pairs from the capture rig.

	AIWE(1)	AIWE(2)	$1-\lvert\rho_s\rvert$
Extended test set	.0188	.0276	.153
Standard test set	**.0175**	**.0264**	**.139**

Table 2. DPNet's accuracy on our standard test set (which only contains devices that are in the training set) vs our extended test set, which contains 7 other devices that were not used to generate the training set. The accuracy is only slightly worse, suggesting that our model has learned to circumvent the need for calibration.

5. Supplementary Results

5.1. VGG vs DPNet

As reported in Table 2 in the main paper, the best results with VGG model are slightly inferior than DPNet in spite of having a larger capacity. This is because VGG has a tendency to overfit due to the larger capacity. Qualitatively, we find the results with VGG to be very similar to results with DPNet (Fig 2). VGG also overfits for RGB input, hence those results are omitted from Table 2 in the main paper.

5.2. Multi-View vs Stereo

In Table 1, we present additional results in which we demonstrate that simultaneously training our model on all 4 alternative views provided by our capture setup outperforms an ablation of our technique that has been trained on only stereo views.

5.3. Generalization Across Devices

Our training and test sets consist of images from only three different phones. As noted by Wadhwa *et al*. [10], the relationship between depth and disparity from dual-pixels can vary from device to device because of variations during the manufacturing process, which they compensate for with a calibration procedure. Though we do not apply any per-device calibration, our performance degrades only slightly for phones that were not used to capture the images in the training set (Table 2). To demonstrate this, we evaluate our model on an extended test set containing devices that were not used to acquire any images in the training set. To construct this set, we use data from all 5 phones on the capture rig, i.e., the center phone and the surrounding phones. This extended set contains data from all 10 devices used for capture while the training set contains only data from just 3 of those devices.

Method	Invariance	Percentile Based WRMSE	
		Our Depth	COLMAP Depth
DPNet (RGB)	Scale	.0890	.0908
	Affine	.1502	.1484
DPNet (RGB+DP)	Scale	.0390	.0417
	Affine	**.0328**	**.0368**

Table 3. WRMSE metric for a subset of rows from Table 2 in the main paper where affine ambiguity is resolved by considering $1/3$ and $2/3$ percentile values. This also shows the importance of DP input and affine invariance being useful for training with DP input.

5.4. Additional Metrics

For scale invariant prediction, [11] introduces a metric where the scale ambiguity is resolved by taking the ratio of the median of the prediction and of the ground truth. This is not directly applicable to our affine invariant prediction, as a single correspondence does not overconstrain an affine transformation. To adapt this technique to the affine invariance case, we compute the $1/3$ and $2/3$ weighted percentile values (where the median would be the $1/2$ percentile) of the prediction and of the ground truth (using the confidences of the ground truth as weights) and then use those two correspondences to recover an affine transformation to resolve the ambiguity. Table 3 shows that DP input is critical and affine invariance helps learning with DP input when measured with this metric.

5.5. Additional Comparisons

We show additional results in Fig. 3 and 4. In addition, we also show the best and the worst 5 results of our method as determined by $1-|\rho_s|$ metric in Fig. 5 and 6 respectively.

5.6. Results on data from Wadhwa *et al*. [10]

We also run our model on the dual-pixel data provided by [10] and show that affine invariant depth can be used to synthetically defocus an image (Fig. 7). We use our DPNet model trained with only DP input, and the unknown affine mapping is determined by choosing the depth to focus at and the amount of synthetic defocus to render. Similar to [10], we ensure that depth maps are edge-aligned with the corresponding RGB image by applying a bilateral filter followed by joint bilateral upsampling [7]. Rendering is done using the algorithm of [10]. As seen in the figure, fewer errors in depth results in fewer errors in the synthetically defocused image.

References

[1] COLMAP. `https://colmap.github.io/`.

[2] Michael Bleyer and Christoph and Rhemann. Patchmatch stereo - stereo matching with slanted support windows. *BMVC*, 2011.

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

[4] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. *CVPR*, 2017.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *ICCV*, 2015.

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015.

