Learning Less is More – 6D Camera Localization via 3D Surface Regression – Supplementary Materials –

Eric Brachmann and Carsten Rother
Visual Learning Lab
Heidelberg University (HCI/IWR)
http://vislearn.de

In this document, we list the main parameter settings for executing (Sec. 1.1) and learning (Sec. 1.2) our pipeline. The information provided here is not necessary to understand the main paper but can be useful for re-implementation. We also include additional experimental analysis (Sec. 2) which did not fit in the main paper due to space constraints.

1. Parameters

We use the same parameter settings for all scenes, indoor and outdoor, expect for the scene coordinate initialization parameter d. See details below.

1.1. Pipeline Parameters

Our FCN [3] network architecture (see also Fig. 1) takes an image of 640×480 px as input. We re-scale larger images to 480 px height. Should an image be wider than 640 px after re-scaling, we crop it centrally to 640 px width. After scene coordinate prediction, we sample $n = 256$ hypotheses using random 4-tuples of points and the algorithm of [1]. We reject hypotheses where the reprojection error of the corresponding 4-tuple is larger than the inlier threshold τ, and sample again. We set the inlier threshold $\tau = 10$px for all scenes. For the soft inlier count (Eq. 5 of the main paper) we use a softness factor $\beta = 0.5$. We refine the selected hypothesis until convergence or for a maximum of 100 iterations.

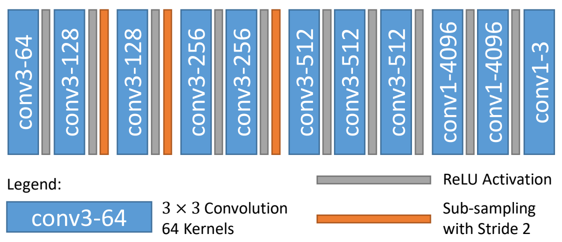

Figure 1. **Network Architecture.** Our FCN network architecture is composed of 3×3 convolutions, ReLU activations and sub-sampling via strided convolutions. The final fully connected layers are implemented via 1×1 convolutions.

1.2. Learning Parameters

Our FCN [3] network architecture predicts 80×60 scene coordinates for a 640×480 input image, *i.e.* it predicts one scene coordinate for each 8×8 px image block. To make full use of the training data, we randomly shift training images by a maximum of 8 px, horizontally and vertically. We re-scale training images to 480 px height. Should an image be wider than 640 px after re-scaling, we crop it to 640 px width using random horizontal offsets.

We optimize using ADAM [2] and a batch size of 1 image. The remaining learning hyper-parameters differ for the three different training steps described in the main paper.

Scene Coordinate Initialization. We use an initial learning rate of 10^{-4}, and train for 300k iterations. After 100k iterations we halve the learning rate every 50k iterations.

When initializing the pipeline using our scene coordinate heuristic instead of rendered scene coordinates, we reduce training to 100k iterations and utilize only 5% of the training data. This is to avoid overfitting to the heuristic. For the heuristic, we use a constant depth prior of $d = 3$m for indoor scenes, and $d = 10$m for outdoor scenes.

Optimization of Reprojection Error. We use an initial learning rate of 10^{-4}, and train for 300k iterations. After 100k iterations we halve the learning rate every 50k iterations. We clamp gradients to ± 0.5 before passing them to the network.

End-to-End Optimization. We use an initial learning rate of 10^{-6}, and train for 50k iterations. We halve the learning rate after 25k iterations. We clamp gradients to $\pm 10^{-3}$ before passing them to the network.

For our entropy control schema, we set scale parameter $\alpha = 0.1$, initially. We optimize using ADAM [2] and a learning rate of 10^{-3} for a target entropy of $S^* = 6$ bit.

2. Additional Analysis

Detailed Quantitative Results. See Table 1 (left) for the accuracy of our pipeline for each scene of the 7Scenes dataset [4], when trained with and without a 3D scene

Table 1. **Detailed Results for 7Scenes.** We report accuracy per scene as percentage of estimated poses with an error below 5cm and 5°. "Complete" denotes the accuracy on all test frames of the dataset combined. **Left:** Results of training the pipeline with and without a 3D model. Numbers are given after end-to-end training. **Right:** Effect of varying the output resolution of scene coordinate regression (default: 80x60). Numbers are given for training with a 3D model and after end-to-end training.

	Training		Scene Coordinate Output Resolution					
	w/ 3D Model	w/o 3D Model	20x15	20x30	40x30	40x60	80x60	320x240
Chess	97.1%	93.8%	96.5%	97.1%	97.3%	97.4%	97.1%	97.6%
Fire	89.6%	75.6%	86.0%	87.8%	89.5%	90.4%	89.6%	91.9%
Heads	92.4%	18.4%	88.6%	89.9%	91.3%	92.1%	92.4%	93.7%
Office	86.6%	75.4%	83.3%	85.2%	86.2%	86.1%	86.6%	87.3%
Pumpkin	59.0%	55.9%	58.2%	60.5%	61.3%	61.3%	59.0%	61.6%
Kitchen	66.6%	50.7%	63.2%	64.3%	64.7%	64.7%	66.6%	65.7%
Stairs	29.3%	2.0%	19.7%	22.5%	23.7%	25.1%	29.3%	28.7%
Complete	**76.1%**	**60.4%**	**72.8%**	**74.4%**	**75.2%**	**75.5%**	**76.1%**	**76.6%**

model. Table 2 shows the corresponding results for the 12Scenes dataset [5].

