Supplemental Material : A Structure-Aware Method for Direct Pose Estimation

Hunter Blanton[1] Scott Workman[2] Nathan Jacobs[1]
[1]University of Kentucky [2]DZYNE Technologies

1. Additional Results on Indoor Scenes

We show per-scene histograms of error for the 7Scenes dataset in Figure 1. We also show the result of training with or without the validity mask in L_{geom} in Figure 2.

We show qualitative results on images from the 12Scenes dataset in Figure 4 and Figure 5. Compared to the 7Scenes dataset, it is less common for images to contain large regions that have no depth information during training. As such, the weighting CNN produces more varied weights in general because it can no longer depend on obviously incorrect points. This can result in seemingly strange results, such as row 5 in Figure 4 and row 12 in Figure 5. Note, however, that even with these strange correspondence weights, the final re-projection error is typically low.

2. Results on Outdoor Scenes

We show quantitative results on common outdoor scenes from the Cambridge Landmarks dataset [8] in Table 3. While our approach does not work as well in this scenario as it does for the indoor scenes from the main paper due to the low quality depth labels, it still performs competitively on all scenes, and is the best method for the Hospital scene by a large margin. On average our method is best for position error, but the ResNet based PoseNet [7] performs best on orientation error. This is surprising since PoseNet was not competitive even against other similar methods on 7Scenes. This shows the difficulty of this dataset for direct pose estimation methods. Note that this is among the most challenging scenarios for our method because of the poor quality of the depth labels generated from rendering from sparse structure-from-motion (SfM) keypoints. Due to the explicit nature of our method, utilizing a better SfM tool to generate more accurate depth images will directly lead to better performance. Examples of depth labels found in the dataset are shown in Figure 6. While there are many areas that have missing depth (depth=0), this is not an issue for our method as we can ignore these pixels during training. However, there are a large number of sky pixels which are incorrectly labeled with depth, as well as erroneous depth values in general. These errors are an issue for our method because they result in incorrect supervision during training. However, as mentioned earlier, even with these labels our method performs well, and there is a clear path to improvement from better label generation alone.

Due to the poor depth quality and fewer training examples per scene, we use the masked version of L_{geom} and train for more epochs compared to indoor scenes. For the geometry optimization phase, we train for 100 epochs with an initial learning rate of $1e^{-4}$ and reduce the learning rate by a factor of 0.5 every 40 epochs. For the pose optimization phase, we train for 20 epochs with a learning rate of $1e^{-3}$ on the weighting CNN parameters and $1e^{-4}$ on the depth and scene coordinate CNN parameters. The higher learning rate on the geometry prediction parameters is similar to the the re-projection error optimization phase of DSAC++ [2] due to the error in ground-truth scene coordinate labels.

We show visualizations of network outputs on several Cambridge Landmarks inputs in Figure 7. Notice that even in areas where depth and scene coordinate predictions seem good, the predicted weights tend to focus on a smaller area. This is apparent mostly in the Kings College scene, examples of which are shown in rows 2 and 8. Also, in row 6 we can see a difficult case where most of the image is a tree, leading to bad predictions. This is reflected in the weights as all predicted correspondence weights for this example are very low. Overall, even with the noisy depth labels, the weighting mechanism is able to capture which points are more reliable for final pose computation.

3. Depth Accuracy

A key part of our method is the choice of depth estimation network. While we could have chosen a large network and trained it for generic depth estimation, we instead chose a more shallow network and trained on a per-scene basis for pose estimation. Table 4 shows the average depth error for each scene. We compare the depth estimation accuracy in the case of 1) training with a single scene, 2) training with all scenes, and 3) holding out the scene. This last case tests