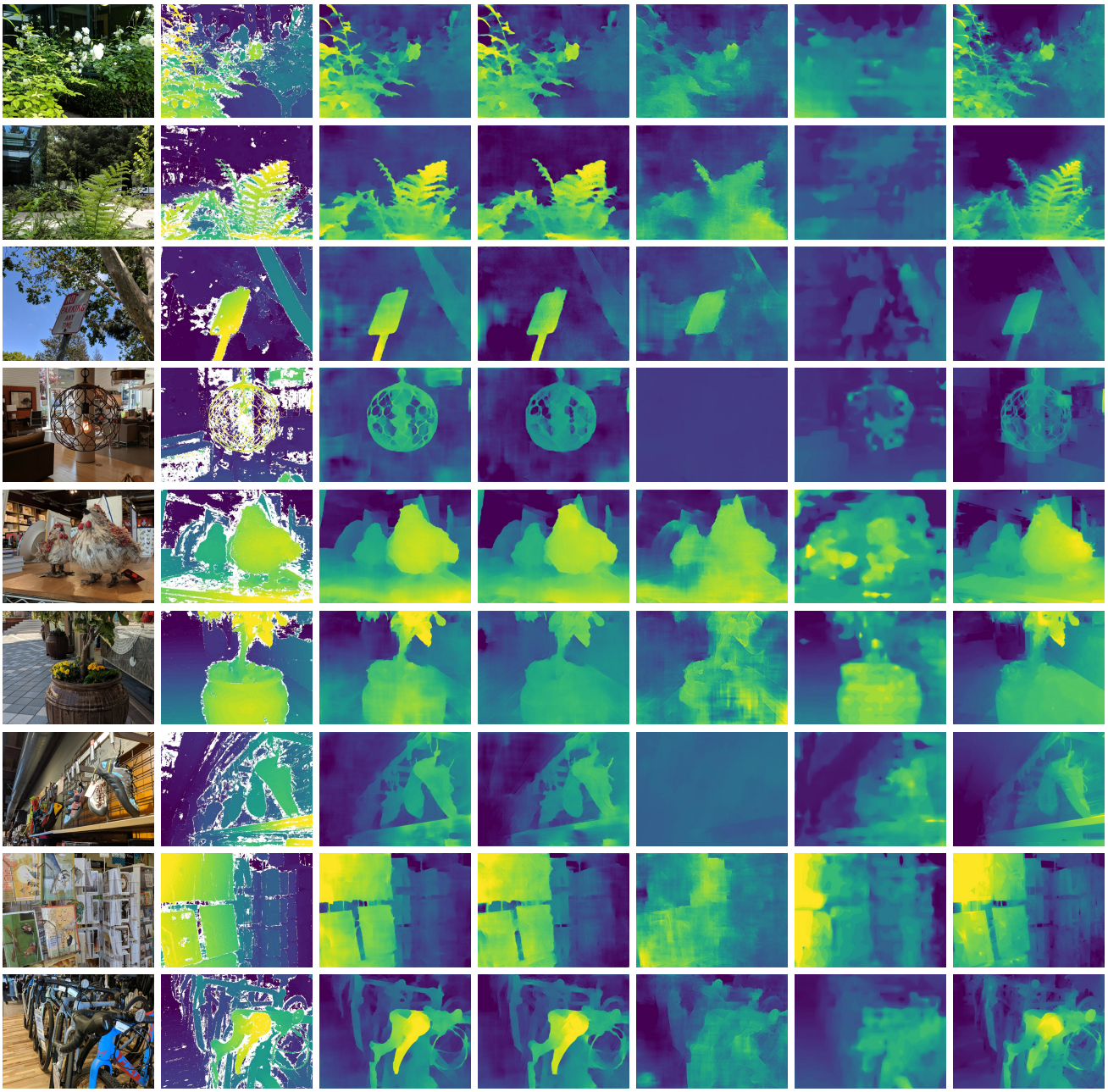
[7] Johannes Kopf, Michael F. Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. *ACM TOG*, 2007.

[8] Christian Richardt, Douglas Orr, Ian Davies, Antonio Criminisi, and Neil A. Dodgson. Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid. *ECCV*, 2010.

[9] Johannes L. Schönberger, Enliang Zheng, Jan Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. *ECCV*, 2016.

