VaPiD: A Rapid Vanishing Point Detector via Learned Optimizers

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

Abstract

*Being able to infer 3D structures from 2D images with geometric principles, vanishing points have been a well-recognized concept in 3D vision research. It has been widely used in autonomous driving, SLAM, and AR/VR for applications including road direction estimation, camera calibration, and camera pose estimation. Existing vanishing point detection methods often need to trade off between robustness, precision, and inference speed. In this paper, we introduce VaPiD, a novel neural network-based rapid **Va**nishing **P**oint **D**etector that achieves unprecedented efficiency with learned vanishing point optimizers. The core of our method contains two components: a vanishing point proposal network that gives a set of vanishing point proposals as coarse estimations; and a neural vanishing point optimizer that iteratively optimizes the positions of the vanishing point proposals to achieve high-precision levels. Extensive experiments on both synthetic and real-world datasets show that our method provides competitive, if not better, performance as compared to the previous state-of-the-art vanishing point detection approaches, while being significantly faster.*

1. Introduction

Vanishing points are defined as the intersection points of 3D parallel lines when projected onto a 2D image. By providing geometry-based cues to infer the 3D structures, they underpin a variety of applications, such as camera calibration [21, 7], facade detection [25], 3D reconstruction [15], 3D scene structure analysis [16, 39], 3D lifting of lines [30], SLAM [43], and autonomous driving [22].

Efforts have been made on vanishing point detection in the past decades. Traditionally, vanishing points are detected in two stages. In the first stage, a line detection algorithm, such as probabilistic hough transformation [18] or LSD [38], is used to extract a set of line segments. In the second stage, a line clustering algorithm [26] or a voting procedure [3] is used to estimate the final positions of vanishing points from detected line segments. The main weakness of this pipeline is that the extracted lines might be noisy, leading to spurious results after clustering or voting when there are too many outliers. To make algorithms more robust, priors of the underlying scenes can be used, such as Manhattan worlds [4] or Atlanta worlds [31], which are common in man-made environments. Nevertheless, additional assumptions complicate the problem setting, and the algorithms might not work well when these hard assumptions do not hold.

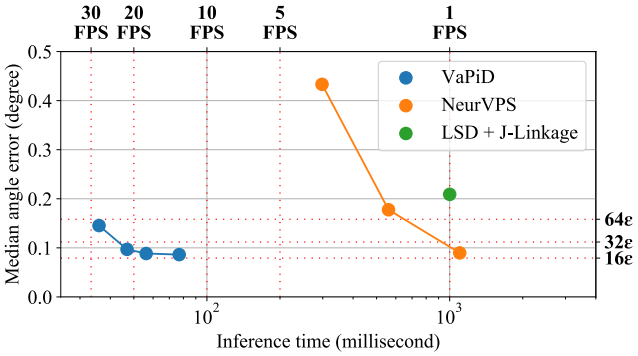


Figure 1: We propose a novel vanishing point detection network VaPiD, which runs in real-time with high accuracy. Speed-accuracy curves compare with state-of-the-art methods on the SU3 dataset [46]. Dotted horizontal lines labeled with $n\epsilon$ represent the nth smallest angle errors that numerically can be represented by 32-bit floating point numbers when computing the angle between two normalized direction vectors, i.e., $\arccos\langle \mathbf{d}_1, \mathbf{d}_2 \rangle$.

Recent CNN-based deep learning approaches [6, 5, 42, 41, 19, 45] have demonstrated the robustness of the data-driven approach. In particular, NeurVPS [45] provides a framework to detect vanishing points in an end-to-end fashion without relying on external heuristic line detectors. It proposes *conic convolution* to exploit the geometric prop-

erties of vanishing points by enforcing the feature extraction and aggregation along the structural lines of vanishing point candidates. This approach achieves satisfactory performance, but it is inefficient as it requires evaluating all possible vanishing points in an image (1FPS is reported in [45]). In contrast, most vanishing point applications must be run online in order to be useful in a practical setting.

To this end, we introduce VaPiD, a novel end-to-end rapid vanishing point detector that significantly boosts the model efficiency using learned optimizers. VaPiD consists of two components: (1) a vanishing point proposal network (VPPN) that takes an image and returns a set of vanishing point proposals. It harnesses a computation sharing scheme to efficiently process dense vanishing point anchors; (2) a neural vanishing point optimizer (NVPO) that takes each proposal as input and optimizes for its position with a neural network in an iterative fashion. In each iteration, it refines the vanishing points by regressing the residuals and updating the estimates. Our approach can be considered as *learning to optimize*. Compared to the previous coarse-to-fine method in [45], our optimizing scheme avoids enumerating all possible vanishing point candidate positions, which largely improves the inference speed.

We comprehensively evaluate our method on four public datasets including one synthetic dataset and three real-world datasets. VaPiD significantly outperforms previous works in terms of the efficiency, while achieving competitive, if not better, accuracy compared with the baselines. Remarkably, on the synthetic dataset, the cosine of the median angle error (0.088°) is close to the *machine epsilon* of 32-bit floating-point numbers[1], which indicates that VaPiD pushes the detection accuracy to the limit of numerical numbers. With fewer refinement iterations, VaPiD runs at 26 frames per second while maintaining a median angle error of 0.145° for 512×512 images with 3 vanishing points.

2. Related Work

Vanishing Point Detection. Early works represent vanishing points with unit vectors on a sphere (the Gaussian sphere), which reveals the link between the 2D vanishing point and the 3D line direction [3, 29]. Modern line-based vanishing point detection approaches first detect the line segments, which are then used to cluster vanishing points [40, 26, 27, 20]. Among them, LSD [38] with J-linkage [35, 10] is probably one of the most widely used algorithms. These methods work well on images with strong line signals, but are not robust to noises and outliers [32]. Therefore, structure constraints such as orthogonality properties are often used to increase the robustness. For example, the "Manhattan world" assumes three mutually orthogonal vanishing directions [8, 28, 24]. Similarly, under the "Atlanta world" assumption [31], vanishing points are detected in a common horizon line [2, 23, 40].

Recently, we have seen the success of CNN-based research on vanishing point detection. Chang *et al*. [6] detects vanishing points in the image frame by classifying over image patches. Zhai *et al*. [41] learns the prior of the horizon and its associated vanishing points in human-made environments. Kluger *et al*. [19] projects lines in the image to the Gaussian sphere and regresses directly on the spherical image. Zhou *et al*. [45] introduces the conic convolutions that can learn the vanishing point-related geometry features. Our approach also builds upon conic convolutions. We propose a computation sharing scheme that enables conic convolutions to process large-scale vanishing points.

