Neural Geometric Parser for Single Image Camera Calibration*

Jinwoo Lee[1], Minhyuk Sung[2], Hyunjoon Lee[3], and Junho Kim[1]

[1]Kookmin University [2]Adobe Research [3]Intel

Abstract. We propose a neural geometric parser learning single image camera calibration for man-made scenes. Unlike previous neural approaches that rely only on semantic cues obtained from neural networks, our approach considers both semantic and geometric cues, resulting in significant accuracy improvement. The proposed framework consists of two networks. Using line segments of an image as geometric cues, the first network estimates the zenith vanishing point and generates several candidates consisting of the camera rotation and focal length. The second network evaluates each candidate based on the given image and the geometric cues, where prior knowledge of man-made scenes is used for the evaluation. With the supervision of datasets consisting of the horizontal line and focal length of the images, our networks can be trained to estimate the same camera parameters. Based on the Manhattan world assumption, we can further estimate the camera rotation and focal length in a weakly supervised manner. The experimental results reveal that the performance of our neural approach is significantly higher than that of existing state-of-the-art camera calibration techniques for single images of indoor and outdoor scenes.

Keywords: Single image camera calibration, Neural geometric parser, Horizon line, Focal length, Vanishing Points, Man-made scenes

1 Introduction

This paper deals with the problem of inferring camera calibration parameters from a single image. It is used in various applications of computer vision and graphics, including image rotation correction [11], perspective control [19], camera rotation estimation [35], metrology [8], and 3D vision [14, 21]. Due to its importance, single image camera calibration has been revisited in various ways.

Conventional approaches focus on reasoning vanishing points (VPs) in images by assembling geometric cues in the images. Most methods find straight line segments in the images using classic image processing techniques [13, 2] and then estimate the VPs by carefully selecting parallel or orthogonal segments in the 3D scene as geometric cues [19]. In practice, however, line segments found in images contain a large amount of noisy data, and it is therefore important to carefully select an inlier set of line segments for the robust detection of VPs [12,

* Corresponding author: Junho Kim (junho@kookmin.ac.kr)

(a) (b) (c) (d)

Fig. 1. Applications of the proposed framework: (a) image rotation correction, (b) perspective control, and virtual object insertions with respect to the (c) horizon and (d) VPs; before (top) and after (bottom).

26]. Because the accuracy of the inlier set is an important performance indicator, the elapsed time may exponentially increase if stricter criteria are applied to draw the inlier set.

Recently, several studies have proposed estimating camera intrinsic parameters using semantic cues obtained from deep neural networks. It has been investigated [29, 30, 16] that well-known backbone networks, such as ResNet [15] and U-Net [23], can be used to estimate the focal length or horizon line of an image without significant modifications of the networks. In these approaches, however, it is difficult to explain which geometric interpretation inside of the networks infers certain camera parameters. In several studies [35, 31], neural networks were designed to infer geometric structures; however, they required a new convolution operator [35] or 3D supervision datasets [31].

In this paper, we propose a novel framework for single image camera calibration that combines the advantages of both conventional and neural approaches. The basic idea is for our network to leverage line segments to reason camera parameters. We specifically focus on calibrating camera parameters from a single image of a man-made scene. By training with image datasets annotated with horizon lines and focal lengths, our network infers pitch, roll, and focal lengths (3DoF) and can further estimate camera rotations and focal lengths through three VPs (4DoF).

The proposed framework consists of two networks. The first network, the Zenith Scoring Network (ZSNet), takes line segments detected from the input image and deduces reliable candidates of parallel world lines along the zenith VP. Then, from the lines directed at the zenith VP, we generate candidate pairs consisting of a camera rotation and a focal length as inputs of the following step. The second network, the Frame Scoring Network (FSNet), evaluates the score of its input in conjunction with the given image and line segment information. Here, geometric cues from the line segments are used as prior knowledge about the man-made scenes in our network training. This allows us to obtain significant

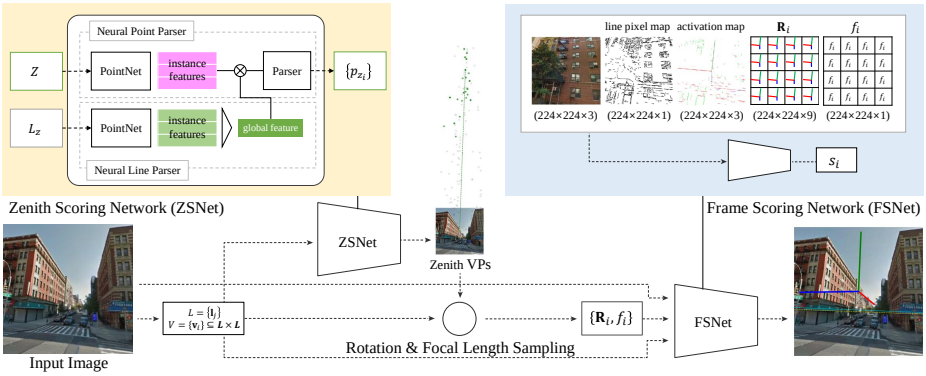


Fig. 2. Overview of the proposed neural geometric parser.

improvement over previous neural methods that only use semantic cues [30, 16]. Furthermore, it is possible to estimate camera rotation and focal length in a *weakly supervised* manner based on the Manhattan world assumption, as we reason camera parameters with pairs consisting of a camera rotation and a focal length. It should be noted that the ground truth for our supervisions is readily available with Google Street View [1] or with consumer-level devices possessing a camera and an inertial measurement unit (IMU) sensor, in contrast to the method in [31], which requires 3D supervision datasets.