Varying Output Resolution. We analyze the impact of the FCN output resolution, *i.e.* the number of scene coordinates predicted, on pose estimation accuracy. The default output resolution of our FCN architecture is 80×60. We simulate smaller output resolutions by sub-sampling correspondences during test time. We simulate higher output resolutions by executing the FCN multiple times with shifted inputs. See results in Table 1 (right) for the 7Scenes dataset. We observe a graceful decrease in accuracy with smaller output resolutions. Therefore, the runtime of the pipeline could potentially be optimized by predicting less scene coordinates while maintaining high accuracy. On the other hand, we observe only a small increase in accuracy with higher output resolutions, *i.e.* when the FCN predicts more scene coordinates. Therefore, we do not expect an advantage in using up-sampling layers to produce full resolution outputs.

Scene Coordinate Initialization. When training our pipeline without a 3D scene model, we initialize scene coordinates to have a constant distance d from the camera plane, see Sec. 2.4 of the main paper. We set this value to $d = 3$m for indoor scenes and $d = 10$m for outdoor scenes, according to the coarse range of depth values we expect for these settings. Without this initialization, scene coordinate predictions might lie behind the camera or near the projection center in the beginning of training, resulting in unstable gradients and very low test accuracy. We found the aforementioned values for d to generalize well on the diverse set of scenes we experimented on. However, setting d to a value that is far off the actual range of distances can harm accuracy. For example, when setting $d = 10$m for the 7Scenes dataset, test accuracy decreases to 49.3%.

Learning Scene Geometry. We visualize the approximate scene geometry discovered by our system when trained without a 3D model in Fig. 2. Although our heuristic initialization ignores scene geometry entirely, the system recovers depth information through optimization of reprojection errors.

Table 2. **Detailed Results for 12Scenes.** We report accuracy per scene as percentage of estimated poses with an error below 5cm and 5°. "Complete" denotes the accuracy on all test frames of the dataset combined. Numbers are given for training the pipeline with and without a 3D model, both after end-to-end training.

		Training	
		w/ 3D Model	w/o 3D Model
Apt. 1	Kitchen	100%	7.6%
	Living	100%	92.0%
Apt. 2	Bed	99.5%	66.1%
	Kitchen	99.5%	87.6%
	Living	100%	89.9%
	Luke	95.5%	67.3%
Office 1	Gates 362	100%	96.3%
	Gates 381	96.8%	27.8%
	Lounge	95.1%	94.8%
	Manolis	96.4%	72.2%
Office 2	Floor 5a	83.7%	11.0%
	Floor 5b	95.0%	83.2%
Complete		**96.4%**	**60.9%**

Run Time. The total processing time of our implementation is ≈200ms per image on a Tesla K80 GPU and an Intel Xeon E5-E5-2680 v3 CPU (6 cores). Time is spent mainly for scene coordinate regression, including the overhead of transferring data between the pose estimation front-end (C++) and the deep learning back-end (LUA/Torch), as well as transferring data between main memory and GPU memory. Pose optimization with DSAC takes ≈10ms on the aforementioned CPU. Training the pipeline takes 1-2 days per training stage and scene on a single Tesla K80 GPU.

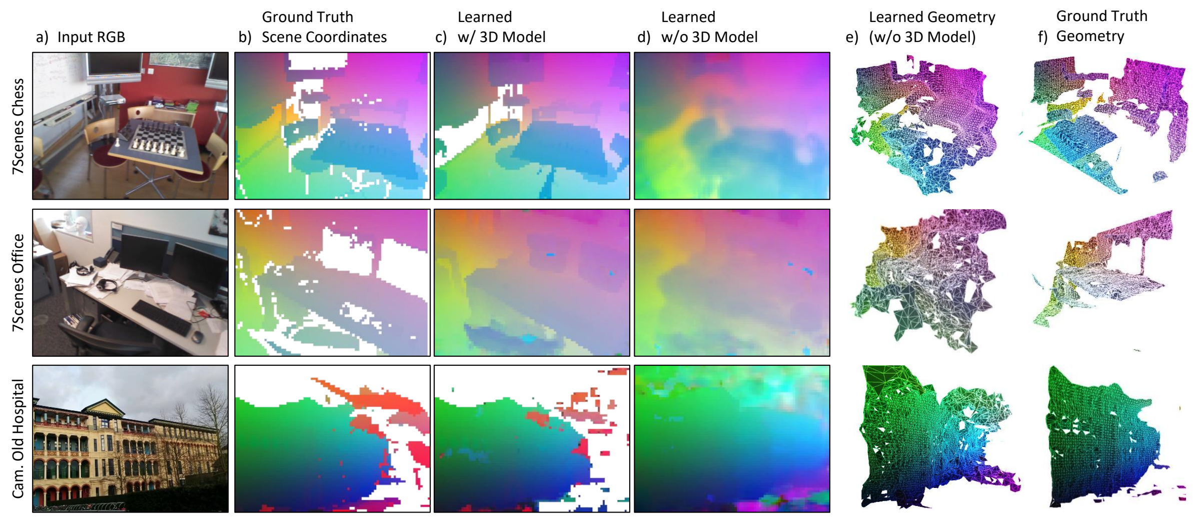

Figure 2. **Learning Scene Geometry.** We show scene coordinate predictions for test images after end-to-end optimization. For comparison, we calculate ground truth scene coordinates (**b**) using measured depth (7Scenes) or the 3D scene model (Cambridge Hospital). When trained using a 3D model (**c**), our method learns the scene geometry accurately. When trained without a 3D model (**d**), our method discovers an approximate geometry, automatically. The last two columns show a 3D mesh representation of column (d) and (b).

References

[1] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. *TPAMI*, 2003. 1

[2] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, 2014. 1

[3] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

[4] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGB-D images. In *CVPR*, 2013. 1

[5] J. Valentin, A. Dai, M. Nießner, P. Kohli, P. Torr, S. Izadi, and C. Keskin. Learning to navigate the energy landscape. *CoRR*, 2016. 2