Geometric priors for deep vanishing point detection

Yancong Lin[1], Ruben Wiersma[2], Silvia L. Pintea[1], Klaus Hildebrandt[2], Elmar Eisemann[2], and Jan C. van Gemert[1]

[1]Computer Vision Lab, Delft University of Technology, the Netherlands
[2]Computer Graphics and Visualization, Delft University of Technology, the Netherlands

Abstract

Deep networks perform admirably on vanishing point detection. However, they require numerous annotated training examples. Moreover, existing approaches often rely on the Manhattan assumption to reduce the search space, thus limiting their applicability to orthogonal vanishing points only. To address this, we incorporate two representations for geometric priors suitable for vanishing point detection: (i) Hough transform – mapping image pixels to parameterized lines, and (ii) Gaussian sphere – mapping lines to great circles whose intersections denote vanishing points. Adding priors leads to data-efficiency as the network no longer needs to learn all the geometric knowledge from the data. Our two geometric priors are implemented as end-to-end trainable modules, and the mapping from pixels to the sphere via Hough Transform is parallelizable, allowing us to detect multiple non-orthogonal vanishing points simultaneously. We experimentally demonstrate the usefulness of our model when compared to state-of-the-art on both Manhattan and non-Manhattan world scenarios.

1. Introduction

Deep learning approaches for vanishing point (VP) detection [7, 8, 56, 57] provide accurate predictions, that can successfully be applied for a plethora of real-life tasks: camera calibration [9, 1, 21], scene understanding [18], visual SLAM [13, 31], or even autonomous driving [28]. However, these models solve the vanishing point detection by learning geometric knowledge from large annotated training sets. Large datasets entail significant costs for annotations and large computations, as CNNs go over the whole dataset for multiple rounds during training. Current works [32, 57] limit the solution space by making the orthogonality assumption which reduces computations, but restricts their applicability to the Manhattan world scenarios. Current deep learning approaches are neither data-efficient, nor compute efficient.

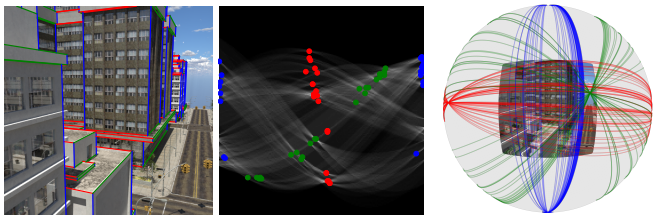


Image domain (i) Hough Transform (ii) Gaussian sphere

Figure 1. We add two geometric priors: (i) Hough Transform and (ii) Gaussian sphere mapping, for vanishing points detection. We transform learned image features to the Hough domain, where lines are mapped to individual bins. We further project the Hough bins to the Gaussian sphere, where lines become great circles and vanishing points are at the intersection of great circles. Each color represents a set of image lines that relate to a vanishing point. Adding geometric prior knowledge makes our model data-efficient, and our parallelizable mapping from pixels domain to the Gaussian sphere via Hough Transform, allows our model to detect a variable number of vanishing points.

Here, we propose a data-efficient and compute-efficient general-purpose method for vanishing point detection. We add two potent geometric representation priors, as exemplified in Fig. 1: (i) Hough transform and (ii) Gaussian sphere mapping. Using such geometric priors is data-efficient as less annotations are needed to extract such geometric knowledge from data. We add a trainable Hough Transform module which represents each line as an offset-angle pair in line polar coordinates. This allows us to identify individual lines in Hough space [15]. We subsequently map these lines in Hough space to the Gaussian sphere, where lines become great circles, and vanishing points are located at the intersection of great circles [4]. The benefit of using great circles is that lines are mapped from the unbounded image plane to a bounded unit sphere, allowing us to easier detect vanishing points outside the image view. Both the Hough Transform and the Gaussian sphere mapping are end-to-end trainable in a deep network and can be pre-computed for all possible lines and spherical points. Retaining the pixel-line-sphere mapping allows for a variable number of vanishing

points to be detected in parallel, and makes our method applicable for both Manhattan and non-Manhattan worlds.

This paper makes the following contributions: (1) we add two geometric priors for vanishing point detection by mapping CNN features to Hough Transform, and mapping Hough bins to the Gaussian sphere; (2) we improve data efficiency of learning-based vanishing point detection; (3) we propose a general method that can be applied to detect a varying numbers of vanishing points; (4) we experimentally demonstrate the usefulness of our approach on large synthetic and real-world datasets, Scene Urban 3D [58], ScanNet [12] and York Urban [14], where we demonstrate data efficiency and parameter reduction. (5) we further demonstrate the usefulness of our approach on the NYU Depth [39] dataset, where the number of vanishing points varies drastically from 1 to 8.

2. Related work

Geometry-based vanishing point detection. Vanishing points are found at the intersection of lines. Prior work relies on hypothesis testing by counting the number of inlier lines [51], voting schemes [19], or measuring the probability of a group of lines passing through the same point [49]. J-Linkage [51] is an example of the hypothesis selection solution, which has been used in [17, 50]. Other approaches see vanishing point detection as a grouping problem by applying line clustering [3, 37, 45], expectation-maximization [1, 14, 27, 46], or branch-and-bound [5, 6, 23]. Inspired by these works we also work on explicit line representations which we learn through a Hough transform module.

To use lines for vanishing point detection, these lines first have to be detected. Lines can be found by contour detection [59] or a dual point-to-line PClines mapping [29]. The common approach, however, is using the Hough transform [35, 43, 41] for line detection. Similarly, we add a Hough transform module into deep networks to detect candidate lines, which are subsequently mapped to the Gaussian sphere for vanishing point detection.

Transforming lines to the Gaussian-sphere has been widely used for vanishing points detection, as the the sphere is a bounded parameter space while the image plane is an open space [10, 36, 48]. A line in the image maps to a great circle on the Gaussian-sphere and the intersections of great circles on this unit sphere denote vanishing points, detected as local maxima [4], which can be recursively detected [43]. Similar to these works we exploit the benefit of the bounded parameter space and also rely on the Gaussian sphere representation to find vanishing points, however we differ by doing so in an end-to-end trainable network.