2 Related Work

Projective geometry [14, 21] has historically stemmed from the study on imaging of perspective distortions occurring in the human eyes when one observes man-made architectural scenes [3]. In this regard, conventional methods of single image camera calibration [7, 18, 24, 10, 27, 28, 32, 19] involve extracting line segments from an image, inferring the combinations of world parallel or orthogonal lines, identifying two or more VPs, and finally estimating the rotation and focal length of the camera. LSD [13] or EDLine [2] were commonly used as effective line segment detectors, and RANSAC [12] or J-Linkage [26] were adopted to identify VPs describing as many of the extracted line segments as possible. Lee *et al*. [19] proposed robust estimation of camera parameters and automatic adjustment of camera poses to achieve perspective control. Zhai *et al*. [34] analyzed the global image context with a neural network to estimate the probability field in which the horizon line was formed. In their work, VPs were inferred with geometric optimization, in which horizontal VPs were placed on the estimated horizon line. Simon *et al*. [25] achieved better performance than Zhai *et al*. [34] by inferring the zenith VP with a geometric algorithm and carefully selecting a line segment orthogonal to the zenith VP to identify the horizon line. Li *et al*. [20] proposed a quasi-optimal algorithm to infer VPs from annotated line segments.

Recently, neural approaches have been actively studied to infer camera parameters from a single image using semantic cues learned by convolutional neural

networks. Workman *et al*. proposed DeepFocal [29] for estimating focal lengths and DeepHorizon [30] for estimating horizon lines using semantic analyses of images with neural networks. Hold-Geoffroy *et al*. [16] trained a neural classifier that jointly estimates focal lengths and horizons. They demonstrated that their joint estimation leads to more accurate results than those produced by independent estimations [29, 30]. Although they visualized how the convolution filters react near the edges (through the method proposed by Zeiler and Fergus [33]), it is difficult to intuitively understand how the horizon line is geometrically determined through the network. Therefore, [29, 30, 16] have a common limitation that it is non-trivial to estimate VPs from the network inference results. Zhou *et al*. [35] proposed NeurVPS that infers VPs with conic convolutions for a given image. However, NeurVPS [35] assumes normalized focal lengths and does not estimate focal lengths.

Inspired by UprightNet [31], which takes geometric cues into account, we propose a neural network that learns camera parameters by leveraging line segments. Our method can be compared to Lee *et al*. [19] and Zhai *et al*. [34], where line segments are used to infer the camera rotation and focal length. However, in our proposed method, the entire process is designed with neural networks. Similar to Workman *et al*. [30] and Hold-Geoffroy *et al*. [16], we utilize semantic cues from neural networks but our network training differs in that line segments are utilized as prior knowledge about man-made scenes. Unlike UprightNet [16], which requires the supervisions of depth and normal maps for learning roll/pitch (2DoF), the proposed method learns the horizon and focal length (3DoF) with supervised learning and the camera rotation and focal length (4DoF) with weakly supervised learning.

The relationship between our proposed method and the latest neural RANSACs [5, 6, 17] is described below. Our ZSNet is related to neural-guided RANSAC [6] in that it updates the line features with backpropagation when learning to sample zenith VP candidates. In addition, our FSNet is related to DSAC [5] in that it evaluates each input pair consisting of a camera rotation and focal length based on the hypothesis on man-made scenes. Our work differs from CONSAC [17], which requires the supervision of all VPs, as we focus on learning single image camera calibrations from the supervision of horizons and focal lengths.

3 Neural Geometric Parser for Camera Calibration

From a given input image, our network estimates up to four camera intrinsic and extrinsic parameters; the focal length f and three camera rotation angles ψ, θ, ϕ. Then, a 3D point $(P_x, P_y, P_z)^T$ in the world coordinate is projected onto the image plane as follows:

$$\begin{bmatrix} p_x \\ p_y \\ p_w \end{bmatrix} = (\mathbf{KR}) \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}, \text{ where } \mathbf{K} = \begin{bmatrix} f & 0 & c_u \\ 0 & f & c_v \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \mathbf{R} = \mathbf{R}_\psi \mathbf{R}_\theta \mathbf{R}_\phi, \quad (1)$$

where $(p_x, p_y, p_w)^T$ represents the mapped point in the image space, and $\mathbf{R}_\psi$, $\mathbf{R}_\theta$, $\mathbf{R}_\phi$ represent the rotation matrices along x-, y-, and z-axes, with rotation

angles ψ, θ, ϕ, respectively. The principal point is assumed to be on the image center such that $c_u = W/2$ and $c_v = H/2$, where W and H represent the width and height of the image, respectively.

Under the Manhattan world assumption, calibration can be done once we obtain the *Manhattan directions*, which are three VPs corresponding to x-, y-, and z-directions in 3D [7]. In Sec. 3.1, we describe how to extract a set of candidate VPs along the zenith direction. Then, in Sec. 3.2, we present our weakly supervised method for estimating all three directions and calibrating the camera parameters.

We use LSD [13] as a line segment detector in our framework. A line segment is represented by a pair of points in the image space. Before estimating the focal length in Sec. 3.2, we assume that each image is transformed into a pseudo camera space as $\mathbf{p} = \mathbf{K}_p^{-1} (p_x, p_y, p_w)^T$, where $\mathbf{K}_p$ represents a pseudo camera intrinsic matrix of $\mathbf{K}$, built by assuming f as $\min(W, H)/2$.

3.1 Zenith Scoring Network (ZSNet)

We first explain our ZSNet, which is used to estimate the zenith VP (see Fig. 2 top-left). Instead of searching for a single zenith VP, we extract a set of candidates that are sufficiently close to the ground truth.