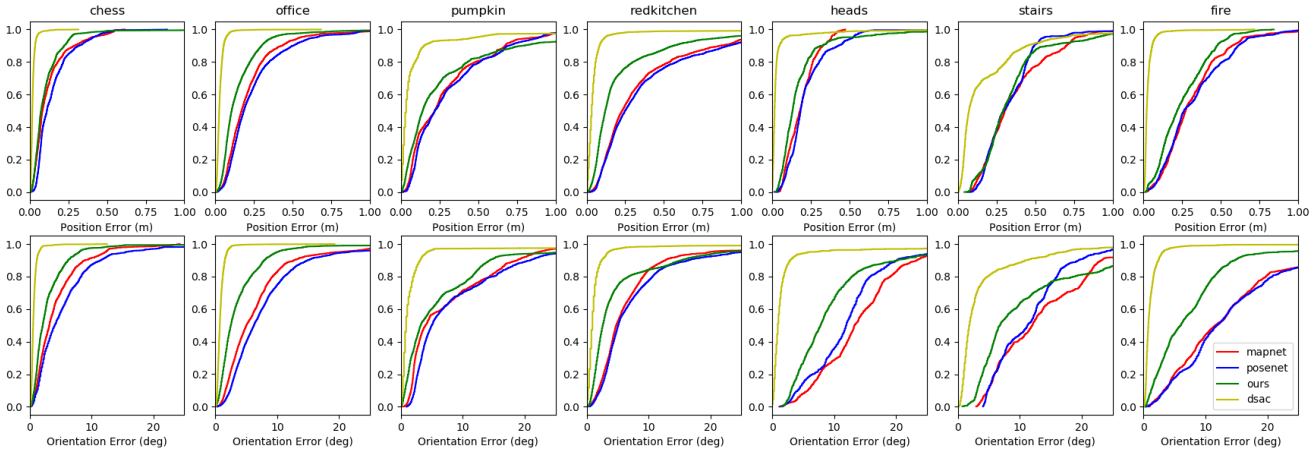

Figure 1: Individual cumulative histograms of error for each scene from 7Scenes for several methods, truncated to 1 meter and 25 degrees error.

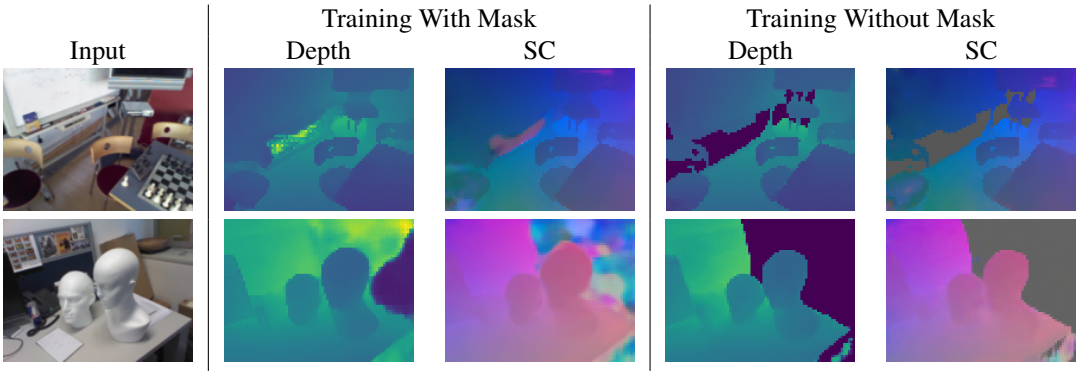

Figure 2: Qualitative results from training with or without a validity mask: (left) input image, (middle) output depth and scene coordinates (SC) trained using the validity mask, and (right) outputs trained without this mask. Note that these are not ground-truth labels. Training without the mask forces the network to recognize unknown areas.

the potential ability of our depth networks to transfer to other scenes. We report mean absolute error, as well as depth accuracy values for different error thresholds. As expected, we typically observe a gradual decline in depth estimation quality as we move from the single scene case where much of the scene structure can be memorized, to the held out case, where no information about the scene was observed during training. Also, we show that a high percentage of pixels have a depth error of less than 0.125 meters, so we believe our simple network is sufficient for this task.

