CTRL-C: Camera calibration TRansformer with Line-Classification

Jinwoo Lee, Kakao Brain
Hyunsung Go, Kookmin University
Hyunjoon Lee, Kakao Brain
Sunghyun Cho, POSTECH
Minhyuk Sung, KAIST
Junho Kim[*], Kookmin University

Abstract

Single image camera calibration is the task of estimating the camera parameters from a single input image, such as the vanishing points, focal length, and horizon line. In this work, we propose Camera calibration TRansformer with Line-Classification (CTRL-C), an end-to-end neural network-based approach to single image camera calibration, which directly estimates the camera parameters from an image and a set of line segments. Our network adopts the transformer architecture to capture the global structure of an image with multi-modal inputs in an end-to-end manner. We also propose an auxiliary task of line classification to train the network to extract the global geometric information from lines effectively. Our experiments demonstrate that CTRL-C outperforms the previous state-of-the-art methods on the Google Street View and SUN360 benchmark datasets. Code is available at https://github.com/jwlee-vcl/CTRL-C.

1. Introduction

Single image camera calibration is a task of inferring intrinsic and extrinsic camera parameters by analyzing the distortion in an input image caused by the perspective projection. It is a key problem in various computer vision applications, including image rotation correction [11, 28], photo upright adjustment [21, 5], metrology [8, 42], visual aesthetics assessment [3, 28], object composition for augmented reality [15, 18, 19], and so on.

Single view geometry is highly related to projective geometric cues such as vanishing points (VPs) and the horizon line, which are the intersections of the world parallel lines and planes, respectively [13, 27]. Hence, for single image camera calibration, a classical approach is first to detect line segments in an input image and then find inlier line segments that account for VPs and the horizon line using RANSAC or other sampling strategies [29, 34, 21, 31]. However, this approach relies solely on lines and may degrade when inlier lines are falsely detected.

[*]Corresponding author: junho@kookmin.ac.kr

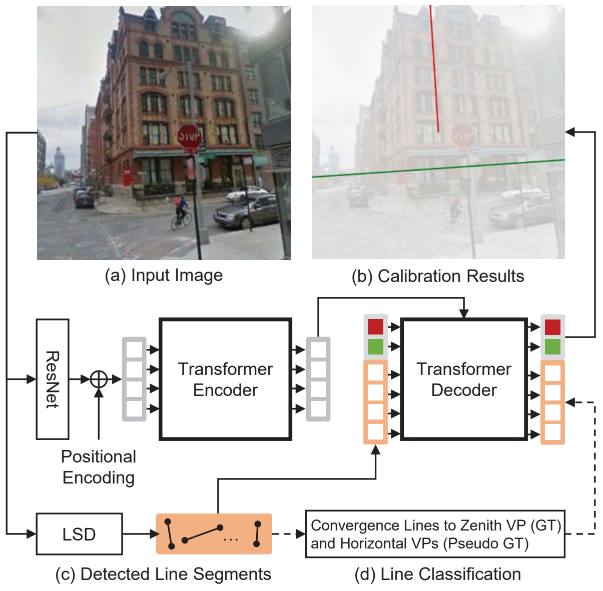


Figure 1. Overview of CTRL-C. From a given image (a), CTRL-C predicts camera parameters including the zenith, FoV, and horizon line (b) by taking multi-modal cues; semantic ones from (a) and geometric ones from detected line segments in (c). While directly regressing the camera parameters, CTRL-C classifies line segments into vertical and horizontal convergence lines, as in (d). This auxiliary line classification task assists the network to have a better understanding of the geometric structure in the image and thus helps improve the accuracy of camera parameter prediction.

Several deep learning-based approaches have recently been proposed to overcome such limitations [37, 16, 38]. These methods directly infer the camera parameters from an input image using semantic cues learned by deep neural networks. While they achieve more accurate calibration results than classical methods, their networks need to learn geometric structures from an image without any explicit supervision, limiting their performance. Recently, a few deep learning-based methods [41, 22] leveraging geometric and semantic cues have been proposed and achieved superior results. Nonetheless, their approaches to leveraging lines are still limited to conducting post-processing [41] or building multiple networks that are trained separately [22]. Moreover, all the existing neural network-based approaches rely on *convolutional* neural networks (CNNs), which are less effective in capturing long-term dependencies over an image, and consequently, global characteristics of an image such as the camera parameters.

To address the aforementioned issues, we propose to leverage *transformers* [35, 9], which have been recently adopted in multiple vision tasks [12, 20]. We observe that transformers are particularly suitable for the following goals related to single image camera calibrations: i) exploiting both geometric and semantic cues, and ii) effectively learning their relationships and global contextual information of an image. As transformers treat any types of input data as a sequence of tokens, both the semantic and geometric cues can be easily incorporated into a single end-to-end network. Also, thanks to the attention mechanism, transformers can readily capture long-term dependencies across local image contexts and line segments. Most importantly, we also empirically demonstrate that an *auxiliary* task using the outputs of the transformers can even improve the performances by facilitating the interactions between these cues.

To this end, we propose a novel neural network named *Camera calibration TRansformer with Line-Classification*, *CTRL-C* in short, whose pipeline is illustrated in Fig. 1. Our CTRL-C takes both an image and line segments as input and regresses the camera parameters based on the transformer encode-decoder architecture. The input image is first fed to a ResNet [14] and converted to a set of features of local patches. The transformer encoder then processes the image features with positional encoding to generate our *semantic tokens*. The line segments, extracted from the input image using the LSD algorithm [36], are also mapped to *geometric tokens*. The subsequent transformer decoder aggregates both semantic and geometric tokens along with the queries for the camera parameters — zenith VP, horizon line, and field of view (FoV) — and learns the relationships across them. As an auxiliary task, the line segments are classified into convergence lines to either the zenith or horizontal VPs, which affects to improve the performance of camera parameter regression.