Learning to Optimize. Using neural network layers to mimic the steps of optimization algorithms has shown to be effective in many computer vision tasks. Gregor *et al*. [14] first explored training a neural network as the approximation of an optimizer for sparse coding. This idea has been further applied to image super-resolution [9], novel view synthesis [11], and optical flow [36]. We follow this line of works and train a neural network that iteratively estimates the residuals and updates the vanishing points. In contrast to previous works, we target at the problem of vanishing point detection and our network optimizes the vanishing points in the semi-spherical space.

3. Method

3.1. Background

Geometry Representation of Vanishing Points. We adopt the *Gaussian sphere representation* of vanishing points [3]. The position of a vanishing point $\mathbf{v} = [v_x, v_y]^T \in \mathbb{R}^2$ in an image encodes a set of parallel 3D lines with direction $\mathbf{d} = [v_x - c_x, v_y - c_y, f]^T \in \mathbb{R}^3$, where $[c_x, c_y]^T \in \mathbb{R}^2$ is the optical center and f is the focal length of the camera. Representing vanishing points with $\mathbf{d}$ instead of $\mathbf{v}$ avoids the degenerate cases where the projected lines are parallel in 2D. In addition, we are now able to use the angle between two 3D unit vectors as the distance between two vanishing points. In this paper, we use both 3D $\mathbf{d} \in \mathbb{R}^3$ and its 2D counterpart of $\mathbf{v} \in \mathbb{R}^2$ to represent a vanishing point.

Conic Convolutions. The conic convolution [45] is designed to extract point related features. A conic convolution takes a feature map and a convolution center (the coordinates of the vanishing point candidates) as input, and outputs another feature map. Mathametically, a 3×3 conic

[1]The cosine error is the dot product of the direction represented by predicted vanishing points and ground truth vanishing points. It limits how accurate you can represent a vanishing point with floating-point numbers.

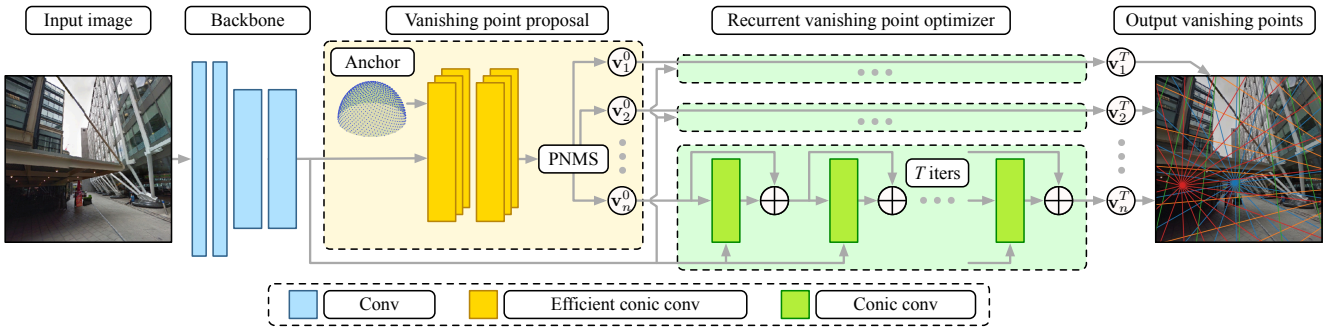


Figure 2: The architecture of our proposed VaPiD. It incorporates three major components: (1) a backbone network for feature extraction from the input image; (2) a vanishing point proposal network (VPPN) to generate reliable vanishing point proposals with efficient conic convolutions; (3) a weight sharing neural vanishing point optimizer (NVPO) to refine each vanishing point proposal to achieve high accuracy. Note that our network is trained in an end-to-end fashion.

convolution operator "$*$" is defined as:

$$(\mathcal{F} * \mathbf{w})(\mathbf{p} \,|\, \mathbf{v}) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} \mathbf{w}(i,j) \cdot \mathcal{F}\left(\mathbf{p} + \mathcal{R}_{\mathbf{v}-\mathbf{p}} \cdot \begin{bmatrix} i \\ j \end{bmatrix}\right), \quad (1)$$

where $\mathcal{F}$ is the input feature map, $\mathbf{w}$ is a 3×3 trainable convolution filter, $\mathbf{p} \in \mathbb{R}^2$ is the coordinates of the output pixel, $\mathbf{v}$ is the convolution center that is set to be the candidate positions of vanishing points, and $\mathcal{R}_{\mathbf{v}-\mathbf{p}}$ is a 2D rotation matrix that rotates the x-axis to the direction of $\mathbf{v}-\mathbf{p}$. In other words, conic convolution is a structured convolution operator that always rotates the filters towards the vanishing point $\mathbf{v}$ regardless of the output pixel coordinates $\mathbf{p}$. Intuitively, conic convolution can be seen as a way to check whether there are enough lines shooting from a vanishing point.

3.2. Overview

Fig. 2 illustrates our overall workflow. VaPiD takes an image as input and predicts the associated vanishing points. Specifically, a backbone network first extracts the feature map from the input image. Our vanishing point proposal network (VPPN) then generates a set of coarse vanishing point proposals using the feature map. Finally, the neural vanishing point optimizer (NVPO) optimizes each proposal individually for a fixed number of iterations. We introduce the designs of VPPN and NVPO in Sec. 3.3 and Sec. 3.4, respectively. In the end, we describe the loss functions for training both modules in Sec. 3.5.

3.3. Vanishing Point Proposals

The goal of the vanishing point proposal network (VPPN) is to produce a set of vanishing point proposals efficiently. Let $\{\mathbf{v}_i\}_{i=1}^{N}$ be an anchor point grid of size N on a unit sphere. The vanishing point proposal network classifies each anchor point to determine whether there is a vanishing point around it. We employ a point-based non-maximum suppression (PNMS) on the score map of the anchor grid to generate the final candidates.