Similar to PointNet [22], ZSNet takes sets of unordered vectors in 2D homogeneous coordinates - line equations and VPs - as inputs. Given a line segment, a line equation $\mathbf{l}$ can be computed as a cross product of its two endpoints:

$$\mathbf{l} = [\mathbf{p}_0]_\times \mathbf{p}_1, \tag{2}$$

where $[\cdot]_\times$ represents a skew-symmetric matrix of a vector. A candidate VP $\mathbf{v}$ can then be computed as an intersection point of the two lines:

$$\mathbf{v} = [\mathbf{l}_0]_\times \mathbf{l}_1. \tag{3}$$

Motivated by [25], we sample a set of line equations roughly directed to the zenith $L_z = \{\mathbf{l}_0, \ldots, \mathbf{l}_{|L_z|}\}$ from the line segments, using the following equation:

$$\left| \tan^{-1}\left(-\frac{a}{b}\right) \right| > \tan^{-1}(\delta_z), \tag{4}$$

where $\mathbf{l} = (a, b, c)^T$ represents a line equation as in Eq. (2) and the angle threshold δ_z is set to 67.5° as recommended in [25]. Then, we randomly select pairs of line segments from L_z and compute their intersection points as in Eq. (3) to extract a set of zenith VP candidates $Z = \{\mathbf{z}_0, \ldots, \mathbf{z}_{|Z|}\}$. Finally, we feed L_z and Z to ZSNet. We set both the number of samples, $|L_z|$ and $|Z|$, 256 in the experiments.

The goal of our ZSNet is to score each zenith candidate in Z; 1 if a candidate is sufficient close to the ground truth zenith, and 0 otherwise. Fig. 2 top-left shows the architecture of our ZSNet.

In the original PointNet [22], each point is processed independently, except for transformer blocks, to generate point-wise features. A global max pooling

layer is then applied to aggregate all the features and generate a global feature. The global feature is concatenated to each point-wise feature, followed by several neural network blocks to classify/score each point.

In our network, we also feed the set of zenith candidates Z to the network, except that we do not compute the global feature from Z. Instead, we use another network, feeding the set of line equations L_z, to extract the global feature of L_z that is then concatenated with each point-wise feature of Z (Fig. 2, top-left).

Let $h_z(\mathbf{z}_i)$ be a point-wise feature of the point $\mathbf{z}_i$ in Z, where $h_z(\cdot)$ represents a PointNet feature extractor. Similarly, let $h_l(L_z) = \left\{h_l(\mathbf{l}_0), \ldots, h_l(\mathbf{l}_{|L_z|})\right\}$ be a set of features of L_z. A global feature $\mathbf{g}_l$ of $h_l(L_z)$ is computed via a global max-pooling operation (gpool), and is concatenated to $h_z(\mathbf{z}_i)$ as follows:

$$\mathbf{g}_l = \text{gpool}\left(h_l(L_z)\right) \tag{5}$$

$$h'_z(\mathbf{z}_i) = \mathbf{g}_l \otimes h_z(\mathbf{z}_i), \tag{6}$$

where $\otimes$ represents the concatenation operation.

Finally, the concatenated features are fed into a scoring network computing $[0, 1]$ scores such that:

$$p_{z_i} = \text{sigmoid}\left(s_z(h'_z(\mathbf{z}_i))\right), \tag{7}$$

where p_{z_i} represents the computed scores of each zenith candidate. The network $s_z(\cdot)$ in Eq. (7) consists of multiple MLP layers, similar to the latter part of the PointNet [22] segmentation architecture.

To train the network, we assign a ground truth label y_i to each zenith candidate $\mathbf{z}_i$ using the following equation:

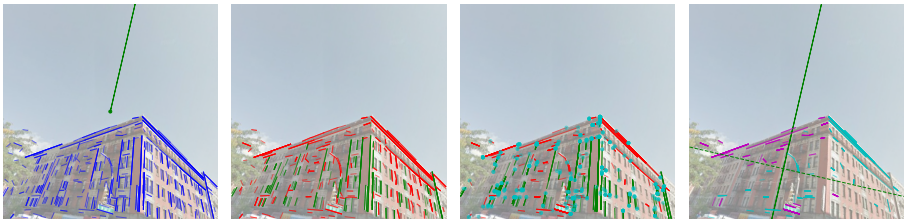
$$y_i = \begin{cases} 1 & \text{if } \text{cossim}(\mathbf{z}_i, \mathbf{z}_{gt}) > \cos(\delta_p) \\ 0 & \text{if } \text{cossim}(\mathbf{z}_i, \mathbf{z}_{gt}) < \cos(\delta_n) \end{cases}, \tag{8}$$

where $\text{cossim}(x, y) = \frac{|x \cdot y|}{\|x\|\|y\|}$ and $\mathbf{z}_{gt}$ represents the ground truth zenith. The two angle thresholds δ_p and δ_n are empirically selected as 2° and 5°, respectively, from our experiments. The zenith candidates each of which y_i is undefined are not used in the training. The cross entropy loss is used to train the network as follows:

$$\mathcal{L}_{cls} = \frac{1}{N} \sum_{i}^{N} -y_i \log(p_{z_i}). \tag{9}$$

To better train our ZSNet we use another loss in addition to the cross entropy. Specifically, we constrain the weighted average of zenith candidates close to the ground truth, where estimated scores p_{z_i} are used as weights. To average the zenith candidates, which represent vertical directions of the scene, we use structure tensors of l_2 normalized 2D homogeneous points. Given a 2D homogeneous point $\mathbf{v} = (v_x, v_y, v_w)^T$, a structure tensor of the normalized point is computed as follows:

$$\text{ST}(\mathbf{v}) = \frac{1}{\left(v_x^2 + v_y^2 + v_w^2\right)} \begin{bmatrix} v_x^2 & v_x v_y & v_x v_w \\ v_x v_y & v_y^2 & v_y v_w \\ v_x v_w & v_y v_w & v_w^2 \end{bmatrix}, \tag{10}$$