Our experimental results on the Google Street View [22] and SUN360 [39] datasets show that CTRL-C outperforms previous single image camera calibration methods [31, 37, 16, 38, 22] in multiple evaluation criteria. Even without the line classification task, our baseline transformer architecture already achieves competitive performance compared to the previous state-of-the-art (SotA) methods. Adopting the line classification task improves the performance, reducing the error of the up direction, pitch, roll, and FoV angles. Especially, CTRL-C increases the AUCs of the horizon line estimation with significant margins, from 83.12% to 87.29% (4.17% gap) for the Google Street View test set and from 80.85% to 85.45% (4.6% gap) for the SUN360 test set, compared to the results of the previous SotA methods.

2. Related Work

Single Image Camera Calibration Conventional methods typically find two or more VPs and estimate the camera's focal length and rotation from the relationship across them. Schaffalitzky and Zisserman [29] propose an automatic detection algorithm of VPs by grouping of planar geometric patterns. Tretyak *et al.* [34] introduce an optimization framework that parses edge pixels into groups of the world parallel lines for different vanishing points and geometrically analyzes them. Lee *et al.* [21] present a maximum-a-posteriori (MAP)-based optimization method that estimates the VPs and a relative camera frame for a 3D scene, and then applies the camera calibration parameters for photo upright adjustment. Simon *et al.* [31] detect the zenith VP and the horizon by finding the maximally meaningful modes.

Contrarily, recent neural network-based approaches propose to directly predict the camera parameters using semantic cues from an input image learned by convolutional networks. Workman *et al.* [37] propose the first neural network-based solution for estimating the horizon line from an image. Hold-Geoffroy *et al.* [16] extend the idea to jointly predict both focal length and the horizon line, and provide perceptual studies of how humans perceive errors in camera calibration. Xian *et al.* [38] even step forward to predict the camera rotation as well, not by simply regressing the parameters, but by predicting the per-pixel 3D surface frames first and integrating the information. While these neural approaches show improvement in accuracy compared to the conventional methods, they do not take any explicit geometric information of the input image, leaving room for further improvement.

A few neural approaches consider both geometric and semantic information for single image camera calibration [41, 22]. Zhai *et al.* [41] obtain an initial horizon line using a CNN and improve it with a separate optimization step using line segments detected by the LSD algorithm [36]. Lee *et al.* [22] propose a neural network-based geometric parsing framework for single image camera calibration, which uses both input image and lines detected from the image with the LSD algorithm [36]. They demonstrate that exploiting geometric information such as lines can help a network better understand the underlying perspective structure in an image and thus improve the camera calibration performance. However, both of these methods are not end-to-end learnable. Zhai *et al.*'s method requires a separate optimization step while Lee *et al.*'s method relies on two separately trained networks, and a complicated network structure to integrate line and image information. Moreover, all the existing neural network-based approaches rely on CNNs, which are ineffective for capturing global information.

Inspired by [41, 22], we also utilize lines as additional cue. In contrast to [41, 22], however, we design CTRL-C

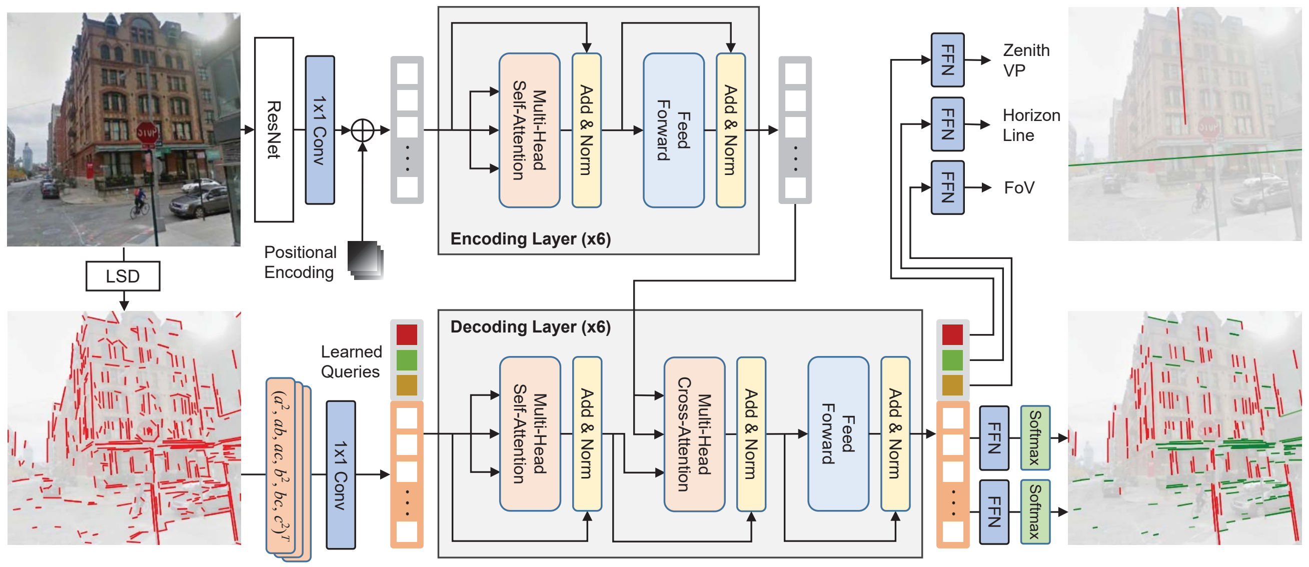


Figure 2. Our network estimates parameters for camera calibration from an input image and a set of line segments. Image features are extracted with the ResNet backbone, flattened and positionally encoded, and then passed into a transformer encoder network. Learned query embedding vectors are fed into a transformer decoder network, alongside with the line embedding vectors, so that queries can attend to image and line features. Two auxiliary outputs, vertical and horizontal convergence line scores, are used to further guide the network to learn the scene geometry for calibration.

based on the transformer architecture so that our network can be end-to-end learnable and effectively gather information from both semantic and geometric information. We also show that our proposed auxiliary task of line classification significantly improves the camera calibration accuracy.