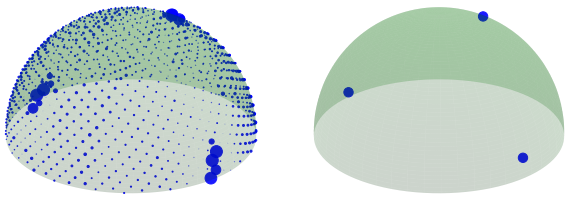
Efficient Conic Convolutions. Given a vanishing point anchor $\mathbf{v}_i$, the conic convolution centered at $\mathbf{v}_i$ relates the vanishing point to its line features. However, the conic convolution needs to process each vanishing point anchor separately, which is slow when N is large. To solve this problem, we propose the *efficient conic convolution* operator to quickly compute N feature maps by reusing some internal results with approximations.

Our key observation is that the rotation matrix $\mathcal{R}_{(\cdot)}$ in Equ. 1 is the only factor that varies regarding the same pixel $\mathbf{p}$. For a dense anchor point grid, the computation is redundant as multiple anchors may share similar rotation angles. Therefore, our method first approximates $\mathcal{R}_{(\mathbf{v}-\mathbf{p})}$ by K rotation matrices $\{\mathbf{R}_k\}_{k=0}^{K-1}$, where $\mathbf{R}_k$ is a 2D rotation matrix that rotates $\frac{2\pi k}{K}$ rad. We then pre-compute the feature map $\mathcal{G}_k$ by convolving the input feature map with the kernel rotated by $\mathbf{R}_k^{-1}$. After that, we can efficiently approximate the vanilla conic convolutions by retrieving the features from the pre-computed feature maps with the closest rotation angles. This process can be described with the following formulas:

$$(\mathcal{F} * \mathbf{w})(\mathbf{p} \,|\, \mathbf{v}) \quad (2)$$

$$= \sum_{i=-1}^{1} \sum_{j=-1}^{1} \mathbf{w}(i,j) \cdot \mathcal{F}\left(\mathbf{p} + \mathcal{R}_{\mathbf{v}-\mathbf{p}} \cdot \begin{bmatrix} i \\ j \end{bmatrix}\right) \quad (3)$$

$$\approx \sum_{i=-1}^{1} \sum_{j=-1}^{1} \mathbf{w}(i,j) \cdot \mathcal{F}\left(\mathbf{p} + \mathbf{R}_{k_{\mathbf{v},\mathbf{p}}} \cdot \begin{bmatrix} i \\ j \end{bmatrix}\right) \quad (4)$$



(a) Confidence score map (b) Vanishing point proposals w/ PNMS

Figure 3: Illustration of point-based non-maximum suppression (PNMS). (a) The confidence score map of a dense vanishing point anchor grid predicted by our efficient conic convolution networks. Higher scores are visualized as solid spheres with larger radius. (b) Top-3 vanishing point proposals after PNMS.

$$\approx \sum_{i=-1}^{1}\sum_{j=-1}^{1} \mathbf{w}\left(\mathbf{R}_{k_{\mathbf{v},\mathbf{p}}}^{-1}\cdot\begin{bmatrix}i\\ j\end{bmatrix}\right)\mathcal{F}\left(\mathbf{p}+\begin{bmatrix}i\\ j\end{bmatrix}\right) \quad (5)$$

$$\approx \left(\mathcal{F}\otimes \textsc{RotateKernel}(\mathbf{w},\mathbf{R}_{k_{\mathbf{v},\mathbf{p}}}^{-1})\right)(\mathbf{p}) \quad (6)$$

$$\doteq \mathcal{G}_{k_{\mathbf{v},\mathbf{p}}}(\mathbf{p}),$$

where $k_{\mathbf{v},\mathbf{p}} = \arg\max_k \mathrm{Tr}(\mathbf{R}_k \cdot \mathcal{R}_{\mathbf{v}-\mathbf{p}}^T)$ is the index of the closest rotation matrix, $\textsc{RotateKernel}(\mathbf{w}, \mathbf{R}_{k_{\mathbf{v},\mathbf{p}}}^{-1})$ rotates the convolutional kernel $\mathbf{w}$ with 2D rotation matrix $\mathbf{R}_{k_{\mathbf{v},\mathbf{p}}}^{-1}$, "$\otimes$" is the symbol of a regular convolution, and $\{\mathcal{G}_k\}_{k=1}^K$ is the set of pre-computed feature maps with 2D regular convolutions. Here, (2)-(3) is the definition of conic convolution, (3)-(4) is the step of rotation discretization, (4)-(5) is integration by substitution, and (5)-(6) is according to the definition of rotation and convolution. Once we obtain the feature maps for all the anchor points, we pass them into a fully connected layer with a Sigmoid activation to obtain the classification scores.

It takes N times of network forwards for the vanilla conic convolutions to compute the feature maps for a vanishing point set of size N, while the proposed efficient conic convolutions only require K times network forwards followed by a pooling operator that requires negligible computation costs, where K is the number of discretized rotation matrices. We find that setting $K = 64 \ll N$ can already give good approximations. In addition, efficient conic convolution only requires regular 2D convolution instead of deformable convolutions as used in [45], which in practice is much faster due to tremendous engineering efforts in modern deep learning frameworks.

Vanishing Point Non-maximum Suppression (PNMS). The dense score maps tend to be locally smooth. To remove duplicated proposals, we adopt a *point-based non-maximum suppression* (PNMS) approach, inspired by the widely adopted NMS techniques in object detection [12]. PNMS keeps the vanishing points with the greatest scores locally and suppresses its neighboring vanishing points within an angle threshold Γ. Figure 3 illustrates the effect of PNMS. After PNMS, we select the top-$\mathcal{K}$ ranked anchors as our proposals.

3.4. Learning to Optimize Vanishing Points

The goal of the neural vanishing point optimizer (NVPO) is to fine-tune the vanishing point positions starting from the initial proposal $\mathbf{d}^{(0)}$. Our NVPO emulates the process of iterative optimization and produces a sequence of estimates $\{\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(T)}\}$. In each iteration, it uses the image feature map and the current vanishing point position $\mathbf{d}^{(t)}$ to regress an update vector $\boldsymbol{\delta}^{(t)} = (\delta\theta^{(t)}, \delta\phi^{(t)})$ using the conic convolution network [45]. It then applies the update vector to the current vanishing point and obtains the next estimate (as shown in Equ. (7)). For each vanishing point proposal, we use T iterations. The network weights of the NVPO are shared across all the refinement iteration. As we only process a small number of vanishing points, we adopt the vanilla conic convolutions in our NVPO. We provide the network structure details in the supplementary materials.

