VaPiD: A Rapid Vanishing Point Detector via Learned Optimizers – Supplementary Material

Shichen Liu[1,2], Yichao Zhou[3], and Yajie Zhao[2]
[1]University of Southern California
[2]USC Institute for Creative Technologies
[3]University of California, Berkeley
{lshichen,zhao}@ict.usc.edu zyc@berkeley.edu

We extend the discussions about the precision in vanishing point detection in Sec. 1, provide the details of our network architectures in Sec. 2 and more implementation details in Sec. 3, discuss more ablation studies in Sec. 4, and show more results in Sec. 5.

1. Floating-Point Precision

Most common deep learning frameworks adopt single-precision floating-point format (*float32*) for the values of both the network parameters and the outputs. We evaluate the position of a predicted vanishing point using the angle error metric in the *Gaussian sphere* representation, which can be computed by:

$$\theta_{\text{err}} = \arccos(|\langle \mathbf{v}, \mathbf{v}^* \rangle|), \tag{1}$$

where $\mathbf{v}$ and $\mathbf{v}^*$ are the Gaussian sphere representation of the prediction and the ground truth vanishing points, respectively. As θ_{err} is close to 0, the dot product of the vanishing point coordinates will be close to 1. We denote the machine epsilon of *float32* as $\epsilon \approx 1.19 \times 10^{-7}$. The minimum angular error that *float32* is able to represent is thus $\arccos(1-\epsilon) \approx 0.028°$.

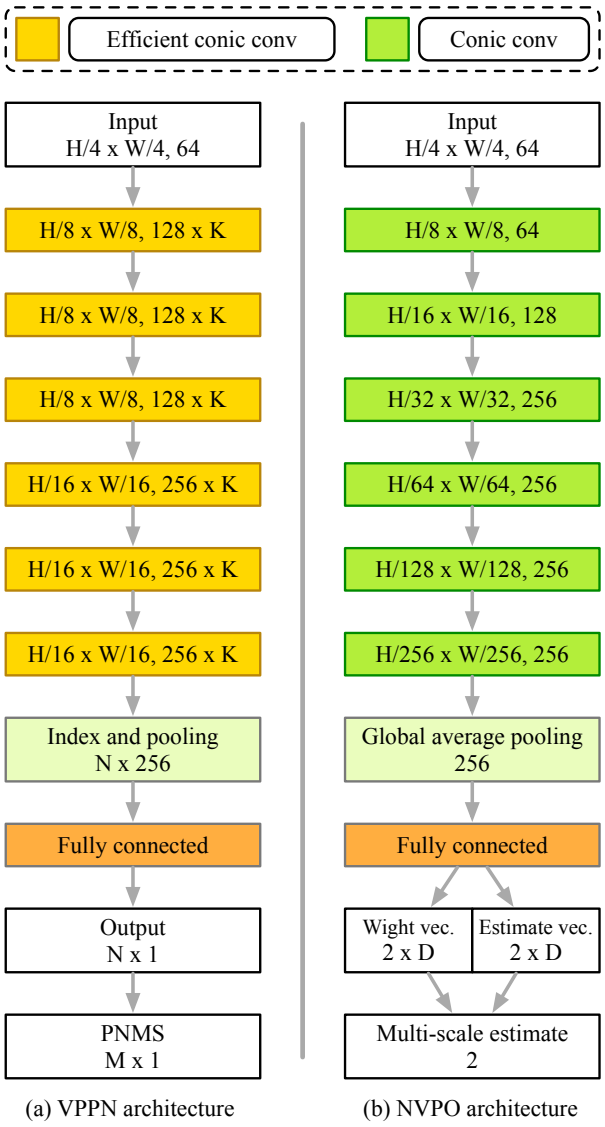
Different from previous method [2], in which each vanishing point candidate is enumerated in *float64* and supervised with a classification loss, we directly supervise our VaPiD with angular errors (in *float32*). Despite this natural error metric is more sensitive to numerical error, VaPiD still achieves a state-of-the-art median angle error of $0.089°$ in the photorealistic dataset [3]. The trivial difference between our results and the minimum angle error of *float32* indicates that the performance of our model is likely to be bounded by the numerical error of *float32* rather than the network prediction accuracy.

2. Network Architectures

VPPN. The architecture of the VPPN is given in Fig. 1(a). The VPPN contains an efficient conic convolution (ECC) block that has 6 stacked ECC layers. The feature maps are downsampled by 2 with each 3 layers. We rotate the kernels in the whole ECC block K times and apply all of them to the input feature maps. After obtaining K feature maps, we index and aggregate the feature maps according to the angles between the pixel coordinates and the vanishing point anchors to obtain the anchor features. Finally, we use a single fully connected layer with a $\mathrm{Sigmoid}$ activation function over each anchor feature to estimate the confidence scores of the anchors.