Image Transformers With remarkable success in natural language processing, transformers have been recently adopted to solve various tasks in computer vision. We refer the readers to [20, 12] for thorough surveys. To name a few, Dosovitskiy *et al.* [10] demonstrate that a standard transformer encoder treating images as a series of patches can classify images well without the image-specific induced bias. Carion *et al.* [4] propose a transformer encoder-decoder architecture for object detection, called DETR, which predicts a set of object bounding boxes. Zhu *et al.* [43] improve DETR [4] with deformable attention modules both in accuracy and training time. Xu *et al.* [40] introduce a multi-scale transformer architecture for line segment detection, called LETR. VisualBERT [23], ViLBERT [25], and their variants [32, 26, 6] also apply transformers to solve joint vision-and-language reasoning problems. Along this direction, we also adopt the transformer in the single image calibration problem. Our main contribution is, however, in maximizing the advantage of the transformer architecture by providing image patches and line features as multimodal inputs. In contrast to [4, 40] that use transformers as a *direct regressor*, e.g., for box parameters (DETR [4]) and for line parameters (LETR [40]), we use the transformer to facilitate leveraging *multi-modal cues*: image patches for semantic cues and line segments for geometric cues.

3. Framework

Fig. 2 shows our framework. From a given input image, we estimate the vertical vanishing point (also known as the zenith VP), the horizon line, and the FoV for the focal length of the camera. Our network has the following components: a CNN backbone network that extracts image features; an encoder transformer that encodes image features; a decoder transformer followed by FFNs that predicts all the outputs.

3.1. Backbone Network

We use ResNet (`resnet50` in `torchvision`) to compute image features [14]. From a given input image of size $3 \times H_0 \times W_0$, we use the `block4` output of ResNet as image features of size $C \times H \times W$ where $H_0 = 32H$ and $W_0 = 32W$. Throughout our experiments we set H_0, W_0 and C as 512, 512 and 2048, respectively.

3.2. Encoder Network

Our encoder architecture is similar to several previous works [4, 10, 40]; image features F are first projected with a 1×1 convolution to have a smaller number of channels. The projected features of size $d \times H \times W$ are then spatially flattened and fed into the transformer encoder network. We set $d = 256$ in our experiments. The encoder network consists of six self-attention blocks, and each block has eight attention heads. Positional encodings are added to features for each self-attention block; refer [35, 4] for details of the architecture.

3.3. Decoder Network

To estimate the calibration parameters, we query all the parameters to the transformer decoder. Similarly to [4, 40], all the queries for the zenith VP, horizon line, and FoV are decoded in parallel; we feed three d-dimensional vectors to the decoder as query embeddings. Multi-headed self-attention and cross-attention blocks are applied for several times to transform query embeddings to estimations of the parameters.

Utilizing Line Segments Although it is already possible to achieve SotA results only with the transformers and encoded image features (see Sec. 4.3, Tables 1 and 2), we can further improve the estimation accuracy by utilizing line segments, especially for images with man-made structures.

Many previous approaches [33, 34, 21] utilize line segments for camera calibration, mostly by detecting the zenith and horizontal VPs and classifying the lines into vertical and horizontal convergences (or none of the two). Classification is often performed via energy minimization under some widely used assumptions [7, 30], and here we replace this optimization to a supervised classification problem. To the end, in addition to inferring calibration parameters, our network is trained to classify *convergence lines* to the zenith and horizontal VPs in a set of line segments.

We first detect a set of line segments from a given input image [36]. For each line segment, the corresponding line equation can be computed as a cross product of the two endpoints of the segment:

$$\mathbf{l} = [\mathbf{p}_0]_\times \, \mathbf{p}_1, \tag{1}$$

where $\mathbf{p}_0$, $\mathbf{p}_1$ and $\mathbf{l}$ represent two endpoints of a line segment and its line equation, respectively. We sample at most 512 line segments, $L = \{\mathbf{l}_0, \ldots, \mathbf{l}_{n<512}\}$, from the detected line segments. Directional ambiguities in line equations ($\mathbf{l}$ and $-\mathbf{l}$ represent the same line) are removed by taking upper triangular part of $\mathbf{l}\mathbf{l}^T$ such that $\mathbf{s} = (a^2, ab, ac, b^2, bc, c^2)^T$ where $\mathbf{l} = (a, b, c)^T$. We apply a 1×1 convolution to $\mathbf{s}$ to build d-dimensional line embedding vectors $\{\mathbf{e}_0, \ldots \mathbf{e}_n\}$.

Our network is then trained to classify input line segments into vertical and horizontal convergence lines (or none of the two) - lines that pass through the zenith VP and horizontal VPs, respectively. For this, we label each line segment $\mathbf{l}_i$ using the following equation:

$$c(\mathbf{l}_i, \mathbf{v}) = \begin{cases} 1 & \text{if } d(\mathbf{l}_i, \mathbf{v}) \le \delta_0 \\ 0 & \text{if } d(\mathbf{l}_i, \mathbf{v}) \ge \delta_1 \\ -1 & \text{otherwise} \end{cases}, \tag{2}$$

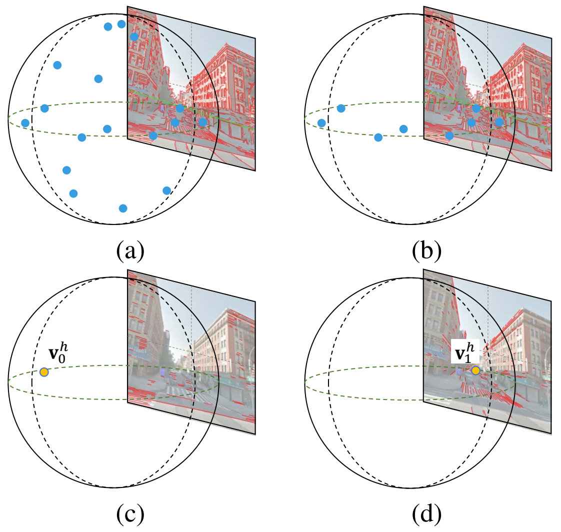


Figure 3. Pseudo horizontal VP estimation process; (a) computing VP candidates with the intersection of different line pairs in L, (b) filtering the VP candidates in nearby the GT horizon line, (c) selecting a VP candidate through which the most lines in L pass nearby, (d) after filtering out the lines considered in (c) from L, selecting another VP candidate through which the most lines pass nearby. Pseudo horizontal VPs are set as the selected VP candidates in (c) and (d).

where $\mathbf{v}$ represents the zenith VP or a horizontal VP. The point-line distance function $d(\cdot)$ is defined as:

$$d(\mathbf{l}, \mathbf{v}) = \left| \frac{\mathbf{v}^T \mathbf{l}}{\|\mathbf{l}\| \, \|\mathbf{v}\|} \right|, \tag{3}$$

where $\|\cdot\|$ represents the L_2 norm of a vector. Throughout our experiments, we set δ_0 and δ_1 as $\sin(2°)$ and $\sin(5°)$, respectively.