We note that the update is applied in a local system defined by the position of vanishing points to avoid the problem of Gimbal lock. We first write out $\mathbf{d}^{(t)}$ using the spherical coordinate and construct $\Delta^{(t)} \in \mathbb{R}^3$ from the regressed vector $\boldsymbol{\delta}^{(t)}$:

$$\mathbf{d}^{(t)} = \begin{bmatrix}\cos\theta^{(t)}\sin\phi^{(t)} & \sin\theta^{(t)}\sin\phi^{(t)} & \cos\phi^{(t)}\end{bmatrix}^T,$$

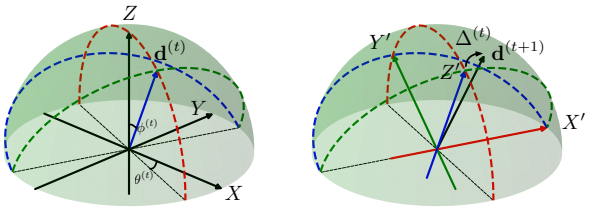
$$\Delta^{(t)} = \begin{bmatrix}\cos\delta\theta^{(t)}\sin\delta\phi^{(t)} & \sin\delta\theta^{(t)}\sin\delta\phi^{(t)} & \cos\delta\phi^{(t)}\end{bmatrix}^T.$$

We then define the local system (X', Y', Z'), where the Z'-axis corresponds $\mathbf{d}^{(t)}$ while keeping Z-axis lies in the $Y'Z'$ plane. The update vector is then applied to the current estimate in (X', Y', Z'). This process can be viewed as a rotation transformation:

$$\mathbf{d}^{(t+1)} = \mathbf{d}^{(t)} \oplus \boldsymbol{\delta}^{(t)} := \begin{bmatrix}\mathbf{e}^{(t)} & \mathbf{d}^{(t)}\times\mathbf{e}^{(t)} & \mathbf{d}^{(t)}\end{bmatrix}\cdot\Delta^{(t)}, \quad (7)$$

where $\mathbf{e}^{(t)} = [-\sin\phi^{(t)}, \cos\phi^{(t)}, 0]^T$. This process is illustrated in Fig. 4. An important property of such update scheme is *rotational equivariant*. If one rotates the vanishing points with respect to the optical center, the refined vanishing points will rotate in the same manner. This property is guaranteed by our method, as the conic convolutions centered at the vanishing points are by nature rotation invariant.

Although the NVPO still uses conic convolution to compute the features, our approach is more efficient compared to NeurVPS [45]. Specifically, NeurVPS samples candidate positions near the current estimates and uses conic convolutional networks to determine if each candidate is near a real vanishing point. Even with a coarse-to-fine strategy, it still needs to forward the conic convolutional networks 144 times per vanishing point in order to reach high precision levels, which largely limits the model efficiency. We instead adopt a "learning to optimize" methodology



(a) Local system (b) Update operator

Figure 4: Illustration of our update operator. (a) Given the camera system (X, Y, Z) and the current vanishing point position $\mathbf{d}^{(t)}$, we define the local system (X', Y', Z'). (b) We obtain the refined vanishing point $\mathbf{d}^{(t+1)}$ by applying the update vector $\Delta^{(t)}$ in the local system.

and directly regress the residuals of the vanishing points with our conic convolutional networks. Such a design greatly accelerates the overall process. For instance, we can now achieve better performance than NeurVPS with only a few network forwards. Alternatively, our approach can be viewed as solving for equilibrium points: our update formulation $\mathbf{d}^{(t+1)} = \mathbf{d}^{(t)} \oplus \boldsymbol{\delta}^{(t)} = f(\mathbf{d}^{(t)})$ can be viewed as a fixed point iteration method, where the function f is learned to fit our objective.

3.5. Loss Functions

For training VPPN, we assign a binary class label for each vanishing point anchor, where only the anchors with the closest angle to a ground truth are assigned with positive labels. This gives $\mathcal{L}_{\text{cls}} = \sum_{i=1}^{N} \text{BCE}(l_i, l_i^*)$, where l_i is the classification score for the i-th anchor and l_i^* is the assigned label. For training NVPO, we sample M anchors around the ground truths as the initial states and supervise NVPO with an angular loss between the estimates and the ground truths for each step $\mathcal{L}_{\text{ref}} = \sum_{i=1}^{M} \sum_{t=1}^{T} \arccos(|\langle \mathbf{v}_i^t, \mathbf{v}_i^* \rangle|)$. We jointly train both modules with the final loss $\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda \mathcal{L}_{\text{ref}}$, where λ is a trade-off hyperparameter.

4. Results

4.1. Datasets

We conduct empirical studies on the following datasets:

SU3 Wireframe [46]. SU3 Wireframe is a photo-realistic synthetic urban scene dataset generated with a procedural building generator. It contains 22,500 training images and 500 validation images. The dataset assumes "Manhattan world" scenes, where each image has exactly *three* mutual perpendicular vanishing points. The ground truths are calculated from the CAD models, which are accurate enough for a systematic investigation of vanishing point detection.

Natural Scene [47]. Collected from AVA and Flickr, this dataset contains images of natural scenes where the authors pick only *one* dominant vanishing point as the ground truth. We adopt the data split from [45] that divides the images into 2,000 training samples and 275 test samples.

HoliCity [44]. HoliCity is a city-scale real-world dataset with rich structural annotations. The ground truths are accurately aligned with the CAD model of downtown London. There are various numbers of vanishing points for each scene. We adopt the standard split that contains 45,032 training samples and 2,504 validation samples.

NYU-VP [20]. This dataset is manually labeled based on the NYU Depth V2 dataset [33]. It contains 1,449 indoor images. While most images show *three* ground truth vanishing points, it ranges from *one* to *eight*. We follow [20] and split the dataset into 1,000 training samples, 224 validation samples, and 225 testing samples.

4.2. Experiment Setups

Evaluation Metrics. We evaluate our method using mean and median angle errors. To better inspect our method under various precision levels, we also make use of angle accuracy (AA) metrics [45], where AA^{α} is defined as the area under the angle accuracy curve between $[0, \alpha]$ divided by α. For NYU-VP dataset, we follow [20] and adopt the AUC metric.

Baselines. We compare our method against robust fitting methods J-Linkage [10], T-Linkage [26], Sequential RANSAC [37], Multi-X [1], MCT [27], and learning-based method CONSAC [20], based on line segments extracted with LSD [38]. For vanishing point detection methods, we compare our method against traditional methods VPDet [47] and Simon *et al.* [34]. For learning-based vanishing point detection methods, we compare our method against Zhai *et al.* [41], Kluger *et al.* [19], direct CNN regression and classification [45], and previous state-of-the-art method NeurVPS [45].