Learning-based vanishing point detection. With the recent popularity of deep learning, many works train convolutional neural networks on large scale datasets for vanishing point detection [7, 8, 55, 56]. We also reap the benefits of deep learnig, yet incorporate explicit geometric knowledge into the deep networks to improve data efficiency and reduce the dependency on large, expensive, manually annotated datasets..

Great recent work combines geometric knowledge with deep networks for vanishing point detection. The task can be solved by splitting the problem into multiple separate stages: line detection, inverse gnomonic projection, network training and post-processing [25], from which we differ as we incorporate all geometric knowledge jointly into a single end-to-end trainable network. Alternatively, the use of conic convolutions on densely sampled points on the hemisphere is effective at detecting vanishing points [57], from which we differ as we remove the need for multi-round dense sampling by mapping pixels to the complete hemisphere in parallel through our Hough Transform and Gaussian sphere modules. It is this complete mapping that allows us to overcome the Manhattan-world assumption and detect a variable number of vanishing points.

Manhattan versus non-Manhattan world. Incorporating the Manhattan world assumption has been proven useful for orthogonal vanishing point detection [2, 5, 38, 54]. However, there are certain scenarios where Manhattan assumption does not hold, such as non-orthogonal walls and wireframes in man-made structures. Vanishing point detection in non-Manhattan world can be done by robust multi-model fitting [26], or deep networks relying on domain-specific knowledge (e.g. horizon lines) [55], or branch-and-bound with a novel mine-and-stab strategy [30]. In our work, we do not add explicit orthogonality constraints to the optimization, but rather rely on the Hough transform and Gaussian sphere to map pixel-wise representations to the entire hemi-sphere in a parallel manner. This makes it possible to detect multiple vanishing points simultaneously by a simple clustering algorithm, making our method applicable in non-Manhattan scenarios as well.

3. Geometric priors for VP detection

General outline of our approach. Given an image, our model outputs a probability for points on the Gaussian hemisphere being a vanishing point. Our model adds two geometric priors: (i) Hough Transform, and (ii) Gaussian sphere mapping; built on top of deep image features. We use a single-stack hourglass network [40] to learn these image features, which are then mapped to a line parameterization via the Hough Transform. In the Hough space, we use convolutions to find the bins that represent lines [34] which we project to the Gaussian sphere via a sphere mapping [4], where spherical convolutions [44] are able to precisely localize vanishing points on the hemisphere. Both the Hough Transform and the Gaussian sphere mapping are parallelizable as fast tensor operations, which allows end-

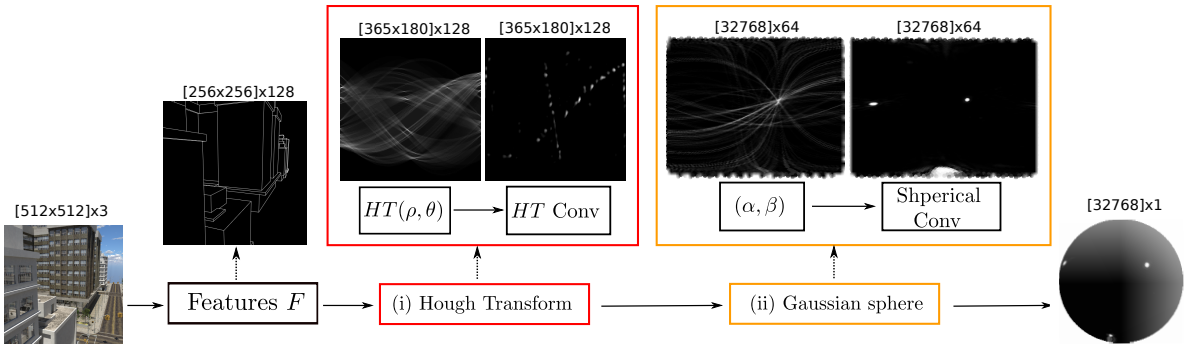


Figure 2. **Overview:** The model takes an image as input, and directly predicts vanishing points on the Gaussian hemisphere by relying on two geometric prions on top of feature extraction: (i) Hough transform, and (ii) Gaussian sphere mapping. We use a convolutional network to extract image features. The features are transformed to Hough space, where a single bin represents a line. Sequentially, we filter the Hough space and project Hough bins to the Gaussian hemisphere and apply spherical convolutions to find vanishing points. The digits above the images indicate the size of the learned features, where the last dimension is the number of channels. We sample 32,768 points on the hemisphere using the Fibonacci lattice [20], which results in features maps of size 32768.

to-end training within deep networks architectures. Fig. 2 depicts the overall structure of our model.

3.1. (i) Hough Transform prior

Given an image, we first use a single-stack hourglass network [40] to learn features, F. These features are mapped into Hough space, HT, via a trainable Hough Transform module [34]. This mapping parameterizes image lines in polar coordinates using a set of discrete offsets ρ and discrete angles θ, defining a 2D discrete histogram. In practice, a set of pixels $(x(i), y(i))$ along a line indexed by i, vote for a line parameterization to which they all belong:

$$HT(\rho,\theta) = \sum_i F(\rho\cos\theta - i\sin\theta, \rho\sin\theta + i\cos\theta) \quad (1)$$

The Hough Transform module takes an $[H\times W]$ feature map F as input and outputs an $[N_\rho\times N_\theta]$ Hough histogram HT, where N_ρ and N_θ are the number of sampled offsets and angles in Hough Transform. We set H=256, W=256, N_ρ=365, and N_θ=180. This results in $[N_\rho\times N_\theta]$ possible line parameterizations in total. Following [34], we find the local maxima in the Hough domain by performing a set of 1D convolutions over the offsets. This removes the noisy responses in the Hough space, as shown in Fig. 2.