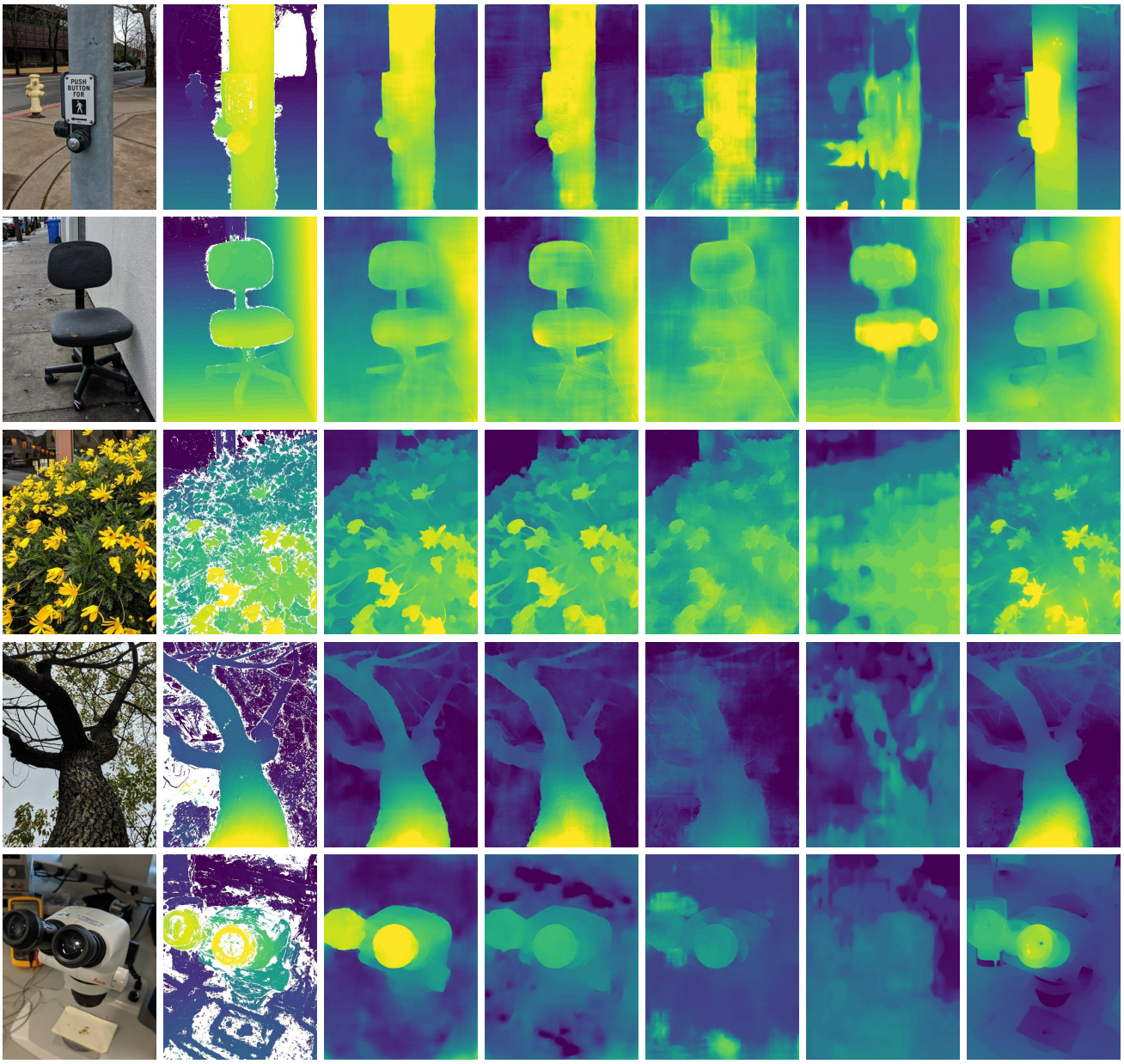
[10] Neal Wadhwa, Rahul Garg, David E. Jacobs, Bryan E. Feldman, Nori Kanazawa, Robert Carroll, Yair Movshovitz-Attias, Jonathan T. Barron, Yael Pritch, and Marc Levoy. Synthetic depth-of-field with a single-camera mobile phone. *SIGGRAPH*, 2018.

[11] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. *CVPR*, 2017.