Estimating Pseudo Horizontal VPs Once we have ground-truth (GT) values for the zenith VP and horizontal VPs, we can assign labels for vertical and horizontal convergence lines using Eq. (2). However, the Google Street View and SUN360 datasets only provide GT values for the zenith VP but not for horizontal VPs. To mitigate this issue, we create pseudo horizontal VPs; 1) extract a set of VP candidates, 2) filter out non-horizontal VP candidates, and 3) choose two VPs from the set of remaining candidates (see Fig. 3).

To extract a set of initial VP candidates, we use a similar method with those of [21, 33]. From the set of line equations L, we randomly select a pair of line segments $(\mathbf{l}_0, \mathbf{l}_1)$ and compute their intersection points $\mathbf{v}$ as $\mathbf{v} = \mathbf{l}_0 \times \mathbf{l}_1$. We repeat this until we have the set of VP candidates V_c.

By definition, all horizontal VPs should be located on the horizon line. We remove the candidates from V_c that are not near enough to the horizon line and keep the remains as the horizontal VP candidates using the following equation:

$$V_{\text{in}} = \{\mathbf{v} \mid d(\mathbf{h}, \mathbf{v}) < \delta \text{ and } \mathbf{v} \in V_c\}, \tag{4}$$

where $\mathbf{h}$ represents the horizon line equation and $d(\cdot)$ is defined in Eq. (3).

From the horizontal VP candidates in V_{in}, we select two dominants, $\mathbf{v}_0^h$ and $\mathbf{v}_1^h$, as the pseudo horizontal VPs. For each candidate in V_{in}, we find line segments close enough to the candidate and compute the sum of their lengths:

$$m_i = \sum_{\mathbf{l} \in L_{\text{in}}^i} \text{len}(\mathbf{l}), \text{ where} \tag{5}$$

$$L_{\text{in}}^i = \{\mathbf{l} \mid d(\mathbf{l}, \mathbf{v}_i) < \delta \text{ and } \mathbf{v}_i \in V_{\text{in}}\} \tag{6}$$

where $\text{len}(\mathbf{l})$ represents the length of the line segment whose line equation is $\mathbf{l}$. We select a pseudo horizontal VP $\mathbf{v}_0^h$ with the largest value of m_i, remove L_{in}^i from L, and find $\mathbf{v}_1^h$ by repeating the above process. We set δ in Eqs. (4) and (6) as $\sin(2.5^\circ)$ in our experiments.

The decoder network gets line embeddings as well as calibration queries as inputs ($n + 3$ queries in total), transforms them, and feeds the results to the feed forward networks. Position embeddings are not added to line features, as lines contain positional information by construction.

3.4. Feed-Forward Networks (FFNs)

We append separate FFNs to estimate calibration parameters $\mathbf{z}$, $\mathbf{h}$, and f, representing zenith VP, horizon line, and FoV, respectively. To classify line embeddings into vertical and horizontal convergence lines, two additional FFNs are appended to line embeddings $\{\mathbf{e}_0, \ldots, \mathbf{e}_{n-1}\}$, followed by softmax layers resulting $\{s_0^z, \ldots, s_{n-1}^z\}$ and $\{s_0^h, \ldots, s_{n-1}^h\}$, respectively, where $s^{\{z,h\}} \in [0, 1]$.

3.5. Loss Functions

Our network is trained with five loss terms:

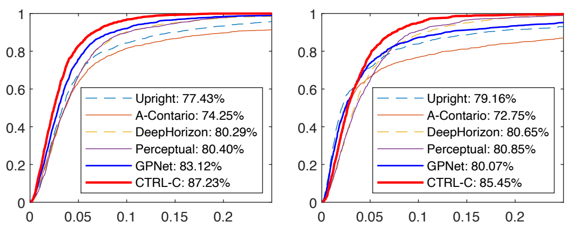
$$l = l_z + l_h + l_f + l_{zc} + l_{hc}. \tag{7}$$

For the zenith VP, we minimize angular distances between the GT points and their corresponding predictions:

$$l_z = 1 - \left| \frac{\mathbf{z}^T \tilde{\mathbf{z}}}{\|\mathbf{z}\| \|\tilde{\mathbf{z}}\|} \right|, \tag{8}$$

where $\mathbf{z}$ and $\tilde{\mathbf{z}}$ represent a GT zenith VP and an estimate, respectively. For the horizon line, we use a metric from [2]. Specifically, we compute intersection points between image boundaries and GT/estimated horizon lines. Let $\mathbf{b}_l$ and $\tilde{\mathbf{b}}_l$ be intersection points of the left image boundary and horizon lines $\mathbf{h}$ and $\tilde{\mathbf{h}}$, respectively, and $\mathbf{b}_r$ and $\tilde{\mathbf{b}}_r$ for the right image boundary. The loss function is defined as:

$$l_h = \max\left(\left\| \tilde{\mathbf{b}_l} - \mathbf{b}_l \right\|_1, \left\| \tilde{\mathbf{b}_r} - \mathbf{b}_r \right\|_1 \right), \tag{9}$$



(a) (b)

Figure 4. Comparison of the cumulative distributions of the horizon line error and their AUCs tested on Google Street View (a) and SUN360 (b). The AUCs are also reported in Tables 1 and 2.

where $\|\cdot\|_1$ represents the L_1 norm of a vector. The loss function for the FoV parameter is defined as:

$$l_f = \left| f - \tilde{f} \right|. \tag{10}$$

As the classification losses of the vertical and horizontal convergence lines, l_{zc} and l_{hc}, binary cross entropy loss is used:

$$l_{\{zc,hc\}} = -\frac{1}{n} \sum_i \left\{ c_i^{\{z,h\}} \log s_i^{\{z,h\}} + \left(1 - c_i^{\{z,h\}}\right) \log \left(1 - s_i^{\{z,h\}}\right) \right\}, \tag{11}$$

where c_i^z and c_i^h are defined using Eq. (2) as follows:

$$c_i^z = c(\mathbf{l}_i, \mathbf{z}) \tag{12}$$

$$c_i^h = \max\left\{ c(\mathbf{l}_i, \mathbf{v}_0^h), c(\mathbf{l}_i, \mathbf{v}_1^h) \right\}. \tag{13}$$

We remove any c_i from computing losses if $c(\mathbf{l}_i, \cdot)$ is -1.