3.2. (ii) Gaussian sphere prior

Instead of working with vanishing points in image coordinates, representing them as normalized 3D line directions δ, known as the Gaussian sphere representation, offers a bounded space and circumvents degenerate cases when the 3D line direction δ and the image plane are parallel.

(ii.1) Gaussian sphere mapping. The Gaussian sphere is a unit sphere centered at the camera origin, $\mathbf{O}$. A line starting at at the camera origin in direction δ maps to a point on the hemisphere. Starting from a bin in the Hough domain (ρ_{AB}, θ_{AB}), corresponding to a line direction in the image plane $\overrightarrow{AB}$, we want to map this to the Gaussian sphere.

Two image points $\mathbf{A}$ and $\mathbf{B}$ sampled from a line represented by its HT bin (ρ_{AB}, θ_{AB}), together with the camera center $\mathbf{O}$, form a plane ψ as depicted in Fig. 3(a). The plane ψ is described by its normal vector:

$$\overrightarrow{n} = (n_x, n_y, n_z) = \frac{\overrightarrow{OA}\times\overrightarrow{OB}}{\left\|\overrightarrow{OA}\times\overrightarrow{OB}\right\|}. \quad (2)$$

This normal vector $\overrightarrow{n}$ is the only information we need to map the image line direction $\overrightarrow{AB}$ to the Gaussian sphere.

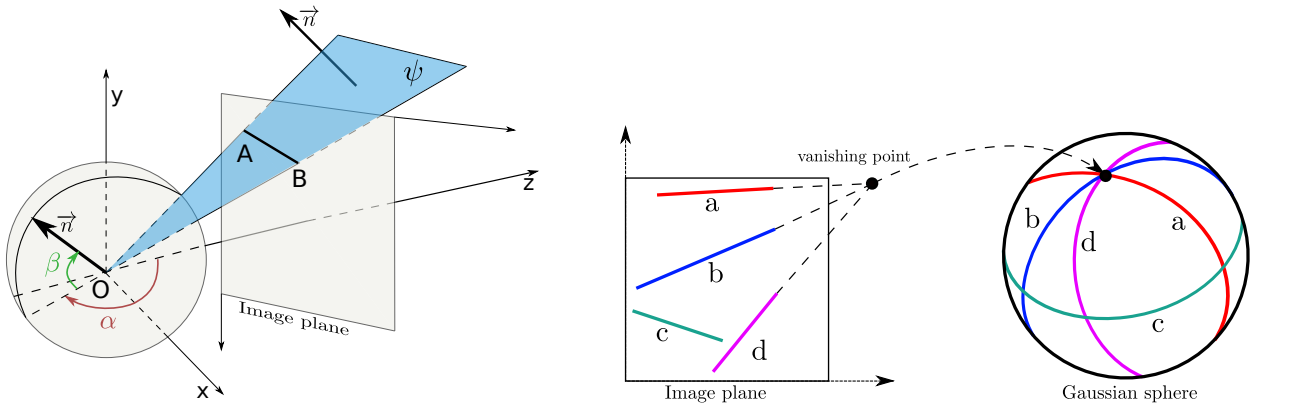
The spherical coordinates (α, β) describe a point on the Gaussian sphere, where α is the azimuth defined as the angle from the z-axis in the xz plane, and β is the elevation representing the angle measured from the xz plane towards the y-axis, as shown in Fig. 3(a). The intersection between the plane ψ and the Gaussian sphere is a great circle. This great circle represents the projection of the image line direction $\overrightarrow{AB}$ on the Gaussian sphere. A vanishing point on the sphere is located at the intersection of multiple great circles, as depicted in Fig. 3(b).

We compute the projection of the image line direction $\overrightarrow{AB}$, by estimating the elevation β as a function of the azimuth α and the normal vector $\overrightarrow{n}$ [43]:

$$\beta(\alpha, \overrightarrow{n}) = \tan^{-1}\frac{-n_x\sin\alpha - n_z\cos\alpha}{n_y}, \quad (3)$$

where we uniformly sample α in the range $[0, \pi)$.

Because the Gaussian sphere is symmetric we only need a hemisphere. We sample N points on the Gaussian hemisphere using a Fibonacci lattice [20] and then project lines, corresponding to bins in the Hough space, to these N sampled sphere points. For each line parameterization in Hough



(a) The Gaussian sphere (b) Vanishing points on the Gaussian sphere

Figure 3. **Gaussian sphere representations for vanishing points [4, 43]:** (a) The Gaussian sphere is a unit sphere located at the camera center, **O**. Points on the sphere are encoded by two angles: (α, β) the azimuth and the elevation, respectively. A line segment **AB** in the image plane together with the camera center **O** forms a plane ψ, highlighted in blue. To define the mapping from the image to the sphere, we only need to know the normal $\overrightarrow{n}$ to the plane ψ. Image lines are projected as great circles on the sphere. (b) The intersection of multiple great circles on the sphere represents a vanishing point.

space (ρ, θ), we first compute its normal vector $\overrightarrow{n}$. We, then, estimate its corresponding (α, β) spherical coordinates using Eq. (3). We subsequently assign each (α, β) pair to its nearest neighbor in the sampled points from the Fibonacci lattice, by computing their cosine distance. To parallelize this process, we precompute the projection of all Hough line parameterizations onto the sampled sphere locations. This mapping is stored in an $[N_\rho \times N_\theta \times M]$ tensor, where $[N_\rho \times N_\theta]$ is the number of line parameterizations in Hough space and M is the number of sampled azimuth angles α. We set N=32,768 and M=1,024.

(ii.2) Spherical convolutions on the hemisphere. After projecting the image features to the Gaussian hemisphere via Hough Transform and Gaussian mapping, we want to precisely localize vanishing points on the hemisphere. Due to the presence of outliers, we employ spherical convolutions to remove false positive predictions. We treat the points sampled from the hemisphere as a point cloud and use EdgeConv [53] to convolve over the hemisphere. EdgeConv operates on a k-nearest neighbor graph on the points. It learns to represent local neighborhoods by applying a non-linear function to the neighbors' features and then aggregating those features with a symmetric operator. The neighbors' features are localized by subtracting the features of the centroid. Like [53], we take the per-feature maximum over the neighbors to aggregate edge features.