4. Extending to Multiple Scenes

Multi-scene absolute pose regression was first explored by Blanton et al. [1]. They show that a PoseNet style architecture can be separated into scene-specific and scene-agnostic components. This method is called MSPN. We show our method used for the multi-scene pose regression task.

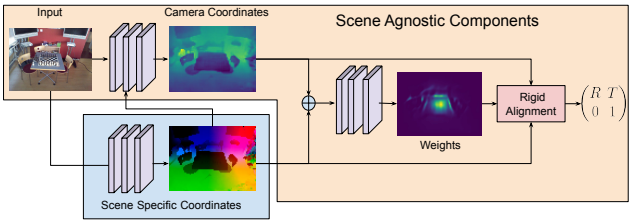

Figure 3: Our absolute pose estimation pipeline for absolute pose regression in multiple scenes. The scene coordinate regression network is the only component that is dependent on the scene. All other parts of the network, namely depth estimation, correspondence weighting, and rigid alignment are generic and are applied to all scenes.

To do this, we slightly alter the architecture as shown in Figure 3. The key changes are that the scene coordinate

	Sequence						
Method	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs
PoseNet (all)	0.15/4.85	0.28/13.13	0.30/**11.54**	0.23/6.34	0.29/5.34	0.29/6.98	0.35/10.63
MSPN	0.09/4.76	0.29/10.50	**0.16**/13.10	0.16/6.80	0.19/5.50	0.21/6.61	0.31/11.63
Ours	**0.07/2.00**	**0.15/5.19**	0.19/12.29	**0.12/3.40**	**0.14/4.00**	**0.13/3.69**	0.39/9.99

Table 1: Results on multi-scene training. Each method was trained on all scene simultaneously.

	Chess	Fire	Heads	Office	Pumpkin	Red Kitchen	Stairs
MSPN (finetune)	0.82/23.28	0.76/31.39	0.44/23.15	0.98/ 45.69	0.76/29.86	1.32/33.37	0.69/32.95
EPnP [9]	**0.12/2.96**	**0.28/7.58**	1.04/60.68	0.48/9.05	**0.21/4.32**	0.31/6.88	0.58/10.25
Ours	0.24/6.58	0.32/10.40	**0.29/18.97**	**0.28/6.31**	0.35/7.81	**0.28/6.51**	**0.37/8.28**

Table 2: Pose error results for held out scenes (median position in meters/median orientation in degrees). In this case, each scene was evaluated using depth and weighting networks trained without examples from the given scene. Our method significantly outperforms MSPN, and manages to beat EPnP in many cases.

regression no longer shares parameters with the depth estimation network, and the scene coordinate estimates are fed into the depth estimation network. For training, we train the scene coordinate regressor separately, then train the depth and weighting network on all scenes simultaneously. We use pretrained ESAC [3] networks to show that this method can be applied to any scene coordinate estimates.

Results for multi-scene training are given in Table 1. Compared even to single scene methods, our proposed approach is very good. Finally, we evaluate on scenes held out of training for the scene-agnostic components (i.e., the depth and weighting networks). Results are shown in Table 2. Compared to MSPN, our method actually generalized to novel scenes without requiring retraining of the entire network. It performs competitively with EPnP [9] without RANSAC applied directly to the scene coordinates.

5. Architecture Details

We use a standard ResNet-34 [6] as the backbone feature extractor for our network. We use the implementation from the Pytorch [11] torchvision library. The ResNet feature extractor has 5 major components: a single convolution and max pooling (ConvBlock) followed by 4 residual blocks (ResBlock). The general ResNet architecture is given in Table 5. We only use the convolutional and residual layers, not the final linear layer.

We provide detailed architecture components for the other components of our network, namely the scene coordinate regression network (Table 6), the depth network (Table 7), and the weighting network (Table 8). These networks make use of intermediate outputs from the shared ResNet backbone which are given in the "Input" column. For convolutions, we list it as Conv2d(in channels, out channels, kernel size, stride, padding).