4. Experiments

In the experiments, we evaluate the performance of our CTRL-C and other baselines using the benchmarks generated based on Google Street View (GSV) [1] and SUN360 [39] datasets. The benchmarks curated by Lee *et al.* [22] are constructed by rectifying the original panoramic images with random samples of FoV, pitch, and roll in the ranges of $40 \sim 80^\circ$ ($40 \sim 90^\circ$ for SUN360), $-30 \sim 40^\circ$, and $-20 \sim 20^\circ$, respectively. Each sampled image's zenith VP and horizon line are then computed from sampled FoV, pitch and roll. The training and test sets contain 13,214 and 1,333 images for the GSV benchmark and 30,837 and 878 images for the SUN360 benchmark. Refer to the supplementary material for more experimental and qualitative results on other datasets.

4.1. Comparison

We compare our method with the following previous methods: Upright [21], A-Contrario detection [31], DeepHorizon [37], Perceptual measure [16], UprightNet [38],

Table 1. Quantitative evaluation results with Google Street View benchmark from Lee *et al.* [22]. Note that the results of previous methods are from Lee *et al.* [22] — see Table 2 in their paper. The accuracy of FoV prediction is not provided for A-Contrario detection [31], DeepHorizon [37], and UprightNet [38], since they cannot predict FoV. *TR* indicates the case of ablating transformers but just using ResNet [14] to directly predict the camera parameters.

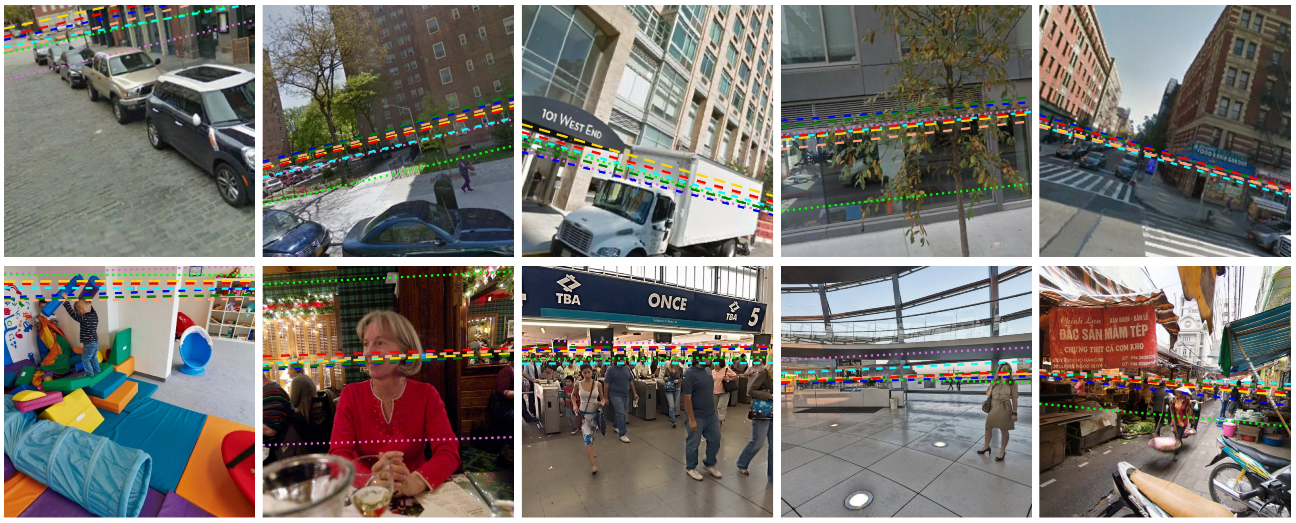
Method	Up Direction (°) ↓		Pitch (°) ↓		Roll (°) ↓		FoV (°) ↓		AUC
	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	(%) ↑
Upright [21]	3.05	1.92	2.90	1.80	6.19	0.43	9.47	4.42	77.43
A-Contrario [31]	3.93	1.85	3.51	1.64	13.98	0.52	-	-	74.25
DeepHorizon [37]	3.58	3.01	2.76	2.12	1.78	1.67	-	-	80.29
Perceptual [16]	2.73	2.13	2.39	1.78	0.96	0.66	4.61	3.89	80.40
UprightNet [38]	28.20	26.10	26.56	24.56	6.22	4.33	-	-	-
GPNet [22]	2.12	1.61	1.92	1.38	0.75	**0.47**	6.01	3.72	83.12
CTRL-C (Ours)	**1.80**	**1.52**	**1.58**	**1.31**	**0.66**	0.53	**3.59**	**2.72**	**87.29**

Id	TR	l_{vc}	l_{hc}	Ablation Study (TR: Transformers)								
1				3.12	2.62	2.79	2.22	1.04	0.82	5.04	4.17	84.48
2	✓			2.17	1.84	1.87	1.51	0.82	0.60	3.71	2.88	85.65
3	✓	✓		1.84	1.53	1.61	1.32	0.65	**0.52**	**3.47**	**2.66**	87.16
4	✓		✓	2.05	1.75	1.75	1.43	0.83	0.63	3.83	3.00	86.09
5	✓	✓	✓	**1.80**	**1.52**	**1.58**	**1.31**	**0.66**	0.53	3.59	2.72	**87.29**

Table 2. Quantitative evaluation results with SUN360 benchmark from Lee *et al.* [22]. Note that the results of previous methods from Lee *et al.* [22] — see Table 2 in their paper. The accuracy of FoV prediction is not provided for A-Contrario detection [31], DeepHorizon [37], and UprightNet [38], since they cannot predict FoV. *TR* indicates the case of ablating transformers but just using ResNet [14] to directly predict the camera parameters.