(a) (b) (c) (d)

Fig. 3. Sampling horizontal line segments into two groups: (a) Based on a zenith VP representative (green), we want to classify the line segments (blue) of a given image. (b) Line segments are classified as follows: vanishing lines of the zenith VP (green) and the remaining lines (red). (c) Junction points (cyan) are computed as the intersections of spatially adjacent line segments that are classified differently; line segments whose endpoints are close to junction points are selected. (d) Using a pseudo-horizon (dotted line), we divide horizontal line segments into two groups (magenta and cyan).

The following loss is used in our network:

$$\mathcal{L}_{loc} = \left\| \mathrm{ST}(\mathbf{z}_{gt}) - \overline{\mathrm{ST}}(\mathbf{z}) \right\|_F, \quad \overline{\mathrm{ST}}(\mathbf{z}) = \frac{\sum_i p_{z_i} \mathrm{ST}(\mathbf{z}_i)}{\sum_i p_{z_i}} \tag{11}$$

where $\| \cdot \|_F$ represents the Frobenius norm. Finally, we select zenith candidates whose scores p_{z_i} are larger than δ_c to the set of zenith candidates, as:

$$Z_c = \{ \mathbf{z}_i \mid p_{z_i} > \delta_c \}, \tag{12}$$

where $\delta_c = 0.5$ in our experiments. The set Z_c is then used in our FSNet.

3.2 Frame Scoring Network (FSNet)

After we extract a set of zenith candidates, we estimate the remaining two horizontal VPs taking into account the given set of zenith VP candidates. We first generate a set of hypotheses on all three VPs. Each hypothesis is then scored by our FSNet.

To sample horizontal VPs, we first filter the input line segments. However, we cannot simply filter line segments using their directions in this case, as there may be multiple horizontal VPs, and lines in any directions may vanish in the horizon. As a workaround, we use a heuristic based on the characteristics of most urban scenes.

Many man-made structures contain a large number of rectangles (e.g., facades or windows of a building) that are useful for calibration parameter estimation, and line segments enclosing these rectangles create junction points. Therefore, we sample horizontal direction line segments by only using their endpoints when they are close to the endpoints of the estimated vertical vanishing lines.

Fig. 3 illustrates the process of sampling horizontal line segments into two groups. Let $\mathbf{z}_{est} = (z_x, z_y, z_w)$ be a representative of the estimated zenith VPs,

which is computed as the eigenvector with the largest eigenvalue of $\overline{\mathrm{ST}}(\mathbf{z})$ in Eq. (11). We first draw a pseudo-horizon by using $\mathbf{z}_{est}$ and then compute the intersection points between each sampled line segment and the pseudo-horizon. Finally, using a line connecting $\mathbf{z}_{est}$ and the image center as a pivot, we divide horizontal line segments into two groups; one that intersects the pseudo-horizon on the left side of the pivot and the other that intersects the pseudo-horizon on the right side of the pivot. The set of horizontal VP candidates is composed of intersection points by randomly sampling pairs of horizontal direction line segments in each group. We sample an equal number of candidates for both groups.

Once the set of horizontal VP candidates is sampled, we sample candidates of Manhattan directions. To sample each candidate, we draw two VPs; one from zenith candidates and the other from either set of horizontal VP candidates. The calibration parameters for the candidate can then be estimated by solving Eq. (1) with the two VPs, assuming that the principal point is on the image center [18].

We design our FSNet for inferring camera calibration parameters to utilize all the available data, including VPs, lines, and the original raw image (Fig. 2, top-right). ResNet [15] is adapted to our FSNet to handle raw images, appending all the other data as additional color channels. To append the information of the detected line segments, we rasterize line segments as a binary line segment map whose width and height are the same as those of the input image, as follows:

$$\mathbf{L}(u,v) = \begin{cases} 1 & \text{if a line } \mathbf{l} \text{ passes through } (u,v) \\ 0 & \text{otherwise} \end{cases}, \tag{13}$$

where (u, v) represents a pixel location of the line segment map. We also append the information of vanishing line segments (i.e., expanding lines are close to a VP) as a weighted line segment map for all three VPs of a candidate, where weights are computed using the closeness between the line segments and VPs. For a given VP $\mathbf{v}$ and the line equation $\mathbf{l}$ of a line segment, we compute the closeness between $\mathbf{v}$ and $\mathbf{l}$ using the conventional line-point distance as follows:

$$\text{closeness}\,(\mathbf{l},\mathbf{v}) = 1 - \frac{|\mathbf{l}\cdot\mathbf{v}|}{\|\mathbf{l}\|\,\|\mathbf{v}\|}. \tag{14}$$

Three activation maps are drawn for each candidate (x-, y- and z-directions), as:

$$\mathbf{A}_{\{x|y|z\}}(u,v) = \begin{cases} \text{closeness}(\mathbf{l},\mathbf{v}_{\{x|y|z\}}) & \text{if a line } \mathbf{l} \text{ passes through } (u,v) \\ 0 & \text{otherwise} \end{cases}. \tag{15}$$

All the maps are appended to the original image as illustrated in Fig. 2. Finally, we append the Manhattan directions and the estimated focal length to each pixel of the concatenated map so that the input to the scoring network have size of $(\textit{height}, \textit{width}, \textit{channel}) = (224, 224, 17)$ (Fig. 2, top-right).

