LaLaLoc - Supplementary Material

Henry Howard-Jenkins
Active Vision Laboratory
University of Oxford
henryhj@robots.ox.ac.uk

Jose-Raul Ruiz-Sarmiento
Machine Perception and
Intelligent Robotics Group
University of Málaga
jotaraul@uma.es

Victor Adrian Prisacariu
Active Vision Laboratory
University of Oxford
victor@robots.ox.ac.uk

In this document, we provide details about implementation such as network architecture and losses. In addition, we perform additional evaluation to complement the results in the main paper.

1. Implementation Details

This section contains precise details about the implementation of LaLaLoc and the baselines that it is compared against in the main paper.

1.1. LaLaLoc Architecture

We implement Φ_{layout} as the convolutional layers from a ResNet18 [1] network, with the first layer replaced with an equivalent taking a 1-channel input. Φ_{image}, on the other hand, is implemented as a ResNet50 [1] network. In each, we remove the default classifier and replace it with an average pooling layer and a single fully connected layer which takes the embedding dimension to 128d, from 512d or 2048d for ResNet18 and ResNet50, respectively. The embedding vectors are all L2 normalised before any retrieval or comparison task.

1.2. Decoder Architecture

The architecture of the decoder that is used to aid training of Φ_{layout} is defined as follows. First, the latent representation is fed into a fully connected layer which expands it to 2048d. This is reshaped to (2×4) with 256 channels. The up-scaling stage is formed of multiple repeated blocks. Each block comprises of a $2\times$ bilinear up-sample, a 2D convolution with kernel size 3, a ReLU non-linearity, and a BatchNorm layer. The number of out channels from each block follows the pattern $(128 \rightarrow 64 \rightarrow 32 \rightarrow 32)$. Finally, depth is predicted with a point-wise convolution which predicts a depth image. With four $2\times$ up-samples the final resolution of the decoded depth image is (32×64).

1.3. Training Routine

We train Φ_{layout} for a total of 20 epochs, with a batch size of 4. We optimise using SGD, with an initial learning rate of 0.01, and decay by a factor of 0.1 after 10 and 15 epochs. For the log-ratio loss, we follow the positive:negative sampling ratio of 1:20 as used in [2], therefore a single minibatch contains a total of 84 layouts. We define positives and negatives by their spatial distance from the anchor: positives less than 0.5m from the anchor; negatives greater than 2m from the anchor.

Φ_{image} is trained for a total of 200 epochs, with a batch size of 64. Due to the dearth of RGB images, only a single image-layout pair is sampled per iteration. We again use a SGD optimiser with a learning rate set to 0.1. This is then decayed by a factor of 0.1 after 100 and 150 epochs. Φ_{layout} is frozen for the whole training process. Worth noting is that, when Φ_{image} is trained with a variant of ℓ_{log_ratio}, the training routine follows that of Φ_{layout}.

1.4. Image Branch Losses

Here we provide the full equations for the additional image losses evaluated in the ablation in the main paper. First, we restate ℓ_{log_ratio} [2], as used for the layout branch:

$$\ell_{log\_ratio}(p,i,j) = \left(log\frac{D(g_i,g_p)}{D(g_j,g_p)} - log\frac{\text{Ch}(C_i,C_p)}{\text{Ch}(C_j,C_p)} \right)^2, \tag{1}$$

where (p,i,j) is a triplet of locations within a floor plan, at which we infer their layouts (L_p, L_i, L_j), $g = \Phi_{layout}(L)$ represents the respective embedding of a layout, C is the back-projection of layout L, $D(\cdot)$ is the Euclidean distance and $Ch(\cdot)$ is the Chamfer distance.

For the training of Φ_{image}, we adapt the loss so that the anchor of the triplet now also has an RGB panorama image $(\{I_p, L_p\}, L_i, L_j)$:

$$\ell'_{log\_ratio}(p,i,j) = \left(log\frac{D(g_i,f_p)}{D(g_j,f_p)} - log\frac{\text{Ch}(C_i,C_p)}{\text{Ch}(C_j,C_p)} \right)^2, \tag{2}$$

where the layout embedding g_p, has now been replaced with the embedding of the panorama captured in the same location, $f_p = \Phi_{image}(I_p)$. Therefore, this loss aims to ensure Φ_{image} captures the relative similarities between layouts.

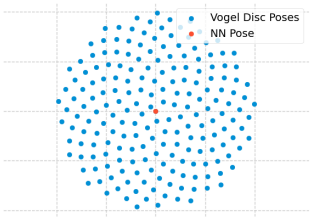

Figure 1. Visualisation of the Vogel Disc sampling pattern with N=200.