Method	Up Direction (°) ↓		Pitch (°) ↓		Roll (°) ↓		FoV (°) ↓		AUC
	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	(%) ↑
Upright [21]	3.43	1.43	3.03	1.13	6.85	0.47	8.62	3.21	79.16
A-Contrario [31]	5.77	1.53	4.91	1.19	6.93	0.66	-	-	72.75
DeepHorizon [37]	2.87	2.12	2.36	1.64	1.16	0.85	-	-	80.65
Perceptual [16]	2.54	1.93	2.11	1.49	1.06	0.77	5.29	3.93	80.85
UprightNet [38]	34.72	34.67	35.31	33.72	4.92	2.88	-	-	-
GPNet [22]	2.33	**1.27**	1.97	**0.96**	0.97	**0.51**	5.66	3.16	80.07
CTRL-C (Ours)	**1.91**	1.57	**1.50**	1.15	**0.96**	0.71	**3.80**	**2.77**	**85.45**

Id	TR	l_{vc}	l_{hc}	Ablation Study (TR: Transformers)								
1				2.34	1.83	1.90	1.41	1.09	0.80	4.60	3.55	83.94
2	✓			2.03	1.71	1.62	1.26	1.00	0.75	4.00	2.98	84.39
3	✓	✓		**1.80**	**1.45**	**1.44**	**1.05**	**0.88**	**0.63**	**3.76**	**2.68**	85.09
4	✓		✓	2.03	1.65	1.59	1.20	1.03	0.78	3.87	2.84	85.36
5	✓	✓	✓	1.91	1.57	1.50	1.15	0.96	0.71	3.80	2.77	**85.45**



🟨🟨🟨🟨 Ground Truth 🟪🟪🟪🟪 Upright [21] 🟩🟩🟩🟩 A-Contario [31] 🟦🟦🟦🟦 DeepHorizon [37]
🟩🟩🟩🟩 Perceptual [16] 🟦🟦🟦🟦 GPNet [22] 🟧🟧🟧🟧 ResNet 🟥🟥🟥🟥 CTRL-C (Ours)

Figure 5. Examples of horizon line prediction on the Google Street View test set (top row) and the SUN360 test set (bottom row).

Table 3. Classification accuracies of convergence lines.

Training	Test	Accuracy (%) ↑ Vertical	Accuracy (%) ↑ Horizontal
GSV	GSV	99.73	93.35
	SUN360	93.76	85.18
SUN360	GSV	97.96	92.16
	SUN360	99.49	91.54

and GPNet [22]. For the comparison, we refer to the results reported by GPNet [22], where all the networks are retrained with the GSV dataset with the same ResNet architecture as the backbone (except for the UprightNet whose code is not publicly available). Note that A-Contrario detection [31], DeepHorizon [37], and UprightNet [38] cannot predict FoV, and thus their results for FoV are not provided.

Tables 1 and 2 show the quantitative results with the GSV [1] dataset and the SUN360 [39] dataset, respectively. We report the angle differences of up direction, pitch, roll, and FoV. For the horizon line, as done by Barinova *et al.* [2], we calculate the area under the curve (AUC) of the cumulative distribution graph as shown in Fig. 4, in which the x-axis indicates the distance between the predicted and GT horizon lines, and the y-axis indicates the percentage. Note that our CTRL-C outperforms all the baseline methods with significant margins in all the evaluation criteria, both in the GSV and SUN360 benchmarks. As shown in Table 1, for the GSV benchmark, our CTRL-C reduces the mean errors of the up direction, pitch, roll, and FoV angles by 15.1%, 17.7%, 12.0%, and 40.3%, respectively, from the results of the previous SotA, GPNet [22]. The AUC of the horizon line errors is also improved from 83.12% to 87.29%, which has a 4.17% gap. The results with the SUN360 benchmark in Table 2 also show a similar trend. Compared to the results of Perceptual [16], the AUC of the horizon line errors is improved from 80.85% to 85.45%, which has a 4.6% gap. Fig. 5 shows qualitative comparisons.

4.2. Line Classification Results

While the convergence line classification is an auxiliary task of our network, we also report the accuracy of the classification in Table 3. When measuring the accuracy, the GT labels of convergence lines are given by the angle thresholds shown in Eq. (2). The results show very high accuracies and also good generalization capabilities even when the training and test datasets are different. For examples, training with GSV and testing with SUN360, the classification accuracies for vertical and horizontal convergence lines are 93.76% and 85.18%, respectively. In training with SUN360 and testing with GSV, the classification accuracies of vertical and horizontal convergence lines are better generalized to 97.96% and 92.16%, respectively.

Fig. 6 shows the qualitative results of the convergence line classification, along with the results of the horizon line and vertical direction prediction. Fig. 6 (a) and (c) show the input image and the line segments detected by LSD [36], which are fed to our network. Fig. 6 (b) shows the output vertical direction toward the zenith VP (red) and the horizon line (green). Fig. 6 (d) illustrates the classification results of vertical and horizontal convergence lines performed as an auxiliary task; red and green indicate line segments classified as vertical and horizontal convergence lines (with a threshold 0.5), respectively. Notice that the classified line segments in (d) are vanishing toward the zenith VP or one of the two horizontal VPs laying on the horizon line in (b).

4.3. Ablation Study

We further demonstrate the impact of each component in our network by conducting an ablation study. The quantitative evaluation results are reported at the bottom of Tables 1 and 2. We first ablate the transformers [35] and just use ResNet [14] as an image encoder that directly outputs the camera parameters. The network is trained with the same losses for the camera parameters (l_z, l_h, and l_f in Sec. 3.5). As shown in the first row at the bottom of each table, the performance is comparable but slightly worse than that of GPNet [22] in some criteria, including the up direction, pitch, and roll. When the transformers are employed to process the patch features coming from ResNet (the second row), the performance becomes similar or even better than GPNet. Further improvements are made when feeding lines as additional inputs to the transformer decoders and adding each classification loss for vertical and horizontal convergence lines (l_{vc} or l_{hc}, the third and fourth rows). Particularly, we found that the classifier for vertical convergence lines makes meaningful improvements in the prediction of all camera parameters. The best performance can be achieved when the both classifiers for vertical and horizontal convergence lines are used (the fifth row).