As shown in Fig. 4, the spherical part of our model contains 5 EdgeConv modules [53]. Each EdgeConv module transforms neighboring features with a fully connected layer, a BatchNorm layer [22] and a LeakyReLU activation. We use N=32,768 nodes on the hemisphere and compute the 20 nearest neighbors for each node. We concatenate the features maps from previous layers and feed them into the last EdgeConv layer to produce the final prediction.

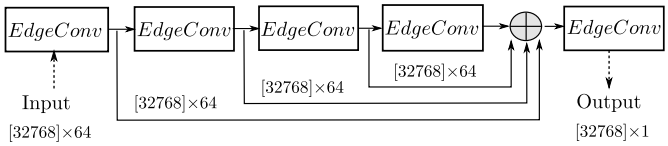


Figure 4. **Spherical convolutions on the hemisphere.** We use EdgeConv [53] for precise vanishing point localization on the Gaussian sphere. The concatenation of the previous feature maps is fed into the final layer to produce a prediction.

3.3. Model training and inference

We train the model using the binary cross-entropy loss. Because the number of positive samples is considerably lower than the number of negative samples, we compute two separate average losses over the positive and the negative samples, and then sum these two losses.

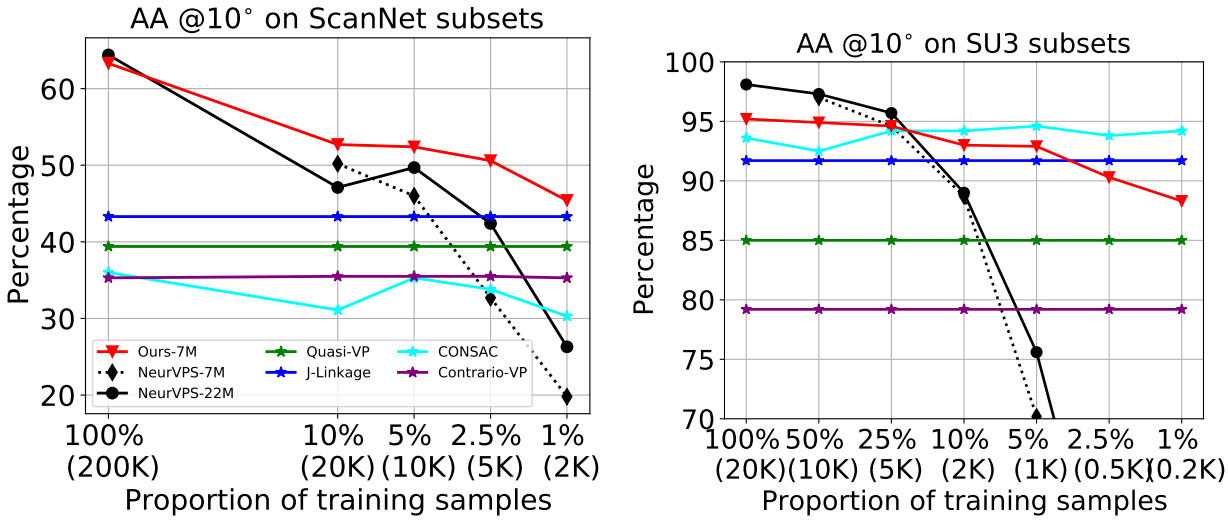
During inference, we use DBSCAN [16] to cluster all points on the Gaussian sphere based on the cosine distance. The *eps* parameter of DBSCAN [1]is set to be 0.005. The point with highest confidence in each cluster is the prediction. We rank all predictions by confidence.

4. Experiments

4.1. Experimental setup

Datasets. We compare with the state of the art on three datasets following the Manhattan world assumption: SU3

[1] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html



(a) Data efficiency on ScanNet

(b) Data efficiency on SU3

Figure 5. **Exp 1: Data efficiency.** We report AA @10° on various subsets of the ScanNet and SU3 datasets, and indicate the model size in the legend. On the ScanNet dataset, our model outperforms other methods on the 10%, 5% and 2.5% subsets. In comparison, the performance of NeurVPS is greatly affected by the reduction in the training data. Our model shows limited accuracy decrease when reducing the data from 20K to 1K on the SU3 dataset, while NeurVPS has a drastic drop in accuracy on small subsets, while performing well on the full dataset. This experiment validates the data efficiency of our model.

(SceneCity Urban 3D) [58], ScanNet [12], YUD [14], as well as the NYU Depth [39] dataset which does not follow the Manhattan world assumption. The SU3 dataset contains 23 000 synthetic images, which are split into 80%, 10% and 10% for training, validation and testing respectively. The ScanNet has more than 200K real-world image, among which 189 916 examples are used for training. In the NYU Depth dataset the number of vanishing points varies from 1 to 8 across images, making it more challenging. The NYU Depth dataset has 1 000 training image, approximately $\times 200$ and $\times 20$ smaller than the ScanNet dataset and the SU3 dataset, respectively, further increasing the difficulty of training CNN models. We additionally demonstrate the effect of geometric priors on the small-scale YUD dataset with only 102 images. We provide a detailed comparison in the supplementary material.

Evaluation. On the SU3, ScanNet and YUD datasets (Manhattan assumption), we evaluate the angle difference between the predicted and the ground-truth vanishing points on the Gaussian sphere, as in [57]. We then estimate the percentage of the predictions that have a smaller angle difference than a given threshold and compare the angle accuracy (AA) under different thresholds, as in [57]. On the NYU Depth dataset we follow [26] and first rank detected vanishing points by confidence, and then use the bipartite matching [11] to calculate the angular errors for the top N predictions, where N is the number of ground truth instances. After matching, we generate the recall curve and measure the area under the curve (AUC) up to a threshold, e.g., 10°.