References

[1] Hunter Blanton, Scott Workman, and Nathan Jacobs. Extending absolute pose regression to multiple scenes. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2020. 2

[2] Eric Brachmann and Carsten Rother. Learning less is more - 6d camera localization via 3d surface regression. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1

[3] Eric Brachmann and Carsten Rother. Expert sample consensus applied to camera re-localization. In *International Conference on Computer Vision*, 2019. 3

[4] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. Geometry-aware learning of maps for camera localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 7

[5] Ming Cai, Chunhua Shen, and Ian D. Reid. A hybrid probabilistic model for camera relocalization. In *British Machine Vision Conference*, 2018. 7

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 3

[7] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 7

[8] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. 2015. 1, 7

[9] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International Journal of Computer Vision*, 81(2):155, 2009. 3

[10] Tayyab Naseer and Wolfram Burgard. Deep regression for monocular camera-based 6-dof global localization in outdoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017. 7

[11] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019. 3

[12] Florian Walch, Caner Hazirbas, Laura Leal-Taixe, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. Image-based localization using lstms for structured feature correlation. In *International Conference on Computer Vision*, 2017. 7

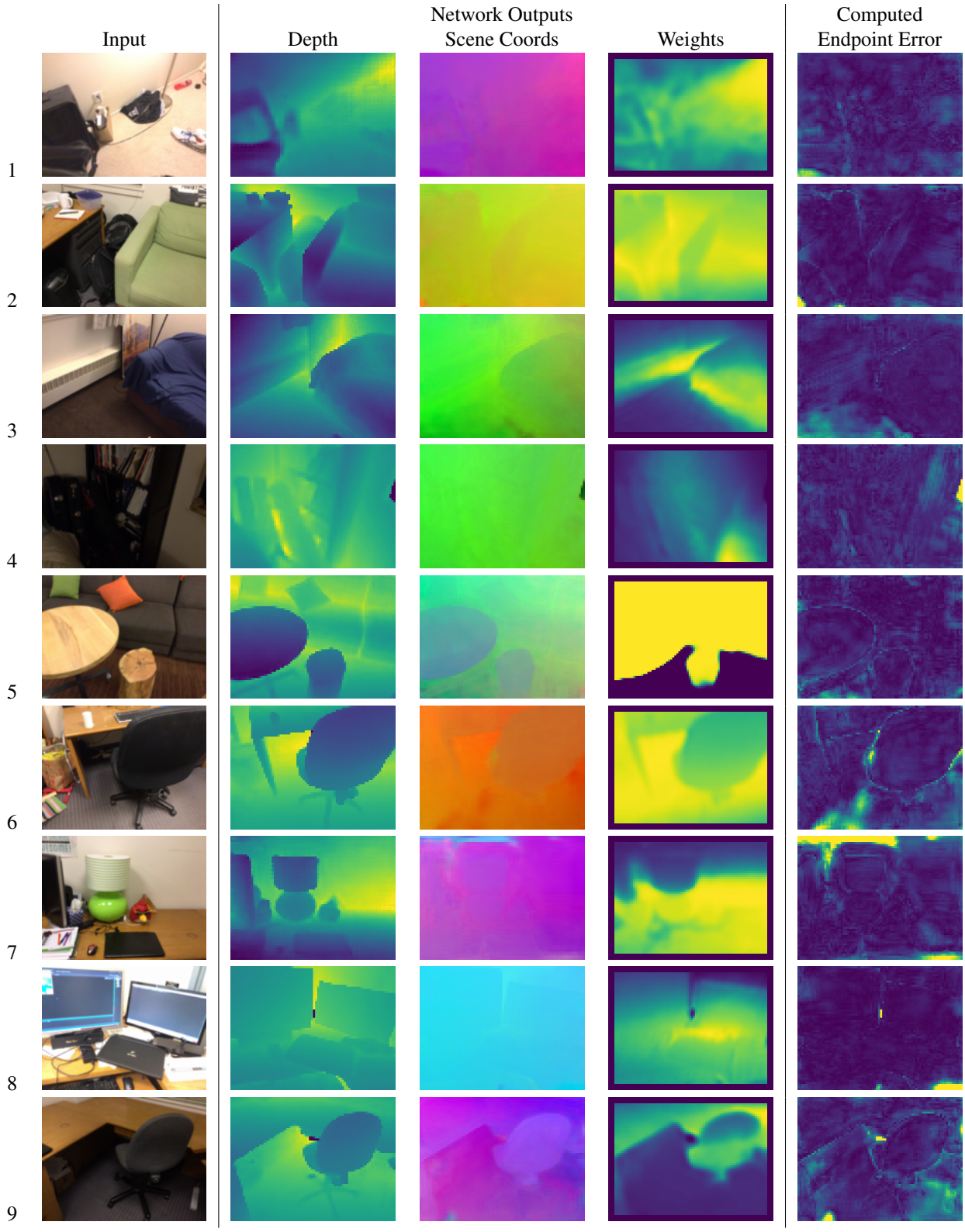

Figure 4: Example network outputs for 12Scenes images. Depth, scene coordinates, and weights are direct outputs of our network. The endpoint error is computed by applying the regressed pose to the scene coordinates and clamped to 1 for visualization. For depth, weights, and endpoint error, brighter means a higher value. Row labels are shown on the left for referencing in text.