4.4. Evaluation of Intermediate Decoding Layers

The decoder of the transformer architecture is designed to repeat a layer including self-attention and cross-attention blocks — typically six layers are used [4, 40]. Since the inputs and outputs of the decoding layer are the same in the recursive structure, the camera parameters can also be predicted from *each* layer by appending the FFN block used in the last layer. The network is then trained to predict calibration parameters from outputs of each decoder layer. Table 4 illustrates the quantitative evaluation results for the camera parameters predicted from each decoding layer. The results show that prediction power is increased as the image patch and line tokens are passed through more decoding layers, but the performance gain also converges as more layers are used.

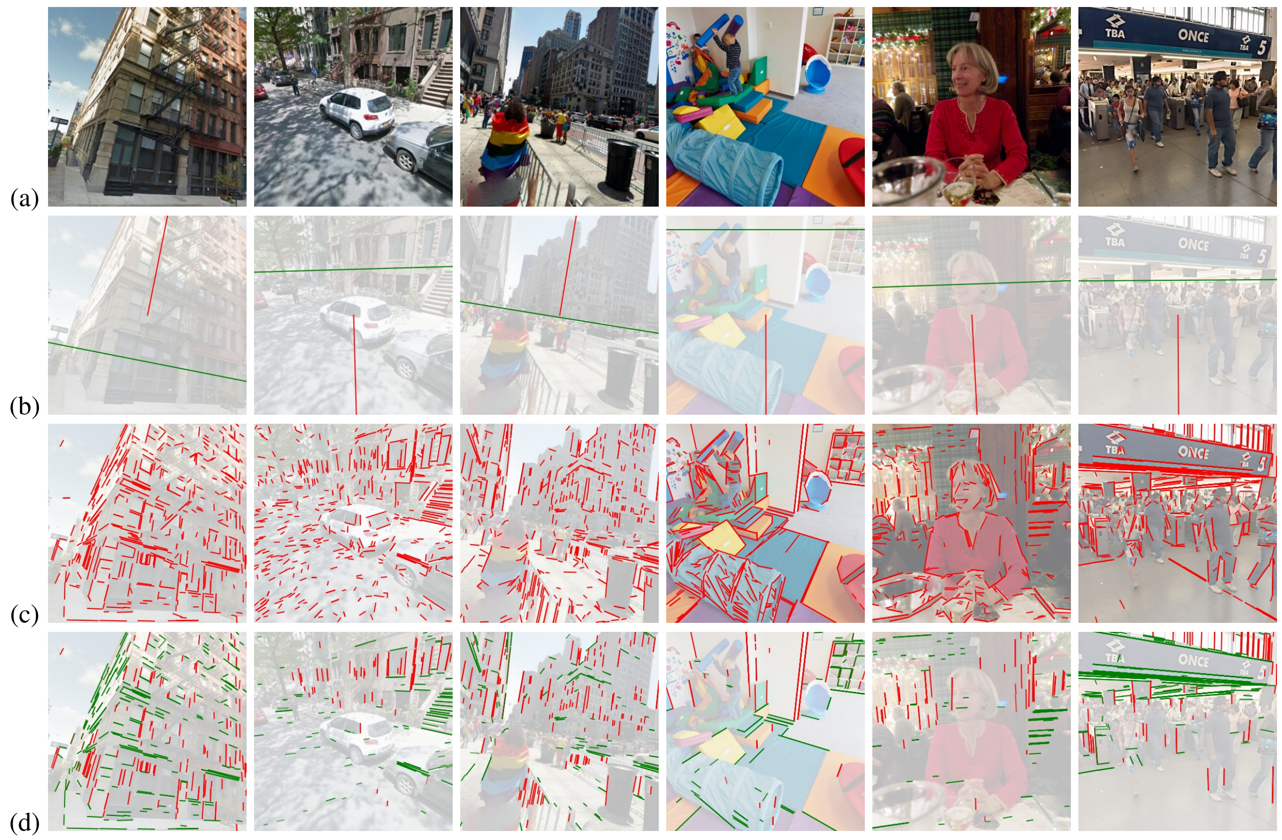


Figure 6. More results with the Google Street View [22] test set (the first three columns) and SUN360 [39] test set (the last three columns): (a) input image, (b) estimated horizon line (green) and vertical direction along with the zenith VP (red), (c) detected lines with LSD [36], (d) estimated vertical (red) and horizontal (green) convergence line segments of (c).

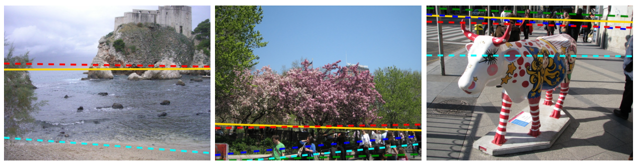


Figure 7. Examples of horizon prediction on natural scenes in the HLW [37] test set. Refer to Fig. 5 for the horizon line color coding.

4.5. Results on Natural Scenes

Since we utilize semantic information learned from the ImageNet pre-trained ResNet, our network performs well even in natural scenes with few line segments, as some results with the HLW [37] dataset are shown in Fig. 7. For the quantitative analyses with the HLW [37] dataset, refer to the supplementary material.

5. Conclusion

We presented CTRL-C, a novel method for single image camera calibration based on transformer encoder-decoder architecture using multi-modal tokens from the input image. The approach achieves the state-of-the-art performances, by capturing long-term dependencies among image tokens and geometric tokens with the self-/cross-attention modules. For future work, we investigate to improve the generalization capability of the proposed network using the self-supervised based pre-training stage training [17, 24].

Table 4. Quantitative evaluation of the different numbers of layers in the transformer decoder ('Id' indicates the index of decoding layers). More decoding layers achieve better performance.