We also include a further adaptation, ℓ_{lr_kd}, where we remove ground-truth layout similarities between (p, i, j):

$$\ell_{lr\_kd}(p,i,j) = \left(log\frac{D(g_i, f_p)}{D(g_j, f_p)} - log\frac{D(g_i, g_p)}{D(g_j, g_p)} \right)^2, \quad (3)$$

such that Φ_{image} should instead learn to capture the relative similarities between layout embeddings, rather than the layouts themselves. This loss more closely follows typical knowledge discrimination, as the "student" is trained without ground-truth labels.

1.5. Vogel Disc Re-sampling

Our Vogel Disc re-sampling method offers a retrieval-based refinement to the estimated pose from the nearest neighbour in the coarse sampling grid. Specifically, we sample a circular local region centred at the nearest neighbour pose. For N total samples, the ith location is given by:

$$r_i = \sqrt{i/N} \qquad \theta_i = 2\pi(1 - 1/\phi)$$

where ϕ is the golden ratio, r_i and θ_i are the radius and angle in polar coordinates centred at the nearest neighbour pose. The resulting local sampling is visualised in Figure 1.

1.6. Latent Pose Optimisation

For our latent pose optimisation, we render layouts using Redner [4]. We use an Adam optimiser [3], with initial learning rate set to 0.01. The learning rate is scaled by a factor of 0.5 as the loss plateaus with a threshold of 0.05 and a patience of 10 iterations. Convergence is considered reached after 20 steps with reduction in the cost less than a threshold of 0.001, or until 150 steps have elapsed.

1.7. Iterative Closest Point Baseline

In the following, we detail the implementation of our ICP baseline. We emulate a point cloud obtained from a laser scan, C_m, through the back-projection of furnished depth images, and perform matching between those and the 2D slices to the point cloud from the known floor plan, C_f. At each iteration, the points in C_m are assigned to their nearest counterparts in C_f. The rigid transform, $[R, t]$, that aligns

Method	Inference Time (s) Retr.	VDR	LPO	Total
2D-ICP	-	-	-	20.1
LaLaLoc w/o VDR	0.05	-	2.33 (0.85)	2.38
LaLaLoc	0.05	2.87 (2.71)	1.73 (0.63)	4.65

Table 1. Inference time comparison. Times are given as: Total (Render), where "Render" refers to the amount of time spent rendering layouts from the floorplan.

C_m and C_f is determined by minimising a point-to-point cost function given this assignment:

$$E(R,t) = \sum_{i=1}^{|C_f|} \sum_{j=1}^{|C_m|} w_{ij} ||f_i - (Rm_j + t)|| \quad (4)$$

where f_i and m_j stand for points in C_f and C_m respectively, being w_{ij} 1 if f_i was matched with m_j and 0 otherwise, and $||\cdot||$ a distance metric (*e.g.* Euclidean distance). The process of assignment and alignment is repeated until a maximum number of 50 iterations is reached, or the update in translation, t is less than a threshold of 0.01mm for 3 successive iterations.

To localise in a room, we initialise the ICP alignment at each of the sampled grid of poses and take the result as the resulting alignment that has the lowest RSME error between point clouds. A typical concern when dealing with this iterative process is its execution time. To handle this, we performed a grid-average sampling method that merges points in the same grid cell, thus preserving the shape of the point clouds but reducing execution time. The grid cell side was heuristically fixed to 5cm.

2. Additional Experiments

In this section, we provide some additional experimental evaluation to complement the results in the main paper.

2.1. Time Complexity

In Table 1 we list the inference times. We omit the rendering time of the sampled grid in the retrieval stage as this time cost is shared between the methods and can be computed offline. In addition, the rendering during VDR could be optimised (currently 2.7s mean). We see that LaLaLoc offers a significant speed advantage over ICP. Interestingly, VDR appears to reduce the time required for LPO since it provides a better initialisation for LPO, reducing the mean number of iterations to convergence, 41.7 down to 31.4.

2.2. Layout Similarity Low-resolution Sampling

Here, we re-perform our evaluation of differing layout representations and similarity metrics for localisation on a lower resolution sampling grid. The results are listed in Table 2. As can be seen, the lower resolution sampling only

Similarity Metric	Recall @1	Pose Error Median (cm)	Correct Room
Pose	*100%*	*40.1*	*99.7%*
Edges	55.6%	48.7	59.2%
Depth	55.2%	48.3	59.5%
Rel. Depth	54.1%	48.5	61.6%
Chamfer	**71.8%**	**42.7**	**74.4%**

Table 2. Evaluation of layout similarity metrics on the Structured3D validation split, now sampled with a lower-resolution 1m $\times$ 1m grid. *Pose* refers to picking the nearest location in the sampled grid to the query.

serves to exaggerate the performance improvement seen by using the 3D Chamfer distance similarity metric, further emphasising the need for careful selection.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Sungyeon Kim, Minkyo Seo, Ivan Laptev, Minsu Cho, and Suha Kwak. Deep metric learning beyond binary supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2288–2297, 2019.

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[4] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018.