4.3. Implementation Details

We implement our network in PyTorch. We resize the input images to 512×512. For generating the vanishing point proposals, we first uniformly sample an anchor grid of size $N = 1,024$ using *Fibonacci lattice* [13]. We then set the PNMS threshold $\Gamma = 15°$ and keep the top-$\mathcal{K}$ proposals if the dataset assumes a fixed number of vanishing points $\mathcal{K}$, otherwise $\mathcal{K} = 6$. For vanishing point refinement, we set the estimation cap $S = 20°$, and compute losses for $T = 8$ refinement steps. We use Adam optimizer [17] with a learning rate of 3×10^{-4} to train the network. We set the trade-off parameter $\lambda = 1$.

4.4. Results on Synthetic Datasets

We show our results on SU3 Wireframe [46] in Tab. 1, and plot the AA curves in Fig. 6. With a similar inference speed, VaPiD achieves an order of magnitude improvement over the naive CNN classification and regression baselines. VaPiD also significantly outperforms the traditional line-based J-Linkage clustering method [10]. Note that the synthetic dataset contains many sharp edges and long lines, which by nature favors the line-based detectors. The improvement of the accuracy validates the efficacy of geometry-inspired network designs such as conic convolutions and efficient conic convolutions. We also observe that our method runs 17 times faster than our strong baseline NeurVPS [45], while achieving better accuracies on all metrics. This indicates that our learning to optimize scheme can greatly improve the model efficiency upon the coarse-to-fine enumerating strategy used in NeurVPS. Remarkably, VaPiD is trained with angular metrics, but

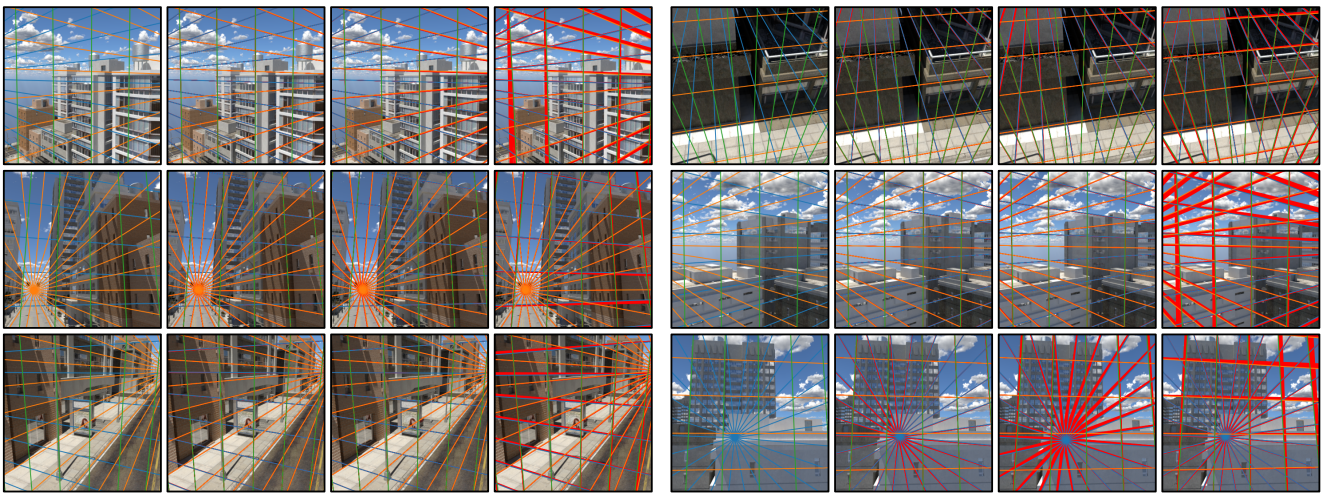


Figure 5: Qualitative comparison with baseline methods on SU3 Wireframe dataset [46]. Line group in the same color indicates the same vanishing point. We highlight all prediction errors in red color. Better view in color.

achieves a median angle error of 0.089° with float32. This is because $1-\cos(0.089°) \approx 1.2\times10^{-6}$, which is already close to the machine epsilon of 32 bit floating point numbers $\epsilon \approx 1.2\times10^{-7}$. This fact indicates that VaPiD is able to push the detection precision close to the numerical precisions. This is also reflected in the stepped curves at high precision levels (i.e. below 0.1°) in Fig. 6.

Qualitative Results. We provide the visual comparison of detected vanishing points with NeurVPS [45] and J-Linkage [10] in Figure 5. The detection errors are shown in red color. We observe that both the learning-based methods outperform the traditional method J-Linkage in prediction accuracy. Although the synthetic scenes contain sharp edges and long lines, the performance of J-Linkage is affected by clouds (second-row right panel), shadows (second-row left panel), and occluded lines (third-row right panel). Compared to NeurVPS, our method is more robust to occlusion when one of the vanishing points is occluded (third-row right panel). We believe this is benefited from our design of using a dense vanishing point anchor grid as initial.

4.5. Results on Real-World Datasets

Comparisons on Natural Scene. We show the comparisons on Natural Scene [47] in Tab. 2 and Fig. 7. Our method significantly outperforms the naive CNN classification and regression baselines as well as the contour-based clustering algorithm VPDet [47] in all metrics. It also outperforms the strong baseline NeurVPS [45] in most of the metrics. We note that the Natural Scene [47] is captured by cameras with different focal lengths. Such data favors the enumeration-based methods over the optimization-based methods, especially at a tighter angle threshold (i.e. below 1°). Nonetheless, we highlight that for images with one dominant vanishing point, VaPiD can run at real-time (43FPS) while maintaining competitive performance.

	$AA^{.2°}$	$AA^{.5°}$	$AA^{1°}$	Mean	Median
CNN-reg	2.03	6.48	15.02	2.077°	1.481°
CNN-cls	2.17	9.10	23.71	1.766°	0.984°
J-Linkage [10]	27.89	48.07	62.34	3.888°	0.209°
NeurVPS [45]	47.59	74.26	86.35	0.147°	0.090°
VaPiD	**48.33**	**74.79**	**86.66**	**0.145°**	**0.088°**

Table 1: Comparisons of mean, median angle errors and the angular accuracies of 0.2°, 0.5°, 1° with baseline methods on SU3 dataset [46].