To train FSNet, we assign GT score to each candidate by measuring similarities between horizon and zenith of each candidate and those of the GT. For the zenith, we measure the cosine similarities of GT zenith and that of candidate as follows:

$$s_{z_i} = \text{cossim}(\mathbf{z}_{gt}, \mathbf{z}_i), \tag{16}$$

where $\mathbf{z}_{gt}$ and $\mathbf{z}_i$ represent the GT and candidate zenith. For the horizon, we adapt the distance metric proposed in [4]. For this, we compute the intersection points between the GT and candidate horizons and left/right image boundaries. Let $\mathbf{h}_l$ and $\mathbf{h}_r$ be intersection points of the predicted horizon and left/right border of the image. Similarly, we compute $\mathbf{g}_l$ and $\mathbf{g}_r$ using the ground truth horizon. Inspired by [4], the similarity between the GT and a candidate is computed as:

$$s_{h_i} = \exp\left(-\max\left(\|\mathbf{h}_{l_i} - \mathbf{g}_l\|_1, \|\mathbf{h}_{r_i} - \mathbf{g}_r\|_1\right)^2\right). \tag{17}$$

Our scoring network h_{score} is then trained with the cross entropy loss, defined as:

$$\mathcal{L}_{score} = \sum_i -h_{score}(\mathbf{R}_i)\log(c_i) \tag{18}$$

$$c_i = \begin{cases} 0 & \text{if } s_{vh_i} < \delta_s \\ 1 & \text{otherwise} \end{cases} \tag{19}$$

$$s_{vh_i} = \exp\left(-\frac{\left(\frac{s_{h_i}+s_{v_i}}{2} - 1.0\right)^2}{2\sigma^2}\right), \tag{20}$$

where $\sigma = 0.1$ and $\delta_s = 0.5$ in our experiments.

Robust score estimation using the Manhattan world assumption. Although our FSNet is able to accurately estimate camera calibration parameters in general, it can sometimes be noisy and unstable. In our experiments, we found that incorporating with the Manhattan world assumption increased the robustness of our network. Given a line segment map and three closeness maps (Eqs. (14) and (15)), we compute the extent to which a candidate follows the Manhattan world assumption using Eq. (21):

$$m_i = \frac{\sum_u \sum_v \max\left(\mathbf{A}_x(u,v), \mathbf{A}_y(u,v), \mathbf{A}_z(u,v)\right)}{\sum_u \sum_v \mathbf{L}(u,v)}, \tag{21}$$

and the final score of a candidate is computed as:

$$s_i = s_{vh_i} \cdot m_i. \tag{22}$$

Once all the candidates are scored, we estimate the final focal length and zenith by averaging those of top-k high score candidates such that:

$$f_{est} = \frac{\sum_i s_i f_i}{\sum_i s_i} \quad \text{and} \quad \overline{\text{ST}}_{est} = \frac{\sum_i s_i \text{ST}(z_i)}{\sum_i s_i}, \tag{23}$$

where $\text{ST}(\mathbf{z}_i)$ represents the structure tensor of a zenith candidate (Eq. (10)). We set $k = 8$ in the experiments. $\mathbf{z}_{est}$ can be estimated from $\overline{\text{ST}}_{est}$, and the two camera rotation angles ψ and ϕ in Eq. (1) can be computed from f_{est} and $\mathbf{z}_{est}$. For the rotation angle θ, we simply take the value from the highest score candidate, as there may be multiple pairs of horizontal VPs that are not close to each other, particularly when the scene does not follow the Manhattan world assumption but the Atlanta world assumption. Note that they still share similar zeniths and focal lengths [4, 19].

3.3 Training Details and Runtime

In training, we train ZSNet first and then train FSNet with the outputs of ZSNet. Both networks are trained with Adam optimizer with initial learning rates of 0.001 and 0.0004 for ZSNet and FSNet, respectively. Both learning rates are decreased by half for every 5 epochs. The mini-batch sizes of ZSNet and FSNet are set to 16 and 2, respectively. The input images are always downsampled to 224×224, and the LSD [13] is computed on these low-res images.

At test time, it takes $\sim$0.08s for data preparation with Intel Core i7-7700K CPU and another $\sim$0.08s for ZSNet/FSNet per image with Nvidia GeForce GTX 1080 Ti GPU.

4 Experiments

We provide the experimental results with Google Street View [1] and HLW [30] datasets. Refer to the supplementary material for more experimental results with the other datasets and more qualitative results.

Google Street View [1] dataset. It provides panoramic images of outdoor city scenes for which the Manhanttan assumption is satisfied. For generating training and test data, we first divide the scenes for each set and rectify and crop randomly selected panoramic images by sampling FoV, pitch, and roll in the ranges of $40 \sim 80°$, $-30 \sim 40°$, and $-20 \sim 20°$, respectively. 13,214 and 1,333 images are generated for training and test sets, respectively.

HLW [30] dataset. It includes images only with the information of horizon line but no other camera intrinsic parameters. Hence, we use this dataset only for verifying the generalization capability of methods at test time.

Evaluation Metrics. We measure the accuracy of the output camera up vector, focal length, and horizon line with several evaluation metrics. For camera up vector, we measure the difference of angle, pitch, and roll with the GT. For the focal length, we first convert the output focal length to FoV and measure the angle difference with the GT. Lastly, for the horizon line, analogous to our similarity definition in Eq. (17), we measure the distances between the predicted and GT lines at the left/right boundary of the input image (normalized by the image height) and take the maximum of the two distances. We also report the area under curve (AUC) of the cumulative distribution with the x-axis of the distance and the y-axis of the percentage, as introduced in [4]. The range of x-axis is $[0, 0.25]$.