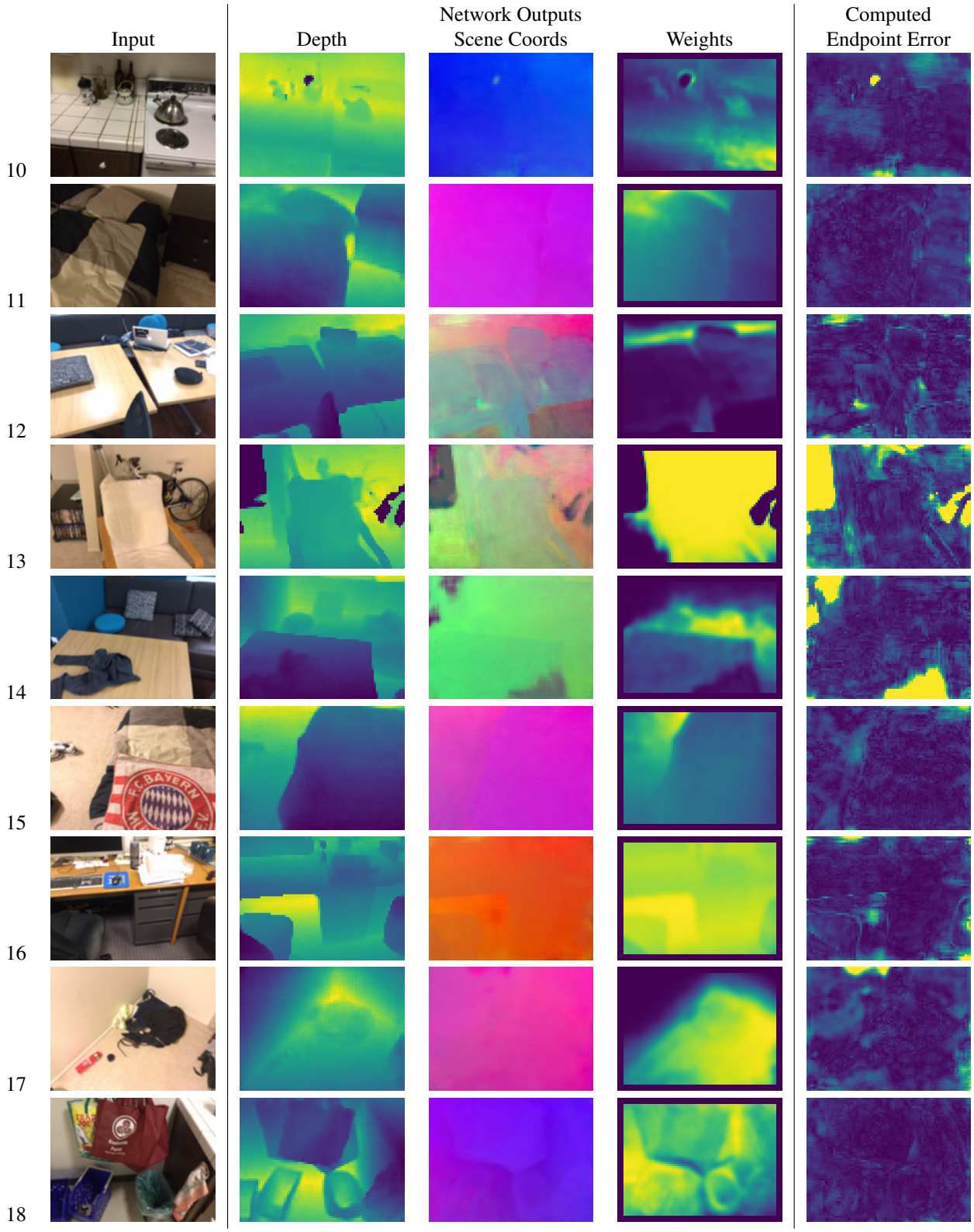

Figure 5: More example network outputs for 12Scenes images. Depth, scene coordinates, and weights are direct outputs of our network. The endpoint error is computed by applying the regressed pose to the scene coordinates and clamped to 1 for visualization. For depth, weights, and endpoint error, brighter means a higher value. Row labels are shown on the left for referencing in text.

Figure 6: Examples of errors in depth labels for Cambridge Landmarks images. Many areas that should have invalid depth are assigned depth values. There are many incorrect depth assignments in general. Dark areas are regions with no depth label (depth=0).

	Sequence				
Method	College	Hospital	Shop	Church	Avg
PoseNet [8]	1.92/5.40	2.31/5.38	1.46/8.08	2.65/8.48	2.08/6.83
PoseNet Learned Weights [7]	0.99/1.06	2.17/2.94	1.05/3.97	1.49/3.43	1.43/**2.85**
Geo PoseNet [7]	**0.88/1.04**	3.20/3.29	0.88/**3.78**	1.57/**3.32**	1.63/2.86
LSTM PoseNet [12]	0.99/3.65	1.51/4.29	1.18/7.44	1.52/6.68	1.30/5.51
GPoseNet [5]	1.61/2.29	2.62/3.89	1.14/5.73	2.93/6.46	2.08/4.59
SVS-Pose [10]	1.06/2.81	1.50/4.03	**0.63**/5.73	2.11/8.11	1.32/5.17
MapNet [4]	1.07/1.89	1.94/3.91	1.49/4.22	2.00/4.53	1.62/3.64
Ours	1.19/2.16	**1.11/1.92**	0.95/6.82	**1.37**/4.45	**1.16**/3.84

Table 3: Direct pose estimation results on Cambridge Landmarks compared to other methods (median position in meters/median rotation in degrees).