	$AA^{1°}$	$AA^{2°}$	$AA^{10°}$	Mean	Median
CNN-reg	2.4	9.9	58.8	5.09°	3.20°
CNN-cls	4.4	14.5	62.4	5.80°	2.79°
VPDet [47]	18.5	33.0	60.0	12.6°	1.56°
NeurVPS [45]	**29.1**	**50.3**	85.5	1.83°	**0.87°**
VaPiD	24.6	49.5	**87.4**	**1.26°**	**0.87°**

Table 2: Comparisons of mean, median angle errors and the angular accuracies of 1°, 2°, 10° with baseline methods on Natural Scene dataset [47].

	$AA^{1°}$	$AA^{2°}$	$AA^{10°}$	Mean	Median
NeurVPS [45]	18.2	31.7	62.1	8.32°	1.78°
VaPiD	**22.1**	**39.6**	**75.4**	**3.00°**	**1.19°**

Table 3: Comparisons of mean, median angle errors and the angular accuracies of 1°, 2°, 10° with baseline methods on HoliCity dataset [44].

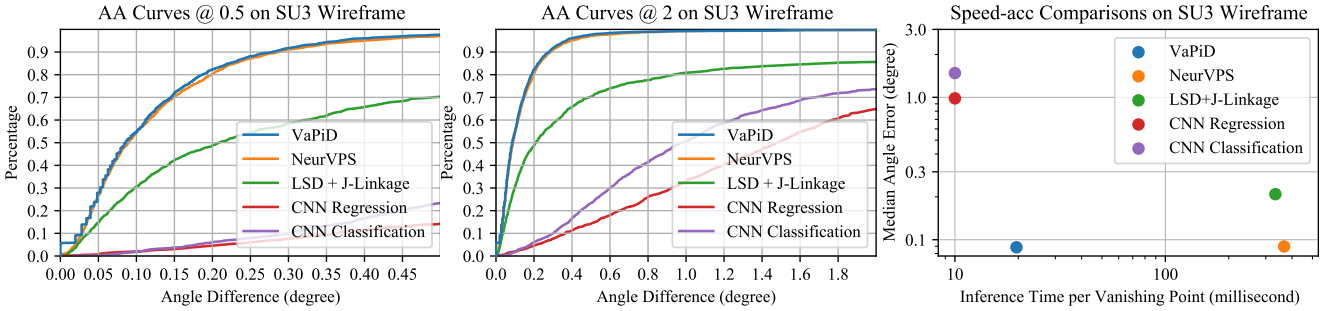


Figure 6: Angle accuracy curves and speed-accuracy comparisons for different methods on SU3 wireframe dataset [46].

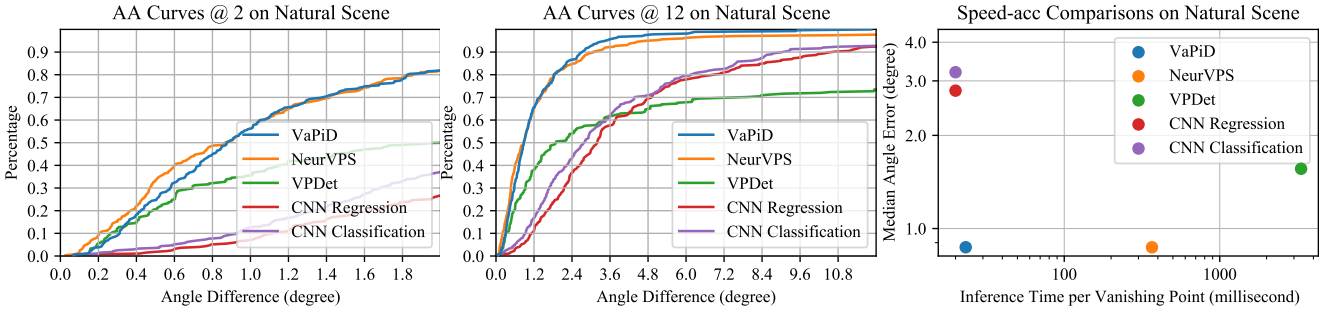


Figure 7: Angle accuracy curves and speed-accuracy comparisons for different methods on Natural Scene dataset [47]

Comparisons on HoliCity. We compare our method and NeurVPS [45] on the challenging real-world dataset, HoliCity [44]. The dataset mostly embraces the Atlanta world and contains images with a variable number of vanishing points. Our method outperforms NeurVPS on HoliCity. We think the gain originates from more vanishing points proposal being retrieved. This shows that our VPPN can adapt to complex scenes. Thanks to our efficient conic convolutions, we can process a denser anchor grid to produce fine-grained proposals.

Comparisons on NYU-VP. We compare against recent robust fitting methods CONSAC [20], T-Linkage [26], Sequential RANSAC [37], Multi-X [1], MCT [27], and vanishing point detection methods Simon *et al*. [34], Zhai *et al*. [41] and Kluger *et al*. [19] on the NYU-VP dataset [20], and show the results in Tab. 4. To fairly compare with the baselines, we follow [20] and use the Hungarian method to match the predictions and the ground truths. We find that in general the supervised methods perform better than traditional methods, and our method outperforms all baselines by a large margin. Compared to robust fitting methods, VaPiD does not rely on prior line detectors. Instead, thanks to our geometry-inspired structures, VaPiD can extract meaningful and robust line features from raw images intrinsically via end-to-end supervised learning. Compared to recent learning-based vanishing point detectors, VaPiD can make use of rich geometry cues, i.e. vanishing point-related line features, to accurately locate the vanishing points.

Qualitative Results. We visualize our detected vanishing points on HoliCity [44] in Figure 8, which shows that VaPiD is able to generalize well to different types of scenes and is robust to the perspective distortions. Thanks to the VPPN, our method can handle the input images with a variable number of vanishing points (2 for the first-row left panel, 3 for the first-row middle panel, and 4 for the second-row left panel) without relying on assumptions on the scene. In some cases, our predictions are even more reasonable than ground truths (orange in the second-row right panel).

	AUC^{10°	Supervised
Multi-X [1] †	41.3	no
MCT [27] †	47.0	no
Sequential RANSAC [37] †	53.6	no
T-Linkage [26] †	57.8	no
Kluger *et al*. [19]	61.7	yes
Simon *et al*. [34]	62.1	no
Zhai *et al*. [41]	63.0	yes
CONSAC [20] †	65.0	yes
VaPiD	**69.1**	yes

Table 4: Comparisons of AUC values at 10° with baseline methods on NYU-VP dataset [20]. Supervised methods are noted as "yes" in the last column. † Method requires additional line segment detector such as LSD [38].