4.1 Comparisons

We compare our method with six baseline methods, where Table 1 presents the required supervision characteristics and outputs. Upright [19] and A-Contrario Detection [25] are non-neural-net methods based on line detection and RANSAC. We use the authors' implementations in our experiments. For Upright [19], the evaluation metrics are applied after optimizing the Manhattan direction per image,

Table 1. Supervision and output characteristics of baseline methods and ours. The first two are unsupervised methods not leveraging neural networks, and the others are deep learning methods. Ours is the only network-based method predicting all four outputs.

Method	Supervision				Output			
	Horizon Line	Focal Length	Camera Rotation	Per-Pixel Normal	Horizon Line	Focal Length	Camera Rotation	Up Vector
Upright [19]	N/A				✓	✓	✓	✓
A-Contrario [25]					✓	✓	✓	✓
DeepHorizon [30]	✓				✓			
Perceptual [16]	✓	✓			✓	✓		✓
UprightNet [31]			✓	✓				✓
Ours	✓	✓			✓	✓	✓	✓

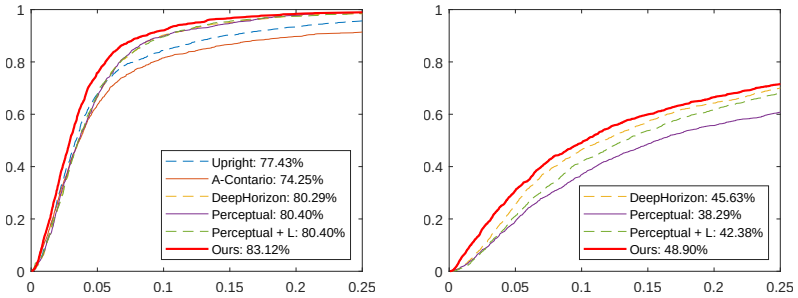
assuming the principal point as the image center. A-Contrario Detection [25] often fails to estimate focal length when horizon VP candidates are insufficient in the input. Thus, we exclude [25] in the evaluation of FoV. The AUC of [25] is measured, regardless of the failures in focal length estimations, with the horizon lines estimated by the method proposed in [25]. The other metrics of [25] are evaluated only for the cases that focal lengths are successfully estimated. DeepHorizon [30] and Perceptual Measure [16] are neural-network-based methods directly taking the global feature of an image and performing classifications in discretized camera parameter spaces. For fair comparisons, we use ResNet [15] as a backbone architecture in the implementations of these methods. Note that DeepHorizon [30] does not predict focal length, and thus we use ground truth focal length in the estimation of the camera up vector. Additionally, we train Perceptual Measure [16] by feeding it both the input image and the line map used in our FSNet (Sec. 3.2), and we assess whether the extra input improves the performance. UprightNet [31] is another deep learning method that requires additional supervision in training, such as camera extrinsic parameters and per-pixel normals in the 3D space. Due to the lack of such supervision in our datasets, in our experiments, we use the author's pretrained model on ScanNet [9], which is a synthetic dataset.

The quantitative results with Google Street View dataset [1] are presented in Table 2. The results demonstrate that our method outperforms all the baseline methods in most evaluation metric. Upright [19] provides a slightly lower median roll than ours, although its mean roll is much greater than the median, meaning that it completely fails in some test cases. In addition, Perceptual Measure [16] gives a slightly smaller mean FoV error; however, the median FoV error is higher than ours. When Perceptual Measure [16] is trained with the additional line map input, the result indicates that it does not lead to a meaning difference in performance. As mentioned earlier, a pretrained model is used for UprightNet [31] (trained on ScanNet [9]) due to the lack of required supervision in Google Street View; thus the results are much poorer than others.

Fig. 5 visualizes several examples of horizon line predictions as well as our weakly-supervised Manhattan directions. Recall that we do not use *full* supervision of the Manhattan directions in training; we only use the supervision of

Table 2. Quantitative evaluations with Google Street View dataset [1]. See **Evaluation Metrics** in Sec. 4 for details. Bold is the best result, and underscore is the second-best result. Note that, for DeepHorizon [30]*, we use GT FoV to calculate the camera up vector (angle, pitch, and roll errors) from the predicted horizon line. Also, for UprightNet [31]**, we use a pretrained model on ScanNet [9] due to the lack of required supervision in the Google Street View dataset.

Method	Angle (°) ↓		Pitch (°) ↓		Roll (°) ↓		FoV (°) ↓		AUC (%) ↑
	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	
Upright [19]	3.05	1.92	2.90	1.80	6.19	**0.43**	9.47	4.42	77.43
A-Contrario [25]	3.93	1.85	3.51	1.64	13.98	0.52	-	-	74.25
DeepHorizon [30]*	3.58	3.01	2.76	2.12	1.78	1.67	-	-	80.29
Perceptual [16]	2.73	2.13	2.39	1.78	0.96	0.66	**4.61**	3.89	80.40
Perceptual [16] +**L**	2.66	2.10	2.31	1.80	0.92	0.93	5.27	3.99	80.40
UprightNet [31]**	28.20	26.10	26.56	24.56	6.22	4.33	-	-	-
Ours	**2.12**	**1.61**	**1.92**	**1.38**	**0.75**	0.47	6.01	**3.72**	**83.12**



(a) Google Street View (b) HLW

Fig. 4. Comparison of the cumulative distributions of the horizon line error and their AUCs tested on (a) Google Street View and (b) HLW. Note that, in (b), neural approaches are trained with the Google Street View training dataset and to demonstrate the generalization capability. The AUCs in (a) are also reported in Table 2.

horizon lines and focal lengths. In each example in Fig. 5, we illustrate the Manhattan direction of the highest score candidate.