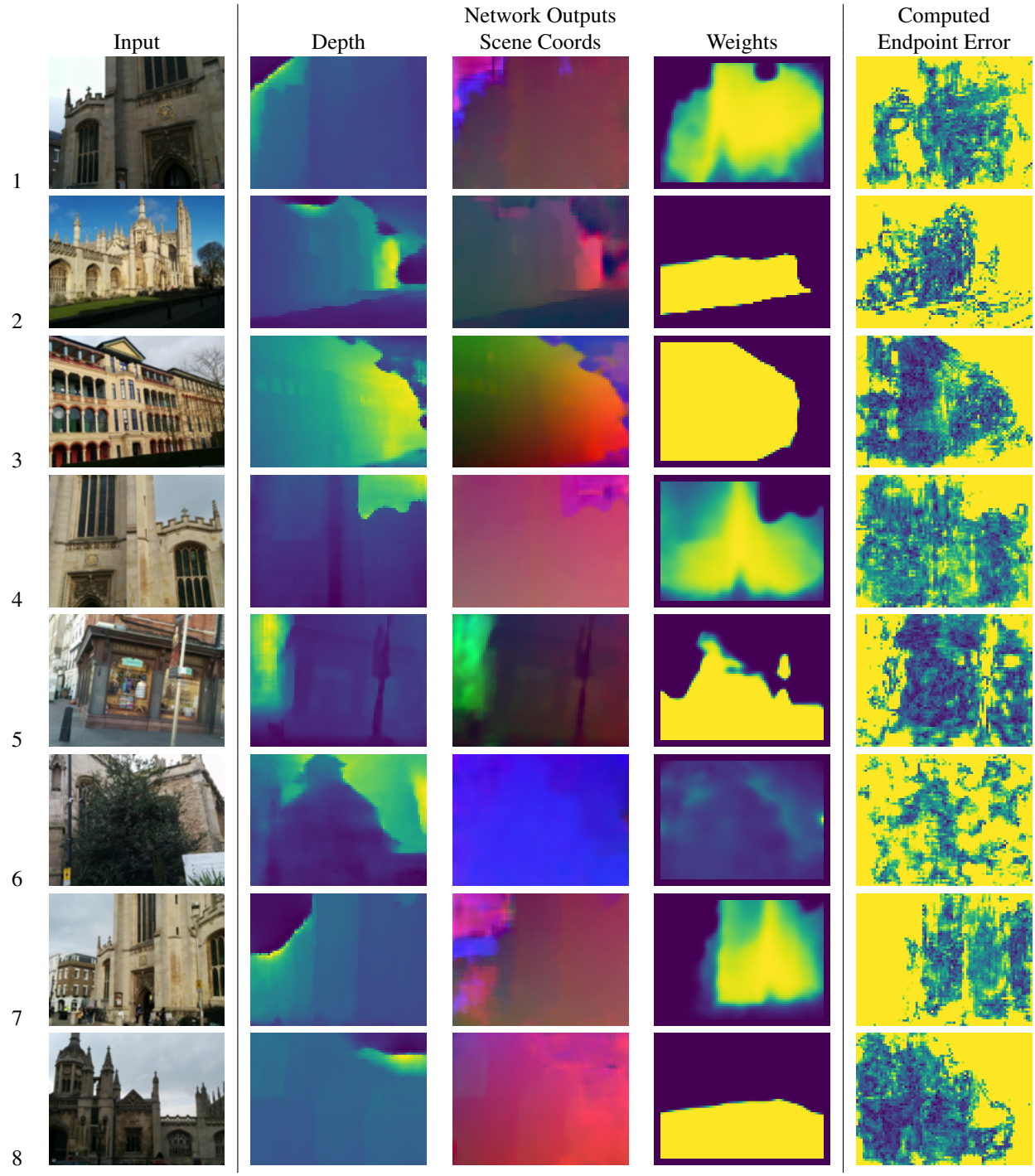

Figure 7: Example network outputs for Cambridge Landmarks images. Depth, scene coordinates, and weights are direct outputs of our network. The endpoint error is computed by applying the regressed pose to the scene coordinates and clamped to 1 for visualization. For depth, weights, and endpoint error, brighter means a higher value. Row labels are shown on the left for referencing in text.