4.6. Ablations

In this section, we show ablation studies to investigate the effect of each component in our model. All experiments are con-

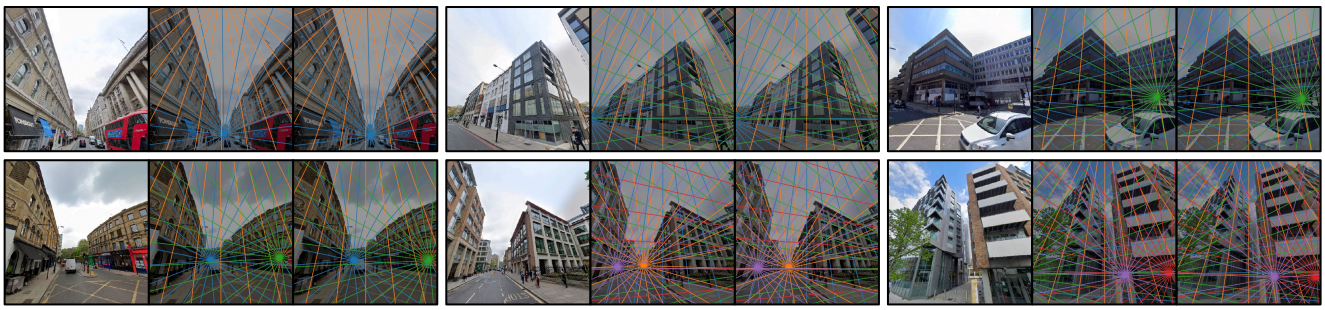


Figure 8: Visualization of our vanishing point detection results on various types of scenes. For each of the vanishing points, we visualize it using a group of 2D lines in the same color. Better view in color.

	Rec^{2°	Rec^{4°	Rec^{6°	Mean	Median
VPPN-ECC	33.20	72.13	89.53	0.554°	0.101°
VPPN	**56.20**	**95.67**	**99.00**	**0.146°**	**0.089°**

Table 5: Ablation study on the efficient conic convolutions. "VaPiD-ECC" denotes the baseline without using our proposed efficient conic convolutions.

	$\text{AA}^{\cdot 1^\circ}$	$\text{AA}^{\cdot 5^\circ}$	AA^{2°	Mean	Med.	FPS
NVPO $\times 4$	15.1	62.1	89.0	0.227°	0.145°	**26**
NVPO $\times 6$	24.3	72.7	92.4	0.157°	0.096°	22
NVPO $\times 8$	26.6	74.8	93.0	0.145°	0.088°	17
NVPO $\times 12$	27.5	75.2	93.2	0.141°	0.086°	13
NVPO $\times 16$	27.6	75.3	93.2	0.140°	0.086°	10
NVPO $\times 24$	**28.1**	**75.6**	**93.3**	**0.139°**	**0.085°**	7

Table 6: Ablation study on the number of refinement steps. ($\times T$) indicates T refinement steps during the inference.

ducted on SU3 Wireframe [46], as we can eliminate the labeling errors with the synthetic images.

The Effect of Efficient Conic Convolutions. The efficient conic convolutions are the core of our VPPN. In Tab. 5, we demonstrate the effectiveness of the efficient conic convolutions by investigating a variant of VPPN (VPPN-ECC) that replaces the efficient conic convolutions with vanilla conic convolutions but has a similar computation cost. As the goal of our VPPN is to pinpoint *all* of the vanishing point proposals, we adopt a recall metric, where Rec^α means the fraction of the ground truths that are successfully retrieved by one of the vanishing point proposals within the threshold of α. In Tab. 5, we observe that VPPN outperforms the VPPN-ECC baseline on all metrics. We note that the average closest neighbor angle of our anchor grid is 4.3°. We find that VPPN achieves 99% recall with a threshold of 6°, whereas the VPPN-ECC variant still struggles at 90%. This validates the efficiency of the computation sharing scheme in our efficient conic convolutions.

Convergence of the Learned Optimizer. In the training stage, we compute losses for a fixed refinement step of 8. To investigate the convergence of our NVPO, we apply different refinement steps during inference and show the results in Tab. 6. As the refinement step increases, it is clear that our NVPO gradually produces better vanishing point estimates, and can converge to a fixed point. We also make two key observations: (1) our method runs at nearly real-time (26FPS) with 4 refinement steps, yet is accurate enough (with a 0.15° median error) for many downstream applications; (2) even with 24 refinement steps, where the performance is saturated, our method still runs 7 times faster than thee previous state-of-the-art method [45], while being more accurate.

5. Conclusion

This paper presents a novel neural network-based vanishing points detection approach that achieves state-of-the-art performance while being significantly faster than previous works. Our method contains two designated modules: a novel vanishing points proposal network and a neural vanishing point optimizer. Our key insight is to use the computation sharing to accelerate massive convolution operations, and embrace a learning to optimize methodology that progressively learns the residual of the objectives. In future work, we will study how to combine VaPiD with downstream applications such as scene understanding, camera calibration, and camera pose estimation.

Acknowledgements

This research was sponsored by the Army Research Office and was accomplished under Cooperative Agreement Number W911NF-20-2-0053, and sponsored by the U.S. Army Research Laboratory (ARL) under contract number W911NF-14-D-0005, the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA and in part by the ONR YIP grant N00014-17-S-FO14. Statements and opinions expressed and content included do not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

[1] Daniel Barath and Jiri Matas. Multi-class model fitting by energy minimization and mode-seeking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 221–236, 2018. 5, 7

[2] Olga Barinova, Victor Lempitsky, Elena Tretiak, and Pushmeet Kohli. Geometric image parsing in man-made environments. In *Proceedings of European conference on computer vision*, pages 57–70. Springer, 2010. 2

[3] Stephen T Barnard. Interpreting perspective images. *Artificial intelligence*, 21(4):435–462, 1983. 1, 2

[4] Jean-Charles Bazin and Marc Pollefeys. 3-line RANSAC for orthogonal vanishing point detection. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4282–4287. IEEE, 2012. 1

[5] Ali Borji. Vanishing point detection with convolutional neural networks. *arXiv preprint arXiv:1609.00967*, 2016. 1

[6] Chin-Kai Chang, Jiaping Zhao, and Laurent Itti. Deepvp: Deep learning for vanishing point detection on 1 million street view images. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018. 1, 2