NVPO. We illustrate the structure of the NVPO in Fig. 1(b). Our NVPO contains 6 conic convolution layers. Different from NeurVPS [2] that flattens the feature maps and employs a fully connected layer, we downsample the feature maps with stride convolutions and employ a global average pooling layer. This is because the spatial sensitive operators such as flatten operator hinder the rotation invariance, which is required by our update scheme. The final output has $4 \times D$ channels. We apply a $\mathrm{softmax}$ function to the first $2 \times D$ channels to find the scale of the output, which gives a weight vector. The last $2 \times D$ channels are processed with a $\tanh$ function to regress the estimates for each scale, which gives an estimate vector. The weight vector and the estimate vector are then utilized to provide a *multi-scale estimate*, which is described in the following paragraph.

Multi-scale Estimation. We use the standard $\tanh$ function as our activation function. We observe that there is a significant precision gap between the initial guess ($\sim 4.28°$) and our target precision ($\sim 0.08°$). Instead of using a single $\tanh$ function, we extend it to multiple estimators in different precision scales to focus the networks on the desired scale of precision depending on the progress during



(a) VPPN architecture (b) NVPO architecture

Figure 1: The architectures of the VPPN and the NVPO.

	$AA^{.1^\circ}$	$AA^{.5^\circ}$	AA^{2°	Mean	Median
NVPO-MSE	0.184	0.684	0.914	0.172	0.117
NVPO-SW	0.265	0.734	0.927	0.150	0.092
NVPO (full)	**0.266**	**0.747**	**0.930**	**0.145**	**0.088**

Table 1: More ablation studies. "NVPO-MSE" denotes our method without multi-scale scheme. "NVPO-SW" shows the ablation on shared weights for each updating iteration.

the inference. For an estimation cap S, we define the different scales as an exponentially decreasing sequence $\{S, S\gamma, \ldots, S\gamma^{D-1}\}$, where $\gamma < 1$ is the scale factor and D is the number of scales. The prediction of the network is:

$$y = \sum_{i=0}^{D-1} S \cdot \gamma^i \cdot \text{softmax}(w_i) \cdot \tanh(x_i), \quad (2)$$

where $\{w_i\}_D$ and $\{x_i\}_D$ are the weight vector and the estimate vector predicted by the conic networks, respectively.

3. Implementation Details

Network Details. We implement our network in PyTorch. Our VPPN makes use of an anchor grid of size 1,024 that is generated using *Fibonacci lattice* algorithm:

$$\phi_i = \arccos(1 - n/N), \quad (3)$$

$$\theta_i = (1 + \sqrt{5})\pi n, \quad (4)$$

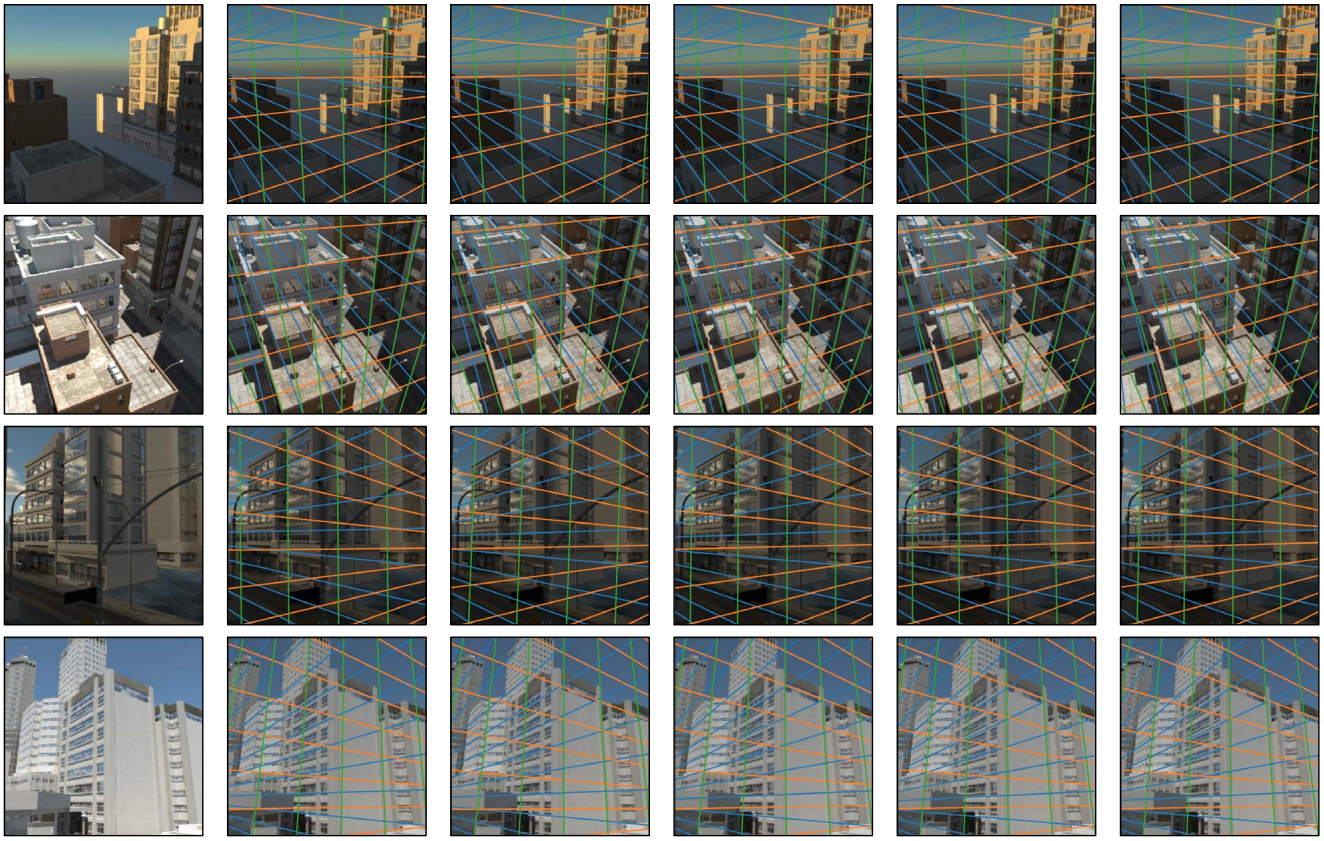
where (ϕ_i, θ_i) are the spherical coordinates of the i-th vanishing point anchor. We use PNMS with $\Gamma = 15^\circ$ and keep the top-$\mathcal{K}$ proposals if the dataset assumes a fixed number of ground truths $\mathcal{K}$, otherwise top-6. For the neural vanishing point optimizer, we set the multi-scale estimate cap $S = 20^\circ$, exponential decay rate $\gamma = 0.7$, and step size $D = 10$. During training, we compute the loss for 8 iterations.