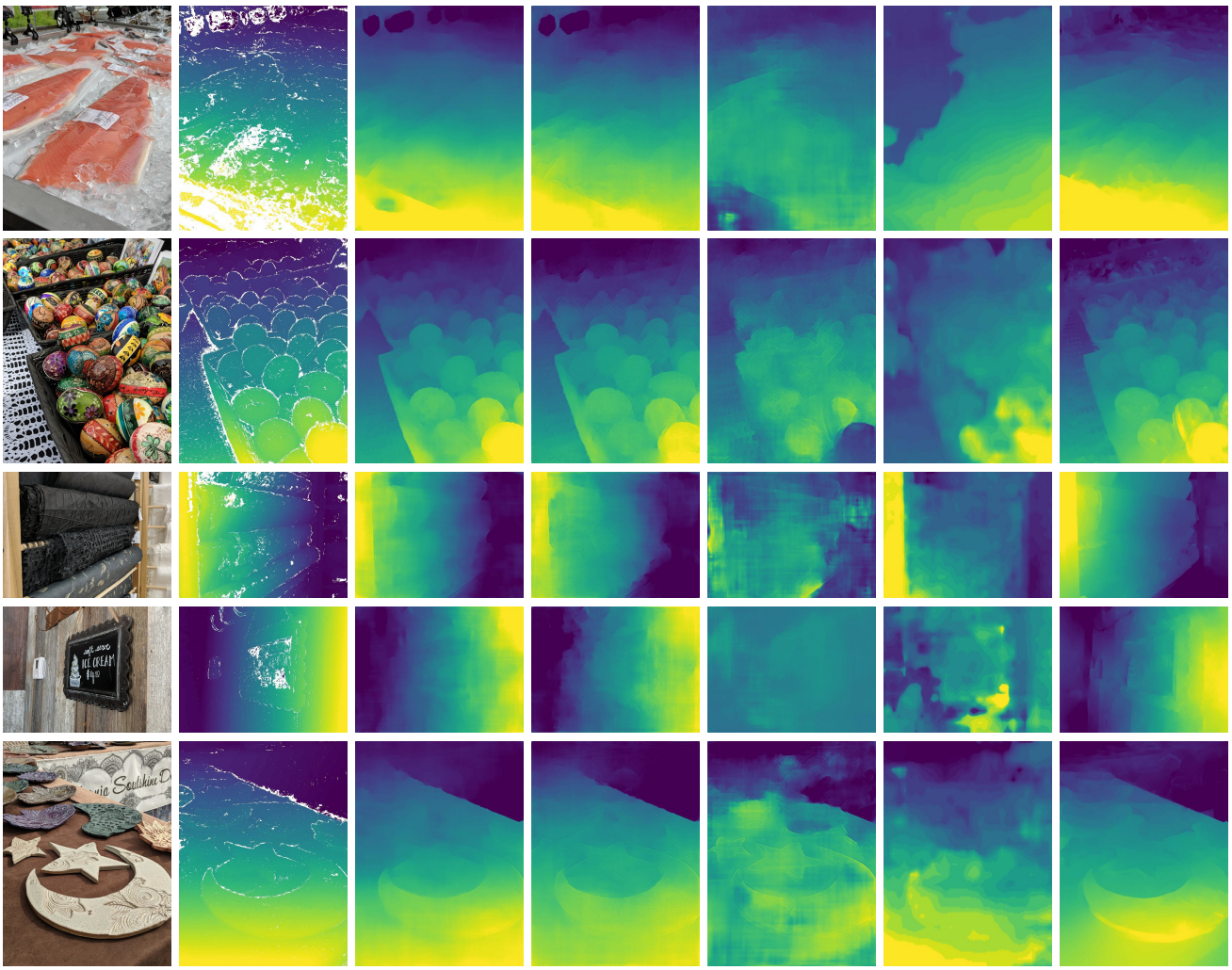
(a) Input Image | (b) GT Depth | (c) DPNet (Affine) RGB + DP | (d) DPNet (Scale) RGB + DP | (e) DPNet (Affine) RGB | (f) DORN [3] RGB | (g) Wadhwa *et al.* RGB + DP [10]

Figure 3. Additional results similar to Table 5 in the main paper.



