Supplementary Material for Learning a Depth Covariance Function

Eric Dexheimer and Andrew J. Davison
Dyson Robotics Lab, Imperial College London
{e.dexheimer21, a.davison}@imperial.ac.uk

1. Network Architecture and Training Details

For the convolutional neural network (CNN) component of our system, we use a UNet [5]. The 256×192 RGB input is first converted into a 16-dimensional feature space, which is the input to the UNet. We use 5 downsampling and upsampling layers. Each downsampling layer consists of max pooling and two residual convolution layers [2]. Each upsampling layer bilinearly interpolates the feature map to a higher resolution, performs a convolution, and then the two residual convolutions. We use LeakyReLU as the activation function for all convolutions, and we also use GroupNorm [6] with 16 groups.

The final four upsampling layers give outputs of the covariance parameters at different resolutions. When training, we compute the mean loss with respect to a resized depth image for each of these levels, and then scale the loss so that higher resolutions are given higher weight according to the number of pixels. For example, the highest resolution will have four times more weight than the second highest.

For the Nyström approximation during training, we fix the rank to be 128. The 128 inducing locations are sampled randomly since this is relatively efficient and was found to be more stable than active sampling. Exploring the rank parameter and how it may trade-off expressiveness and compactness is an interesting avenue for future work.

As mentioned previously, to handle scale, we solve for the optimal mean log-depth that minimizes the data term of the negative log marginal likelihood. To minimize the variational free energy, we use the Adam [3] optimizer with an initial learning rate of $3\text{e}{-4}$. During training, we used a batch size of 4 and performed data augmentation with random rotations, resized crops, flips, and color jitter.

2. Active Sampling Implementation

For active sampling of pixel locations to condition on or select as inducing points, we use the greedy variance selection described in [1]. We calculate the conditional variance for all pixel locations with respect to the current samples, and select the location with the highest variance. The conditional variance is the diagonal of the conditional covariance matrix described previously:

$$\Sigma_* = K_{\text{ff}} - K_{\text{fn}}(K_{\text{nn}} + \sigma_n^2 I)^{-1} K_{\text{nf}}. \tag{1}$$

Since this form requires maintaining the inverse $(K_{\text{nn}} + \sigma_n^2 I)^{-1}$ which is dynamically changing, we perform $\mathcal{O}(n)$ updates to the Cholesky factorization as in [4]. The decomposition is written as

$$(K_{\text{nn}} + \sigma_n^2 I)^{-1} = (LL^T)^{-1} = L^{-T} L^{-1} \tag{2}$$

Furthermore, we may also avoid recomputing the entire variance $\text{diag}[\Sigma_*]$ from scratch for each newly added point. We may write the conditional covariance as

$$\Sigma_* = K_{\text{ff}} - (L^{-1} K_{\text{nf}})^T (L^{-1} K_{\text{nf}}). \tag{3}$$

Since we only require $\text{diag}[\Sigma_*]$, we only need to add the new row of $L^{-1} K_{\text{nf}}$ for each new input point, and take the squared norm of each column when computing the variance. We do not actually invert L, but instead use efficient triangular solves via forward substitution. Thus, we only need to update L and $L^{-1} K_{\text{nf}}$ each with a new row. This avoids $\mathcal{O}(n^3)$ inversions and $\mathcal{O}(n^2)$ triangular solves for each step of the greedy selection.

3. Visual Odometry Photometric Factor

As mentioned previously, given log-depths reference frame i, we form the photometric constraint with respect to frame j:

$$E = \sum_{i,j \in E} \sum_{n} ||I_i(\mathbf{x}_n) - I_j(w(\mathbf{x}_n, \mathbf{T}_i, \mathbf{T}_j, \mathbf{y}_i, m_i)||^2_{\sigma_r^2 I} \tag{4}$$

where I is the image intensity, $\mathbf{x}$ are pixel coordinates, w is the warping function that queries a depth map and projects the corresponding point into the neighboring image, $\mathbf{T}$ are camera poses, $\mathbf{y}$ are the latent log-depth observations, and m is the mean log depth for a given frame.

First, the dense log-depth map is formed via the GP conditional mean:

$$\mathbf{f}_* = m_i + K_{\text{fn}}(K_{\text{nn}} + \sigma_n^2 I)^{-1}(\mathbf{y}_i - m_i). \tag{5}$$

The 3D points $\mathbf{P}_i$ in the reference frame can be calculated via backprojection of the vectorized pixel coordinates $\mathbf{x}_i$ via the known camera intrinsics:

$$\mathbf{P}_i = \pi^{-1}(\mathbf{x}_i, e^{\mathbf{f}_*}). \tag{6}$$

The points may then be transformed into frame j and projected into the image to yield the correspondence

$$\mathbf{x}_j = \pi(\mathbf{T}_j^{-1}\mathbf{T}_i\mathbf{P}_i). \tag{7}$$

These steps describe the warping function w that is used to achieve correspondence.

When the exposure times may vary, we also include affine brightness parameters (a, b) for the photometric factor. For brevity, we write the correspondences from w as $\mathbf{x}_j$, so that the unwhitened residual becomes

$$\mathbf{r}_{i,j} = I_i(\mathbf{x}_i) + b_i - \left(\frac{e^{-a_i}}{e^{-a_j}} I_j(\mathbf{x}_j) + b_j\right). \tag{8}$$

The affine brightness terms are jointly optimized with the other unknowns. To further robustify the cost against occlusion and specular surfaces, we use the Huber robust cost function instead of the non-robust least-squares cost.

References

[1] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in Gaussian processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2005. 1

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. 1

[4] Ananth Ranganathan, Ming-Hsuan Yang, and Jeffrey Ho. Online sparse Gaussian process regression and its applications. *IEEE Transactions on Image Processing*, 2011. 1

[5] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2015. 1

[6] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 1