Baselines. We compare our model with J-Linkage [17], Contrario-VP [47], Quasi-VP [32, 33], NeurVPS [57] and CONSAC [26] on the SU3, ScanNet and YUD dataset. On the NYU Depth dataset we compare only with J-Linkage and CONSAC, as the other models rely on the Manhattan assumption. J-Linkage, Contrario-VP and Quasi-VP are non-learning methods, and employ line segment detection [52]. NeurVPS and our model are end-to-end trainable, while CONSAC requires line segments as inputs.

Implementation details. We implement our model in Pytorch [42], and provide the code online [2]. All models are trained from scratch on Nvidia GTX1080Ti GPUs with the Adam optimizer [24]. The learning rate and weight decay are set to be 4×10^{-4} and 1×10^{-5}, respectively. To maximize the used GPU memory, we set the batch size to 4. On the SU3 and NYU Depth datasets, we train the model for a maximum of 36 epochs, with the learning rate decreased by 10 after 24 training epochs. On the ScanNet dataset, we train the model for 6 epochs and decay the learning rate by 10 after 4 epochs. On the YUD datset, we first pre-train on the SU3 dataset and then fine-tune with 25 YUD images for another 36 epochs with the learning rate 1×10^{-5}.

4.2. Exp 1: Data efficiency

We evaluate data efficiency by reducing the number of training samples to $\{10\%, 5\%, 2.5\%, 1\%\}$ on the ScanNet dataset, resulting in approximately 20K, 10K, 5K and 2K training images. Similarly, we also sample the SU3 dataset

[2] Code will be released upon acceptance.

Datasets	NYU Depth [39]		
Metrics	AUC@5°	AUC@10°	AUC@20°
J-Linkage[17]	49.30	61.28	70.08
CONSAC[26]	49.46	63.08	72.92
CONSAC[26] +Deep LSD[34]	46.75	61.11	72.32
Ours	**53.36**	**67.50**	**77.77**

Table 1. **Exp 2: Non-Manhattan world.** We report AUC scores on the NYU Depth dataset. In the non-Manhattan case, our model exceeds state-of-the-art models when detecting a varying number of vanishing points, even with only 1000 training images. This demonstrates the usefulness of our geometric priors.

into $\{50\%, 25\%, 10\%, 5\%, 2.5\%, 1\%\}$ subsets.

In Fig. 5 we compare the AA scores at $10°$ with state-of-the-art methods on the corresponding test sets. On the ScanNet dataset, our model visibly exceeds the other methods on the 10%, 5% and 2.5% subsets. In comparison, NeurVPS suffers of large accuracy decreases on small training data subsets. When decreasing the number of samples to 2K (1% subset), we still achieve competitive accuracy when compared to the non-learning methods, while NeurVPS fails to make reasonable prediction due to the lack of data. This shows the capability of our model to learn from limited data thanks to the added geometric priors. The NeurVPS model has $\times 3$ more parameters than our model due to a huge fully connected layer with 16M parameters. This may affect its accuracy on small subets. Therefore we also consider 'NeurVPS-7M' with a similar amount of parameters as our model, by reducing the number of neurons in its fully connected layers. However, both NeurVPS variants achieve similar results on various subsets. On the SU3 dataset, we observe a similar trend, as the the accuracy of NeurVPS decreases significantly when reducing the training dataset size, despite its superiority on the large training subsets. In comparison, our model shows a limited accuracy decrease when training data varies from 20K to 1K. Notably, on the 1% subset, with only 200 images for training, we are still able to achieve comparable performance with non-learning methods, indicating the effectiveness of the added geometric priors.

Systematically reducing the training data size, shows that our geometric priors are useful as they aid our model when the amount of training samples is limited. This geometric knowledge encoded in the priors no longer needs to be learned from data.

4.3. Exp 2: Non-Manhattan world

We compare with state-of-art methods in the more challenging no-Manhattan scenario, where the orthogonality constraint is no longer applicable for an unknown number of vanishing points. NeurVPS [57], Quasi-VP [32] and Contrario-VP [47] are not considered here, as they rely on the Manhattan assumption. The original CONSAC [26] method relies on line segment detection [52]. We also consider a variant of CONSAC with the more recent line detection method in [34]. Fig. 6 displays the recall at varying angle differences on the non-Manhattan NYU Depth dataset. Tab. 1 shows the angle accuracy on the NYU Depth dataset. The 'CONSAC + Deep LSD' has similar performance with the original CONSAC. Our model consistently outperforms state-of-the-art baselines, and the improvement is more pronounced for larger angular differences. The NYU Depth dataset only contains 1,000 images for training, which is significantly less than the SU3 and ScanNet datasets. However, our model outperforms existing methods by exploiting geometric priors. This validates the assumption that adding geometric priors improves the data efficiency.

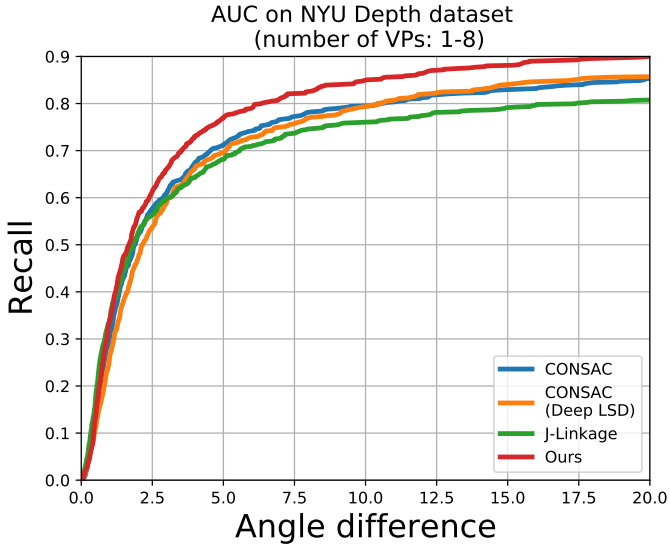


Figure 6. **Exp 2: Non-Manhattan world.** We plot the recall curve on the NYU Depth dataset containing 1 to 8 vanishing points. Our method performs visibly better than the state-of-the-art models. The NYU Depth dataset is small-scale with only 1000 training images and we train our model from scratch without any pre-training. This additionally illustrates the suitability of our geometric priors for detecting vanishing points.