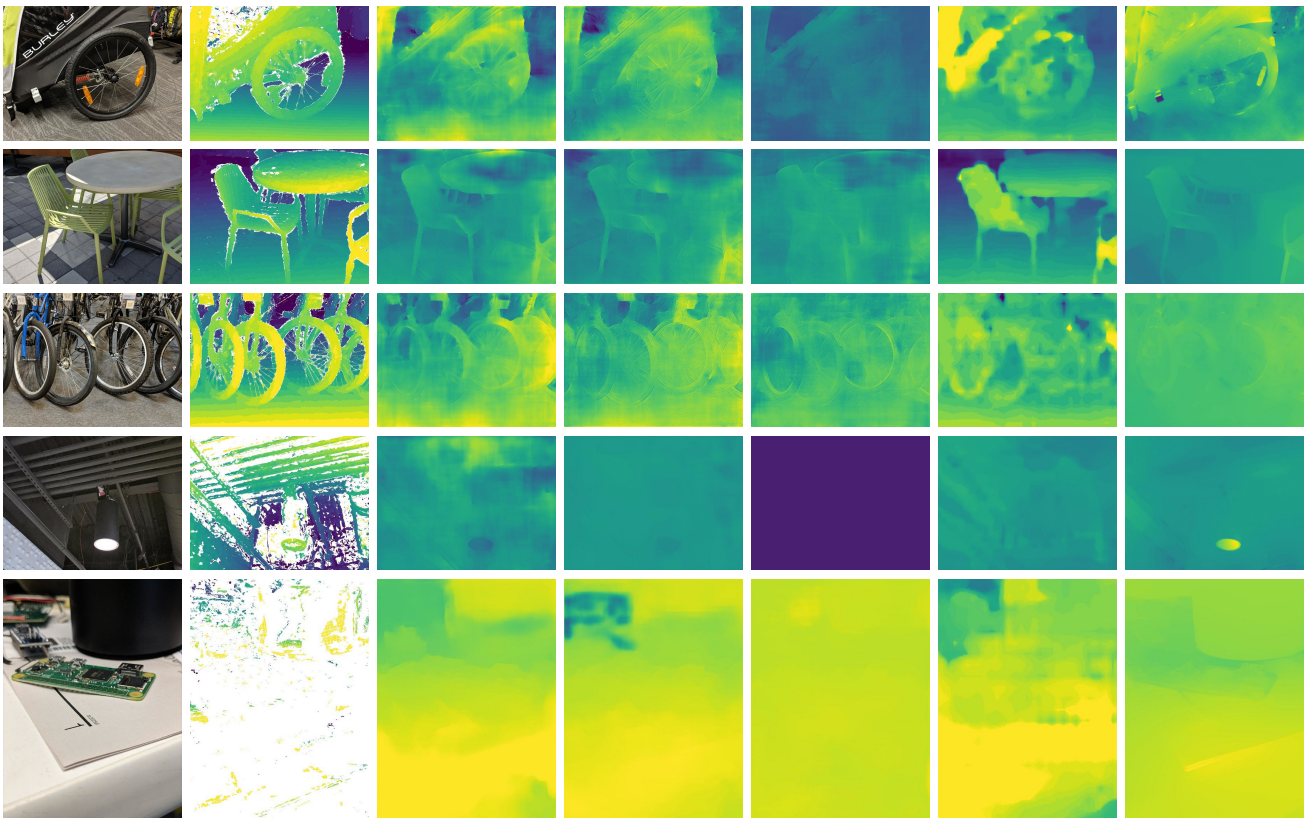
(a) Input Image | (b) GT Depth | (c) DPNet (Affine) RGB + DP | (d) DPNet (Scale) RGB + DP | (e) DPNet (Affine) RGB | (f) DORN [3] RGB | (g) Wadhwa *et al*. RGB + DP [10]

Figure 4. Additional results similar to Table 5 in the main paper.