To evaluate the generalization capability of neural-network-based methods, we also take the network models trained on Google Street View training dataset and test them on the HLW dataset [30]. Because the HLW dataset only has the GT horizon line, Fig. 4(b) only reports the cumulative distributions of the horizon prediction errors. As shown in the figure, our method provides the largest AUC with a significant margin compared with the other baselines. Interestingly, Perceptual Measure [16] shows improvement when trained with the additional line map, meaning that the geometric interpretation helps more when parsing *unseen* images in network training.

We conduct an experiment comparing the outputs of ZSNet with the outputs of Upright [19] and A-Contrario [25]. Using the weighted average of the zenith candidates (in Eq. (11)), we measured the angle to the GT zenith $\text{cossim}(z_i, z_{gt})$, as provided in Eq. (8). Table 3 show that our ZSNet computes the zenith VP more accurately than the other non-neural-net methods.

Fig. 5. Examples of horizon line prediction on the Google Street View test set (top two rows) and on the HLW test set (bottom two rows). Each example also shows the Manhattan direction of the highest score candidate.

Table 3. Evaluation of ZSNet.

	Angle (°) ↓	
	Mean	Med.
ZSNet (Ours)	**2.53**	**1.52**
Upright [19]	3.15	2.11
A-Contrario [25]	3.06	2.38

Fig. 6. Visualizations of FSNet focus: (left) input; (right) feature highlight.

Fig. 6 visualizes the weights of the second last convolution layer in FSNet (the layer in the ResNet backbone); red means high, and blue means low. It can be seen that our FSNet focused on the areas with many line segments, such as buildings, window frames, and pillars. The supplementary material contains more examples.

4.2 Ablation Study

We conduct an ablation study using Google Street View dataset to demonstrate the effect of each component in our framework. All results are reported in Table 4, where the last row shows the result of our final version framework.

We first evaluate the effect of the entire ZSNet by ablating it in the training. When sampling zenith candidates in FSNet (Sec. 3.2), the score p_{z_i} in Eq. (7)

Table 4. Ablation study results. Bold is the best result. See Sec. 4.2 for details.

	Angle (°) ↓		Pitch (°) ↓		Roll (°) ↓		FoV (°) ↓		AUC
	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	(%) ↑
w/o ZSNet	3.00	2.04	2.81	1.98	1.62	0.95	8.42	4.47	74.01
$h'_z(\mathbf{z}_i) = h_z(\mathbf{z}_i)$ (Eq. (6))	4.34	1.96	3.91	1.76	1.64	0.59	7.88	4.16	77.65
FSNet−Image	2.45	1.78	2.19	1.52	**0.68**	**0.47**	6.71	4.35	80.20
FSNet−$\mathbf{L}-\mathbf{A}$	3.74	2.22	3.09	1.91	1.68	0.66	8.26	5.40	74.31
$s_i = s_{vh_i}$ (Eq. (22))	2.32	1.80	2.09	1.57	0.72	0.54	6.06	4.12	80.85
Ours	**2.12**	**1.61**	**1.92**	**1.38**	0.75	**0.47**	**6.01**	**3.72**	**83.12**

is not predicted but set uniformly. The first row of Table 4 indicates that the performance significantly decreases in all evaluation metrics; e.g., the AUC decreased by 9%. This indicates that ZSNet plays an important role in finding inlier Zenith VPs that satisfy the Manhattan/Atlanta assumption. In ZSNet, we also evaluate the effect of global line feature $\mathbf{g}_l$ in Eq. (5) by not concatenating it with the point feature $h_z(\mathbf{z}_i)$ in Eq. (6). Without the line feature, ZSNet is still able to prune the outlier zeniths in some extent, as indicated in the second row, but the performance is far inferior to that of our final framework (the last row). This result indicates that the equation of a horizon line is much more informative than the noisy coordinates of the zenith VP.

In FSNet, we first ablate some parts of the input fed to the network per frame. When we do not provide the given image but the rest of the input (the third row of Table 4), the performance decreases somewhat; however, the change is less significant than when omitting the line map $\mathbf{L}$ (Eq. (13)) and the activation map $\mathbf{A}$ (Eq. (15)) in the input (the fourth row). This demonstrates that FSNet learns more information from the line map and activation map, which contain explicit geometric interpretations of the input image. The combination of the two maps (our final version) produces the best performance. In addition, the results get worse when the activation map score m_i is not used in the final score of candidates — i.e., $s_i = s_{vh_i}$ in Eq. (22) (the fifth row).

5 Conclusion

In this paper, we introduced a neural method that predicts camera calibration parameters from a single image of a man-made scene. Our method fully exploits line segments as prior knowledge of man-made scenes, and in our experiments, it exhibited better performance than that of previous approaches. Furthermore, compared to previous neural approaches, our method demonstrated a higher generalization capability to unseen data. In future work, we plan to investigate neural camera calibration that considers a powerful but small number of geometric cues through analyzing image context, as humans do.

Acknowledgements. This research was supported by the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2017R1D1A1B03034907).