4.4. Exp 3: Manhattan world

We compare our model with five state-of-the-art baselines [17, 47, 32, 57, 26] on the ScanNet, SU3 and YUD datasets. On the ScanNet and SU3 datasets, we train NeurVPS and our model from scratch on the full training split. On the YUD dataset, we pre-train both our model and the NeurVPS model on SU3 dataset, and fine-tune on all 25 training images. For CONSAC and J-Linkage, we select top 3 orthogonal predictions.

Tab. 2 shows the AA scores of all considered methods on the ScanNet, SU3 and YUD datasets, while Fig. 7 de-

Datasets			SU3 [58]		ScanNet [12]			YUD [14]		
Metrics	#Params	FPS	AA@3°	AA@5°	AA@5°	AA@10°	AA@20°	AA@5°	AA@10°	AA@20°
J-Linkage [17]	—	1.0	82.0	87.2	27.3	43.0	56.5	71.8	81.5	87.3
Contrario-VP [47]	—	0.6	64.8	72.2	21.4	35.3	48.9	70.7	81.8	88.5
Quasi-VP [32]	—	**29**	75.9	80.7	25.3	39.4	52.1	61.0	74.0	82.4
CONSAC [26]	0.2 M	3.0	81.4	87.1	18.9	30.3	41.7	71.8	82.7	89.3
NeurVPS [57]	22 M	0.5	**93.9**	**96.3**	**41.8**	**64.4**	**79.7**	73.3	84.0	90.5
Ours	7M	8.4	84.6	90.6	41.7	63.3	78.6	**75.2**	**86.8**	**93.0**

Table 2. **Exp 3: Manhattan world.** Angular accuracy on SU3, ScanNet and YUD datasets. Our model shows the best result on the the YUD dataset, and performs competitively on the larger ScanNet and SU3 datasets. This validates the usefulness of our proposed model.

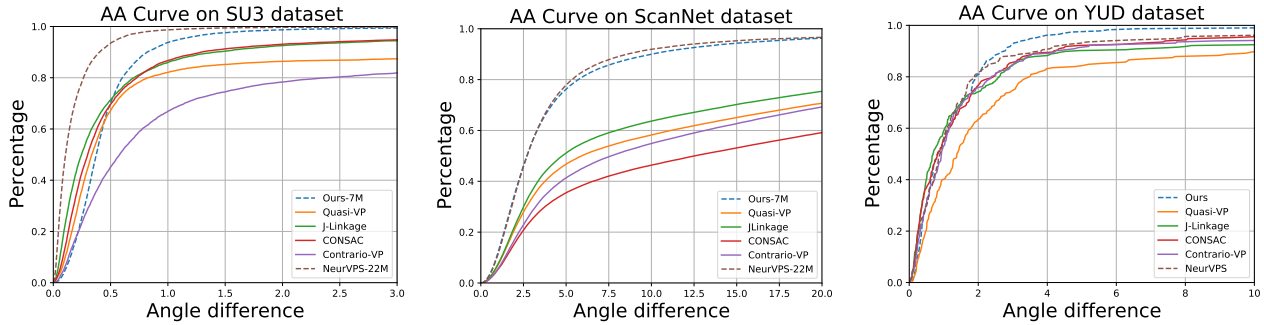


Figure 7. **Exp 3: Manhattan world.** AA curves on on the ScanNet, SU3 and YUD datasets containing 3 orthogonal vanishing points. Our model shows comparable results while using 3× less parameters. Deep learning-based approaches outperform methods relying purely on line segments detection and grouping. On the smaller YUD dataset, our model outperforms state-of-the-art, demonstrating the usefulness of the geometric priors in improving data efficiency.

picts the AA curve at different angle differences. The SU3 dataset is easier as most images contain strong geometric cues (e.g. sharp edges and contours); this is no longer the case in the ScanNet dataset. The prediction error on the more realistic ScanNet dataset is significantly larger for all methods. On the ScanNet dataset, NeurVPS and our model are visible better than the other methods relying on predefined line segments as inputs. The main advantage of NeurVPS and our model is their ability to learn useful feature representations directly from the images. Contrario-VP and Quasi-VP fail in some cases due to incorrect parameter settings. In comparison, the deep learning-based approaches estimate vanishing points more precisely in real-world scenarios, where large amounts of training data are available. On the SU3 dataset, NeurVPS exceeds the other methods in the low-error region (e.g. from 0° to 1°). J-Linkage, Quasi-VP and CONSAC have similar results, and all of them stabilize at 1°. Our model is less accurate in the 0°-1° error-range, yet it gradually compensates at 1° and is the best model at 3°. The inferior performance in the 0°-1° range results from the the quantization errors in Hough Transform and Sphere mapping.

Of the small-scale YUD dataset, there are only 25 images for fine-tuning [26, 14]. However, our model still achieves the best performance, indicating the generalization ability of our model in the small data regime.

4.5. Ablation studies, visualizations, and limitations

Ablation studies. Our model consists of two classic techniques: Hough Transform and Gaussian sphere mapping, and it is important to evaluate the impacts of them individually. We refer the readers to the supplementary material for ablation studies, where we show the effects of the two geometric priors quantitatively on the ScanNet dataset.

Visualizations. We visualize the predictions on the NYU Depth dataset in Fig. 8. We show the input images, labeled line segments and detected vanishing points on the hemisphere. Each color represents a group of lines and their corresponding vanishing point. In the third row, our model correctly detects all vanishing points, as the colored × and ○ overlap. In comparison, CONSAC fails to localize the red one and J-Linkage is unable to detect the green one, since their predictions are far away from the ground truth. In addition, CONSAC makes nearby predictions: e.g., the blue and pink × markers in second row. This is caused by the LSD [52] method producing a large number of outlier segments, resulting in incorrect predictions. Our method is suitable for real-world scenarios, where the image content varies substantially. We provide more visualizations on other datasets in the supplementary material.