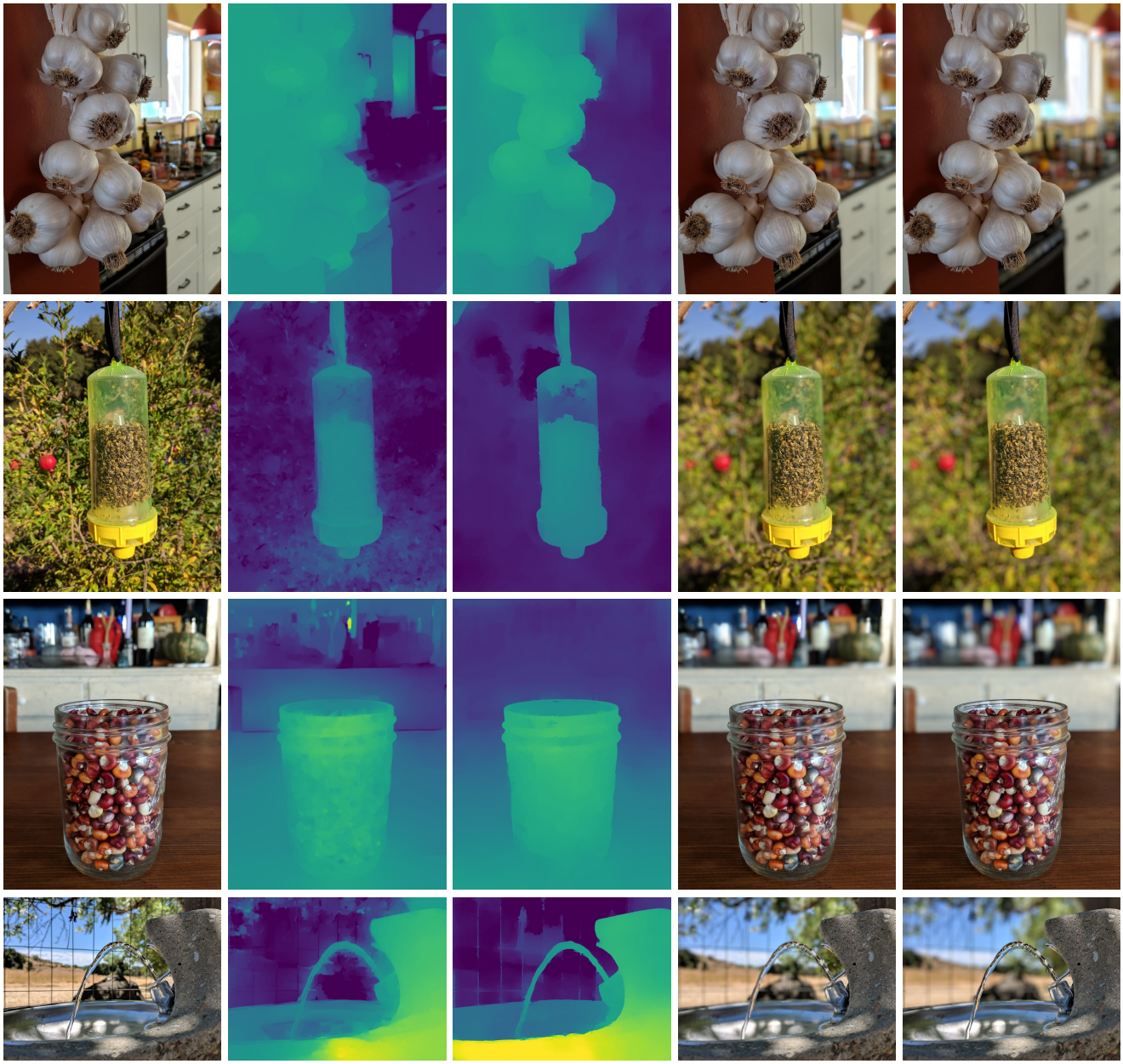
(a) Input Image | (b) GT Depth | (c) DPNet (Affine) RGB + DP | (d) DPNet (Scale) RGB + DP | (e) DPNet (Affine) RGB | (f) DORN [3] RGB | (g) Wadhwa *et al.* RGB + DP [10]

Figure 5. Top 5 results of our method (DPNet with affine invariance) as determined by $1 - |\rho_s|$ metric. These tend to be textured scenes with close focus distances.



(a) Input Image | (b) GT Depth | (c) DPNet (Affine) RGB + DP | (d) DPNet (Scale) RGB + DP | (e) DPNet (Affine) RGB | (f) DORN [3] RGB | (g) Wadhwa *et al*. RGB + DP [10]

Figure 6. Worst 5 results of our method (DPNet with affine invariance) as determined by $1 - |\rho_s|$ metric. These tend to be scenes with textureless surfaces or containing far off objects. For the fourth image in column (e), a particularly bad prediction results in an incorrect affine mapping that collapses all depths to a single value.



(a) Input Image (b) Depth from [10] (c) Our Depth (d) Defocus result from [10] (e) Our defocus result

Figure 7. Using our learned model to predict depth and render synthetic defocus results on data from Wadhwa *et al*. [10]. While depth maps from [10] contain errors, especially on saturated and optically blurred regions, our depth maps are clean and produce pleasing synthetic defocus results. E.g., notice the sharp highlights in the background in the first and the third image, uneven background blur in second image, and incorrect foreground blur in final image in results from [10].