Datasets. For the SU3 Wireframe dataset [3], we train the model for 40 epochs with a learning rate decay of 0.1 at the 30th epoch. We train our model for 160 epochs on the Natural Scene dataset [4] with a learning rate decay of 0.1 at the 120th epoch. We also adopt horizontal flip data augmentation as this dataset contains limited training examples. Because the dataset does not provide ground truth camera matrices, we simply set the focal length of all images to be 1. We train our model on the Holicity dataset [2] for 30 epochs with a learning rate decay of 0.1 at the 24th epoch. For NYU-VP dataset [1], we keep the original image resolution as 480×640. we train our model for 40 epochs with a learning rate decay of 0.1 at the 30th epoch.

4. More ablation studies

Multi-scale Estimate. We provide the ablation study on the multi-scale estimate in Tab. 1, where the NVPO-MSE is a variant of our NVPO that replace the multi-scale estimate with a naive Tanh function. We observe a significant improvement at high precision levels (e.g. 0.1°) by using the proposed multi-scale estimator.

Shared Weight Optimizers. We notice that the weight sharing scheme also improves the accuracy (NVPO-SW vs. NVPO). We hypothesis that the weight sharing scheme can serve as a regularizer that helps the network better converge to a fixed point. It also enables the flexibility of references with arbitrary steps.



(a) Input (b) VPPN prediction (c) NVPO $T = 1$ (d) NVPO $T = 2$ (e) NVPO $T = 4$ (f) NVPO $T = 8$

Figure 2: Vanishing points prediction using our networks with intermediate results. (a) Input images. (b) results from our VPPN. (c)-(f). different results by using different refinement steps ($T = 1, 2, 4, 8$) in NVPO taken the predictions in (b) as initials.

5. Additional Qualitative Results

We provide more examples with intermediate results visualized in Figure 2. We show the predictions from our VPPN in Figure 2(b). The results demonstrate that VPPN provides reliable initial guess of vanishing points for NVPO. The different refinement steps ($T = 1, 2, 4, 8$) used in NVPO are shown in Figure 2(c)-(f). We can observe that NVPO gradually refines the vanishing point estimation during the update. We also find that the visual difference between results of $T = 4$ and $T = 8$ is subtle, which indicates that our NVPO converges rapidly and saturated at the step of $T = 8$. Thus we pick $T = 8$ as the best time-accuracy trade-off as described in the main paper.

References

[1] Florian Kluger, Eric Brachmann, Hanno Ackermann, Carsten Rother, Michael Ying Yang, and Bodo Rosenhahn. Consac: Robust multi-model fitting by conditional sample consensus. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4634–4643, 2020. 2

[2] Yichao Zhou, Haozhi Qi, Jingwei Huang, and Yi Ma. Neurvps: Neural vanishing point scanning via conic convolution. In *Advances in Neural Information Processing Systems*, pages 866–875, 2019. 1, 2

[3] Yichao Zhou, Haozhi Qi, Yuexiang Zhai, Qi Sun, Zhili Chen, Li-Yi Wei, and Yi Ma. Learning to reconstruct 3d manhattan wireframes from a single image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7698–7707, 2019. 1, 2

[4] Zihan Zhou, Farshid Farhat, and James Z Wang. Detecting dominant vanishing points in natural scenes with application to composition-sensitive image retrieval. *IEEE Transactions on Multimedia*, 19(12):2651–2665, 2017. 2