[7] Roberto Cipolla, Tom Drummond, and Duncan P Robertson. Camera calibration from vanishing points in image of architectural scenes. In *BMVC*, volume 99, pages 382–391, 1999. 1

[8] James M Coughlan and Alan L Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 941–947. IEEE, 1999. 2

[9] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015. 2

[10] Chen Feng, Fei Deng, and Vineet R Kamat. Semi-automatic 3d reconstruction of piecewise planar building models from single image. *CONVR (Sendai:)*, 2010. 2, 5, 6

[11] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2367–2376, 2019. 2

[12] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 4

[13] Álvaro González. Measurement of areas on a sphere using fibonacci and latitude–longitude lattices. *Mathematical Geosciences*, 42(1):49, 2010. 5

[14] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th international conference on international conference on machine learning*, pages 399–406, 2010. 2

[15] Erwan Guillou, Daniel Meneveaux, Eric Maisel, and Kadi Bouatouch. Using vanishing points for camera calibration and coarse 3d reconstruction from a single image. *The Visual Computer*, 16(7):396–410, 2000. 1

[16] Varsha Hedau, Derek Hoiem, and David Forsyth. Recovering the spatial layout of cluttered rooms. In *2009 IEEE 12th international conference on computer vision*, pages 1849–1856. IEEE, 2009. 1

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[18] Nahum Kiryati, Yuval Eldar, and Alfred M Bruckstein. A probabilistic Hough transform. *Pattern recognition*, 24(4):303–316, 1991. 1

[19] Florian Kluger, Hanno Ackermann, Michael Ying Yang, and Bodo Rosenhahn. Deep learning for vanishing point detection using an inverse gnomonic projection. In *German Conference on Pattern Recognition*, pages 17–28. Springer, 2017. 1, 2, 5, 7

[20] 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, 5, 7

[21] Jana Košecká and Wei Zhang. Video compass. In *European conference on computer vision*, pages 476–490. Springer, 2002. 1

[22] Seokju Lee, Junsik Kim, Jae Shin Yoon, Seunghak Shin, Oleksandr Bailo, Namil Kim, Tae-Hee Lee, Hyun Seok Hong, Seung-Hoon Han, and In So Kweon. Vpgnet: Vanishing point guided network for lane and road marking detection and recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 1947–1955, 2017. 1

[23] José Lezama, Rafael Grompone von Gioi, Gregory Randall, and Jean-Michel Morel. Finding vanishing points via point alignments in image primal and dual domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 509–515, 2014. 2

[24] Haoang Li, Ji Zhao, Jean-Charles Bazin, Wen Chen, Zhe Liu, and Yun-Hui Liu. Quasi-globally optimal and efficient vanishing point estimation in manhattan world. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1646–1654, 2019. 2

[25] Jingchen Liu and Yanxi Liu. Local regularity-driven city-scale facade detection from aerial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3778–3785, 2014. 1

[26] Luca Magri and Andrea Fusiello. T-Linkage: A continuous relaxation of J-Linkage for multi-model fitting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3954–3961, 2014. 1, 2, 5, 7

[27] Luca Magri and Andrea Fusiello. Fitting multiple heterogeneous models by multi-class cascaded t-linkage. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7460–7468, 2019. 2, 5, 7

[28] Faraz M Mirzaei and Stergios I Roumeliotis. Optimal estimation of vanishing points in a manhattan world. In *2011 International Conference on Computer Vision*, pages 2454–2461. IEEE, 2011. 2

[29] Long Quan and Roger Mohr. Determining perspective structures using hierarchical hough transform. *Pattern Recognition Letters*, 9(4):279–286, 1989. 2

[30] Srikumar Ramalingam and Matthew Brand. Lifting 3d manhattan lines from a single image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 497–504, 2013. 1

[31] Grant Schindler and Frank Dellaert. Atlanta world: An expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments. In *Proceedings of CVPR*, volume 1. IEEE, 2004. 1, 2

[32] Jefferey A Shufelt. Performance evaluation and analysis of vanishing point detection techniques. *IEEE transactions on pattern analysis and machine intelligence*, 21(3):282–288, 1999. 2

[33] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012. 5

[34] Gilles Simon, Antoine Fond, and Marie-Odile Berger. A-contrario horizon-first vanishing point detection using second-order grouping laws. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 318–333, 2018. 5, 7

[35] Jean-Philippe Tardif. Non-iterative approach for fast and accurate vanishing point detection. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1250–1257. IEEE, 2009. 2

[36] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. *arXiv preprint arXiv:2003.12039*, 2020. 2

[37] Etienne Vincent and Robert Laganiére. Detecting planar homographies in an image pair. In *ISPA 2001. Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis. In conjunction with 23rd International Conference on Information Technology Interfaces (IEEE Cat.*, pages 182–187. IEEE, 2001. 5, 7

[38] Rafael Grompone Von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. LSD: A fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence*, 32(4):722–732, 2008. 1, 2, 5, 7

[39] Rui Wang, David Geraghty, Kevin Matzen, Richard Szeliski, and Jan-Michael Frahm. Vplnet: Deep single view normal estimation with vanishing points and lines. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 689–698, 2020. 1

[40] Yiliang Xu, Sangmin Oh, and Anthony Hoogs. A minimum error vanishing point detection approach for uncalibrated monocular images of man-made environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1376–1383, 2013. 2

[41] Menghua Zhai, Scott Workman, and Nathan Jacobs. Detecting vanishing points using global image context in a non-manhattan world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5657–5665, 2016. 1, 2, 5, 7

[42] Xiaodan Zhang, Xinbo Gao, Wen Lu, Lihuo He, and Qi Liu. Dominant vanishing point detection in the wild with application in composition analysis. *Neurocomputing*, 311:260–269, 2018. 1

[43] Huizhong Zhou, Danping Zou, Ling Pei, Rendong Ying, Peilin Liu, and Wenxian Yu. Structslam: Visual slam with building structure lines. *IEEE Transactions on Vehicular Technology*, 64(4):1364–1375, 2015. 1

[44] Yichao Zhou, Jingwei Huang, Xili Dai, Linjie Luo, Zhili Chen, and Yi Ma. HoliCity: A city-scale data platform for learning holistic 3D structures, 2020. arXiv:2008.03286 [cs.CV]. 5, 6, 7

[45] 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, 4, 5, 6, 7, 8

[46] 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, 5, 6, 7, 8

[47] 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. 5, 6, 7