Limitations. We pre-compute offline the mapping from the image domain to the Hough bins and fix the size of

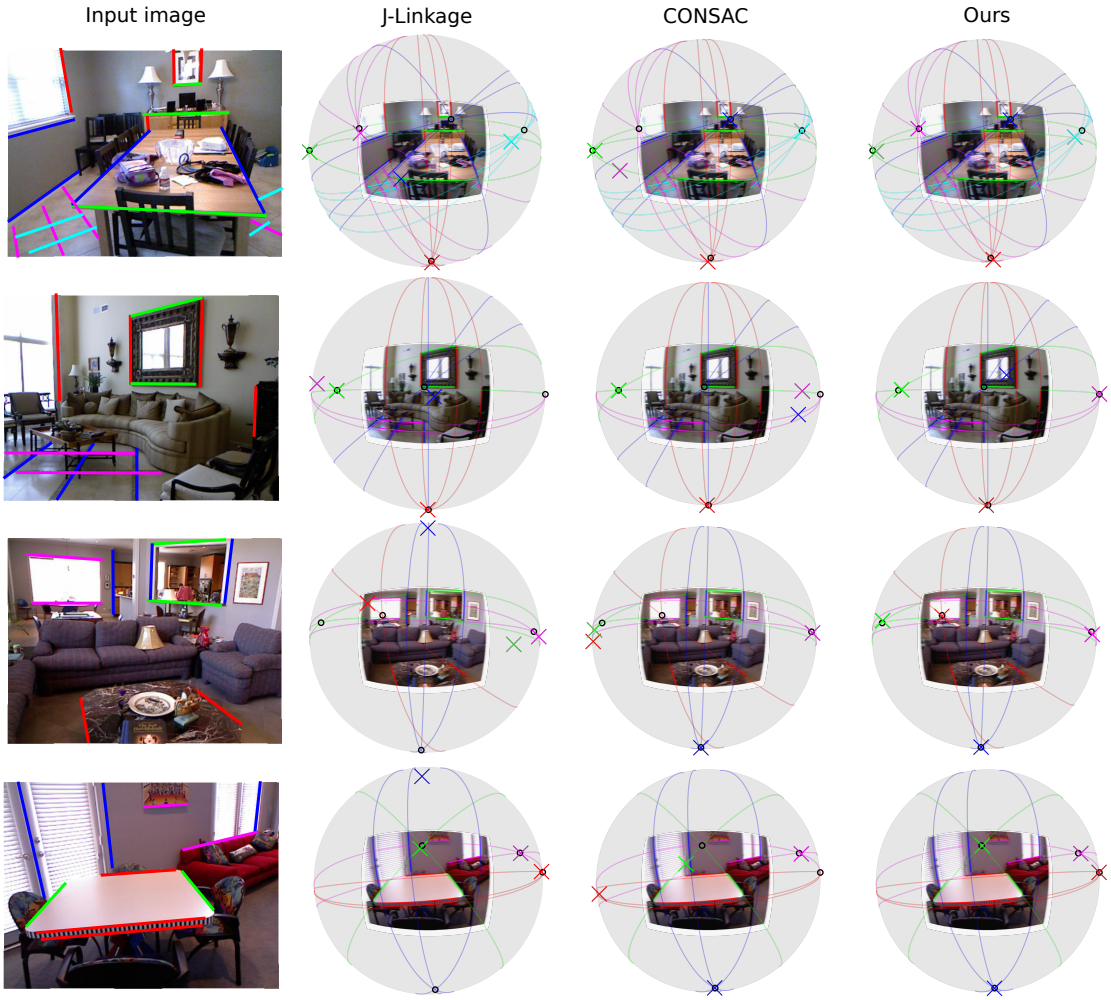


Figure 8. **Visualizations on the NYU Depth dataset.** The black ∘ represents the ground truth, while the colored × indicates predictions. Each color corresponds to a set of lines and their related vanishing point. Our model is better at localizing multiple vanishing points in the non-Manhattan world, having predictions (colored cross ×) closer to the ground truth (black ∘), while the predictions of the other methods scatter away from the ground truth, as shown in the first example.

the Hough histogram, as well as the Fibonacci sampling on the sphere. These samplings introduce quantization errors, which set an upper bound on the accuracy. This limits the accuracy of the predictions in the ranges of lower angular differences, as shown on the SU3 dataset. To further improve the result, future work should explore analytical mapping from image pixels to the Gaussian sphere. In addition, one may consider testing the added geometric priors for vanishing point detection in a weakly-supervised manner, as the prior knowledge no longer needs to be learned from the data.

5. Conclusion

This paper focuses on vanishing point detection relying on well-founded geometric priors. We add two geometric priors as building blocks in the deep neural networks for vanishing point detection: Hough Transform and Gaussian sphere mapping. We combine the learned image features with a Hough transform trainable module that effectively detects lines in the images. Subsequently, we map the detected lines in Hough space to the Gaussian sphere where lines are projected into great circles and vanishing points are found at the intersection of great circles. We precisely localize the vanishing points on the Gaussian sphere using spherical convolutions. Our proposed method is end-to-end trainable and is general, being applicable on tasks with a varying number of vanishing points. We validate experimentally the added value of our geometric priors and demonstrate improved data efficiency, as well as improved accuracy in the non-Manhattan scenario, where the challenge is to predict a varying number of vanishing points without the orthogonality assumption.