References

1. Google Street View Images API. https://developers.google.com/maps/
2. Akinlar, C., Topal, C.: EDLines: A real-time line segment detector with a false detection control. Pattern Recognition Letters **32**(13), 1633–1642 (2011)
3. Alberti, L.B.: Della Pittura (1435)
4. Barinova, O., Lempitsky, V., Tretiak, E., Kohli, P.: Geometric Image Parsing in Man-Made Environments. In: Proc. ECCV. pp. 57–70 (2010)
5. Brachmann, E., Krull, A., Nowozin, S., Shotton, J., Michel, F., Gumhold, S., Rother, C.: DSAC — Differentiable RANSAC for Camera Localization. In: Proc. CVPR. pp. 6684–6692 (2017)
6. Brachmann, E., Rother, C.: Neural-Guided RANSAC: Learning Where to Sample Model Hypotheses. In: Proc. ICCV. pp. 4322–4331 (2019)
7. Coughlan, J.M., Yuille, A.L.: Manhattan World: Compass Direction from a Single Image by Bayesian Inference. In: Proc. ICCV. pp. 941–947 (1999)
8. Criminisi, A., Reid, I., Zisserman, A.: Single View Metrology. International Journal of Computer Vision **40**(2), 123–148 (2000)
9. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In: Proc. CVPR. pp. 5828–5839 (2017)
10. Denis, P., Elder, J.H., Estrada, F.J.: Efficient Edge-Based Methods for Estimating Manhattan Frames in Urban Imagery. In: Proc. ECCV. pp. 197–210 (2008)
11. Fischer, P., Dosovitskiy, A., Brox, T.: Image Orientation Estimation with Convolutional Networks. In: Proc. GCPR. pp. 368–378 (2015)
12. Fischler, M.A., Bolles, R.C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Communications of the ACM **24**(6), 381–395 (1981)
13. von Gioi, R.G., Jakubowicz, J., Morel, J.M., Randall, G.: LSD: A Fast Line Segment Detector with a False Detection Control. IEEE Trans. Pattern Analysis Machine Intelligence **32**(4), 722–732 (2010)
14. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, 2 edn. (2003)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: Proc. CVPR. pp. 770–778 (2016)
16. Hold-Geoffroy, Y., Sunkavalli, K., Eisenmann, J., Fisher, M., Gambaretto, E., Hadap, S., Lalonde, J.F.: A Perceptual Measure for Deep Single Image Camera Calibration. In: Proc. CVPR. pp. 2354–2363 (2018)
17. Kluger, F., Brachmann, E., Ackermann, H., Rother, C., Yang, M.Y., Rosenhahn, B.: Consac: Robust multi-model fitting by conditional sample consensus (2020), https://arxiv.org/pdf/2001.02643.pdf
18. Košecká, J., Zhang, W.: Video Compass. In: Proc. ECCV. pp. 476–491 (2002)
19. Lee, H., Shechtman, E., Wang, J., Lee, S.: Automatic Upright Adjustment of Photographs with Robust Camera Calibration. IEEE Trans. Pattern Analysis Machine Intelligence **36**(5), 833–844 (2014)
20. Li, H., Zhao, J., Bazin, J.C., Chen, W., Liu, Z., Liu, Y.H.: Quasi-globally Optimal and Efficient Vanishing Point Estimation in Manhattan World. In: Proc. ICCV. pp. 1646–1654 (2019)
21. Ma, Y., Soatto, S., Kosecka, J., Sastry, S.S.: An Invitation to 3-D Vision: From Images to Geometric Models. Springer (2004)

22. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In: Proc. CVPR. pp. 652–660 (2017)
23. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. In: MICCAI. pp. 234–241 (2015)
24. Schindler, G., Dellaert, F.: Atlanta World: An Expectation Maximization Framework for Simultaneous Low-level Edge Grouping and Camera Calibration in Complex Man-made Environments. In: Proc. CVPR (2004)
25. Simon, G., Fond, A., Berger, M.O.: A-Contrario Horizon-First Vanishing Point Detection Using Second-Order Grouping Laws. In: Proc. ECCV. pp. 318–333 (2018)
26. Tardif, J.P.: Non-Iterative Approach for Fast and Accurate Vanishing Point Detection. In: Proc. ICCV. pp. 1250–1257 (2009)
27. Tretyak, E., Barinova, O., Kohli, P., Lempitsky, V.: Geometric Image Parsing in Man-Made Environments. International Journal of Computer Vision **97**(3), 305–321 (2012)
28. Wildenauer, H., Hanbury, A.: Robust Camera Self-Calibration from Monocular Images of Manhattan Worlds. In: Proc. CVPR. pp. 2831–2838 (2012)
29. Workman, S., Greenwell, C., Zhai, M., Baltenberger, R., Jacobs, N.: DeepFocal: A Method for Direct Focal Length Estimation. In: Proc. ICIP. pp. 1369–1373 (2015)
30. Workman, S., Zhai, M., Jacobs, N.: Horizon Lines in the Wild. In: Proc. BMVC. pp. 20.1–20.12 (2016)
31. Xian, W., Li, Z., Fisher, M., Eisenmann, J., Shechtman, E., Snavely, N.: UprightNet: Geometry-Aware Camera Orientation Estimation From Single Images. In: Proc. ICCV. pp. 9974–9983 (2019)
32. Xu, Y., Oh, S., Hoogs, A.: A Minimum Error Vanishing Point Detection Approach for Uncalibrated Monocular Images of Man-made Environments. In: Proc. CVPR. pp. 1376–1383 (2013)
33. Zeiler, M.D., Fergus, R.: Visualizing and Understanding Convolutional Networks. In: Proc. ECCV. pp. 818–833 (2014)
34. Zhai, M., Workman, S., Jacobs, N.: Detecting Vanishing Points using Global Image Context in a Non-Manhattan World. In: Proc. CVPR. pp. 5657–5665 (2016)
35. Zhou, Y., Qi, H., Huang, J., Ma, Y.: NeurVPS: Neural Vanishing Point Scanning via Conic Convolution. In: Proc. NeurIPS (2019)