	Sequence						
Method	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs
Single Scene							
Abs Error	0.2318	0.1954	0.1491	0.2367	0.2244	0.2570	0.3515
$\delta < 0.5^3$	0.4883	0.4743	0.6206	0.3651	0.4164	0.3616	0.3260
$\delta < 0.5^2$	0.7482	0.7461	0.8371	0.6554	0.7030	0.6349	0.5610
$\delta < 0.5$	0.8968	0.9364	0.9385	0.9011	0.9261	0.8870	0.7959
All Scene							
Abs Error	0.2271	0.1842	0.1730	0.2325	0.2591	0.2654	0.4374
$\delta < 0.5^3$	0.4909	0.4945	0.5817	0.3876	0.3599	0.3580	0.2559
$\delta < 0.5^2$	0.7499	0.7710	0.7842	0.6694	0.6607	0.6280	0.4535
$\delta < 0.5$	0.8980	0.9493	0.9216	0.9029	0.9044	0.8787	0.7196
Held Out							
Abs Error	0.2768	0.2157	0.2425	0.3333	0.3549	0.3386	0.4800
$\delta < 0.5^3$	0.3759	0.4060	0.3014	0.2212	0.1996	0.2615	0.2316
$\delta < 0.5^2$	0.6372	0.6893	0.5940	0.4526	0.4479	0.4814	0.4173
$\delta < 0.5$	0.8666	0.9156	0.9119	0.8048	0.8251	0.7910	0.6766

Table 4: Depth error on each scene in three scenarios: training with only one scene, training with all 7 scenes, and holding one scene out during training. Results are reported in meters.

ConvBlock1
ResBlock1
ResBlock2
ResBlock3
ResBlock4
Linear Layer

Table 5: ResNet

Name	Operation	Input
conv1	Conv2d(256,256,1,1,0)	ResBlock2
act-conv1	ReLU	conv1
conv2	Conv2d(256,256,1,1,0)	act-conv1
act-conv2	ReLU	conv2
scene-coords	Conv2d(256,3,1,1,0)	act-conv2

Table 6: Scene Coordinate Regression CNN

Name	Operation	Input
conv1	Conv2d(512, 128, 1,1,0)	ResBlock4
bn-conv1	BatchNorm	conv1
act-conv1	ReLU	bn-conv1
tp-conv2	ConvTranspose2d(128, 128, 3,2,1)	act-conv1
bn-tp-conv2	BatchNorm	tp-conv2
act-tp-conv2	ReLU	bn-tp-conv2
conv3	Conv2d(128, 256, 1,1,0)	act-tp-conv2
bn-conv3	BatchNorm	conv3
act-conv3	ReLU	bn-conv3
conv4	Conv2d(256, 64, 1,1,0)	ResBlock3+act-conv3
bn-conv4	BatchNorm	conv4
act-conv4	ReLU	bn-conv4
tp-conv5	ConvTranspose2d(64, 64, 3,2,1)	act-conv4
bn-tp-conv5	BatchNorm	tp-conv5
act-tp-conv5	ReLU	bn-tp-conv5
conv6	Conv2d(64, 128, 1,1,0)	act-tp-conv5
bn-conv6	BatchNorm	conv6
act-conv6	ReLU	bn-conv6
Half Res Depth Net		
half-depth-conv1	Conv2d(256,256,1,1,0)	ResBlock3+act-conv3
act-half-depth-conv1	ReLU	half-depth-conv1
half-depth-conv2	Conv2d(256,256,1,1,0)	act-half-depth-conv1
act-half-depth-conv2	ReLU	half-depth-conv2
half-depth-conv3	Conv2d(256,1,1,1,0)	act-half-depth-conv2
hald-depth	Sigmoid()	half-depth-conv3
Depth Net		
depth-conv1	Conv2d(128,128,1,1,0)	ResBlock2+act-conv6
act-depth-conv1	ReLU	depth-conv1
depth-conv2	Conv2d(128,128,1,1,0)	act-depth-conv2
act-depth-conv2	ReLU	depth-conv2
depth-conv3	Conv2d(128,1,1,1,0)	act-depth-conv2
depth	Sigmoid()	depth-conv3

Table 7: Depth CNN

Name	Operation	Input
conv1	Conv2d(6,64,3,1,0)	cat(Scene Coordinates, Camera Coordinates)
act-conv1	ReLU	conv1
conv2	Conv2d(64,128,3,1,0)	act-conv1
act-conv2	ReLU	conv2
conv3	Conv2d(128,512,3,1,0)	act-conv2
act-conv3	ReLU	conv3
weights	Conv2d(512,1,1,1,0)	act-conv3

Table 8: Weighting CNN