References

[1] Matthew E Antone and Seth Teller. Automatic recovery of relative camera rotations for urban scenes. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 2, pages 282–289. IEEE, 2000. 1, 2

[2] Michel Antunes and Joao P Barreto. A global approach for the detection of vanishing points and mutually orthogonal vanishing directions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1336–1343, 2013. 2

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

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

[5] Jean-Charles Bazin, Yongduek Seo, Cédric Demonceaux, Pascal Vasseur, Katsushi Ikeuchi, Inso Kweon, and Marc Pollefeys. Globally optimal line clustering and vanishing point estimation in manhattan world. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 638–645. IEEE, 2012. 2

[6] Jean-Charles Bazin, Yongduek Seo, and Marc Pollefeys. Globally optimal consensus set maximization through rotation search. In *Asian Conference on Computer Vision*, pages 539–551. Springer, 2012. 2

[7] Ali Borji. Vanishing point detection with convolutional neural networks. *CVPR Scene Understanding Workshop*, 2016. 1, 2

[8] 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, 2018. 1, 2

[9] 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

[10] Robert T Collins and Richard S Weiss. Vanishing point calculation as a statistical inference on the unit sphere. In *ICCV*, volume 90, pages 400–403, 1990. 2

[11] David F Crouse. On implementing 2d rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4):1679–1696, 2016. 5

[12] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017. 2, 5, 7

[13] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007. 1

[14] Patrick Denis, James H Elder, and Francisco J Estrada. Efficient edge-based methods for estimating manhattan frames in urban imagery. In *European conference on computer vision*, pages 197–210. Springer, 2008. 2, 5, 7

[15] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972. 1

[16] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996. 4

[17] 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, 7

[18] Alex Flint, David Murray, and Ian Reid. Manhattan scene understanding using monocular, stereo, and 3d features. In *2011 International Conference on Computer Vision*, pages 2228–2235. IEEE, 2011. 1

[19] Paolo Gamba, Alessandro Mecocci, and U Salvatore. Vanishing point detection by a voting scheme. In *Proceedings of 3rd IEEE International Conference on Image Processing*, volume 2, pages 301–304, 1996. 2

[20] Álvaro González. Measurement of areas on a sphere using fibonacci and latitude–longitude lattices. *Mathematical Geosciences*, 42(1):49, 2010. 3

[21] Lazaros Grammatikopoulos, George Karras, and Elli Petsa. An automatic approach for camera calibration from vanishing points. *ISPRS journal of photogrammetry and remote sensing*, 62(1):64–76, 2007. 1

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 4

[23] Kyungdon Joo, Tae-Hyun Oh, Junsik Kim, and In So Kweon. Robust and globally optimal manhattan frame estimation in near real time. *IEEE transactions on pattern analysis and machine intelligence*, 41(3):682–696, 2018. 2

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

[25] 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. 2

[26] 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, 6, 7

[27] Jana Košecká and Wei Zhang. Video compass. In *European conference on computer vision*, pages 476–490, 2002. 2

[28] 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

[29] 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

[30] Haoang Li, Pyojin Kim, Ji Zhao, Kyungdon Joo, Zhipeng Cai, Zhe Liu, and Yun-Hui Liu. Globally optimal and efficient vanishing point estimation in atlanta world. In *European Conference on Computer Vision*, pages 153–169. Springer, 2020. 2

[31] Haoang Li, Yazhou Xing, Ji Zhao, Jean-Charles Bazin, Zhe Liu, and Yun-Hui Liu. Leveraging structural regularity of atlanta world for monocular slam. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2412–2418. IEEE, 2019. 1

[32] 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. 1, 5, 6, 7

[33] Haoang Li, Ji Zhao, Jean-Charles Bazin, and Yun-Hui Liu. Quasi-globally optimal and near/true real-time vanishing point estimation in manhattan world. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 5

[34] Yancong Lin, Silvia L Pintea, and Jan C van Gemert. Deep hough-transform line priors. 2020. 2, 3, 6

[35] Evelyne Lutton, Henri Maitre, and Jaime Lopez-Krahe. Contribution to the determination of vanishing points using hough transform. *IEEE transactions on pattern analysis and machine intelligence*, 16(4):430–438, 1994. 2

[36] Michael J Magee and Jake K Aggarwal. Determining vanishing points from perspective images. *Computer Vision, Graphics, and Image Processing*, 26(2):256–267, 1984. 2

[37] GF McLean and D Kotturi. Vanishing point detection by line clustering. *IEEE Transactions on pattern analysis and machine intelligence*, 17(11):1090–1095, 1995. 2

[38] 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, 2011. 2

[39] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 2, 5, 6

[40] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016. 2, 3

[41] Phil L Palmer and A Tai. An optimised vanishing point detector. In *BMVC*, pages 1–10, 1993. 2

[42] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 5

[43] Long Quan and Roger Mohr. Determining perspective structures using hierarchical hough transform. *Pattern Recognition Letters*, 9(4):279–286, 1989. 2, 3, 4

[44] Klaus Hildebrandt Ruben Wiersma, Elmar Eisemann. Cnns on surfaces using rotation-equivariant features. *Transactions on Graphics*, 39(4), July 2020. 2

[45] Frederik Schaffalitzky and Andrew Zisserman. Planar grouping for automatic detection of vanishing lines and points. *Image and Vision Computing*, 18(9):647–658, 2000. 2

[46] 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 the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I, 2004. 2

[47] 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, 6, 7

[48] Marco Straforini, C Coelho, and Marco Campani. Extraction of vanishing points from images of indoor and outdoor scenes. *Image and Vision Computing*, 11(2):91–99, 1993. 2

[49] A Tai, Josef Kittler, Maria Petrou, and Terry Windeatt. Vanishing point detection. In *BMVC92*, pages 109–118. Springer, 1992. 2

[50] 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

[51] Roberto Toldo and Andrea Fusiello. Robust multiple structures estimation with j-linkage. In *European conference on computer vision*, pages 537–547. Springer, 2008. 2

[52] 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. 5, 6, 7

[53] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 4

[54] Horst Wildenauer and Allan Hanbury. Robust camera self-calibration from monocular images of manhattan worlds. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2831–2838, 2012. 2

[55] 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. 2

[56] 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, 2

[57] 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, 5, 6, 7

[58] 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. 2, 5, 7

[59] 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