Id	Up Dir. (°) ↓		Pitch (°) ↓		Roll (°) ↓		FoV (°) ↓		AUC
	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	(%) ↑
1	2.34	1.99	1.95	1.57	1.02	0.78	3.83	3.00	86.46
2	2.14	1.84	1.86	1.53	0.81	0.65	3.70	2.76	87.20
3	1.98	1.70	1.72	1.42	0.74	0.58	3.67	2.84	87.22
4	1.85	1.59	1.61	1.35	0.69	0.55	3.68	2.78	87.21
5	1.82	1.53	1.60	1.32	**0.66**	**0.52**	3.66	2.76	**87.29**
6	**1.80**	**1.52**	**1.58**	**1.31**	**0.66**	0.53	**3.59**	**2.72**	**87.29**

Acknowledgements. This was was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2017R1D1A1B03034907, 2021R1F1A1045604, 2020R1C1C1014863) and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2020-0-01826, Problem-Based Learning Program for Researchers to Proactively Solve Practical AI Problems (Kookmin University) and No.2019-0-01906, Artificial Intelligence Graduate School Program (POSTECH)).

References

[1] Google Street View Images API. `https://developers.google.com/maps/`.

[2] Olga Barinova, Victor Lempitsky, Elena Tretiak, and Pushmeet Kohli. Geometric Image Parsing in Man-Made Environments. In *Proc. ECCV*, pages 57–70, 2010.

[3] Subhabrata Bhattacharya, Rahul Sukthankar, and Mubarak Shah. A Framework for Photo-Quality Assessment and Enhancement based on Visual Aesthetics. In *Proc. ACM Multimedia*, pages 271–280, 2010.

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers. In *Proc. ECCV*, pages 213–229, 2020.

[5] Krishnendu Chaudhury, Stephen DiVerdi, and Sergey Ioffe. Auto-rectification of user photos. In *Proc. ICIP*, pages 3479–3483, 2014.

[6] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. UNITER: UNiversal Image-TExt Representation Learning. In *Proc. ECCV*, pages 104–120, 2020.

[7] James M. Coughlan and Alan L. Yuille. The Manhattan World Assumption: Regularities in scene statistics which enable Bayesian inference. In *Proc. NIPS*, pages 845–851, 2000.

[8] Antonio Criminisi, Ian Reid, and Andrew Zisserman. Single View Metrology. *International Journal of Computer Vision*, 40(2):123–148, 2000.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. NAACL-HLT*, pages 4171–4186, 2019.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proc. ICLR*, 2021.

[11] Philipp Fischer, Alexey Dosovitskiy, and Thomas Brox. Image Orientation Estimation with Convolutional Networks. In *Proc. GCPR*, pages 368–378, 2015.

[12] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, Zhaohui Yang, Yiman Zhang, and Dacheng Tao. A Survey on Visual Transformer. *arXiv*, 2020.

[13] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2004.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proc. CVPR*, pages 770–778, 2016.

[15] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Putting Objects in Perspective. *International Journal of Computer Vision*, 80:3–15, 2008.

[16] Yannick Hold-Geoffroy, Kalyan Sunkavalli, Jonathan Eisenmann, Matthew Fisher, Emiliano Gambaretto, Sunil Hadap, and Jean-François Lalonde. A Perceptual Measure for Deep Single Image Camera Calibration. In *Proc. CVPR*, pages 2354–2363, 2018.

[17] Longlong Jing and Yingli Tian. Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey. *arXiv*, 2020.

[18] Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. Rendering Synthetic Objects into Legacy Photographs. *ACM Trans. Graphics (Proc. SIGGRAPH Asia 2011)*, 30(6):Article 157, 2011.

[19] Kevin Karsch, Kalyan Sunkavalli, Sunil Hadap, Nathan Carr, Hailin Jin, Rafael Fonte, Michael Sittig, and David Forsyth. Automatic Scene Inference for 3D Object Compositing. *ACM Trans. Graphics*, 33(3):Article 32, 2014.

[20] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in Vision: A Survey. *arXiv*, 2021.

[21] Hyunjoon Lee, Eli Shechtman, Jue Wang, and Seungyong Lee. Automatic Upright Adjustment of Photographs with Robust Camera Calibration. *IEEE Trans. Pattern Analysis Machine Intelligence*, 36(5):833–844, 2014.

[22] Jinwoo Lee, Minhyuk Sung, Hyunjoon Lee, and Junho Kim. Neural Geometric Parser for Single Image Camera Calibration. In *Proc. ECCV*, pages 541–557, 2020.

[23] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. VisualBERT: A Simple and Performant Baseline for Vision and Language. *arXiv*, 2019.

[24] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised Learning: Generative or Contrastive. *arXiv*, 2020.

[25] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. In *Proc. NeurIPS*, 2019.

[26] Jiasen Lu, Vedanuj Goswami, Marcus Rohrbach, Devi Parikh, and Stefan Lee. 12-in-1: Multi-Task Vision and Language Representation Learning. In *Proc. CVPR*, pages 10437–10446, 2020.

[27] Yi Ma, Stefano Soatto, Jana Košecká, and S. Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer, 2004.

[28] Armin Samii, Radomír Měch, and Zhe Lin. Data-Driven Automatic Cropping Using Semantic Composition Search. *Computer Graphics Forum*, 34(1):141–151, 2015.

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

[30] 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 *Proc. CVPR*, 2004.

[31] Gilles Simon, Antoine Fond, and Marie-Odile Berger. A-Contrario Horizon-First Vanishing Point Detection Using Second-Order Grouping Laws. In *Proc. ECCV*, pages 318–333, 2018.

[32] Hao Tan and Mohit Bansal. LXMERT: Learning Cross-Modality Encoder Representations from Transformers. In *Proc. EMNLP-IJCNLP*, pages 5100–5111, 2019.

[33] Jean-Philippe Tardif. Non-Iterative Approach for Fast and Accurate Vanishing Point Detection. In *Proc. ICCV*, pages 1250–1257, 2009.

[34] Elena Tretyak, Olga Barinova, Pushmeet Kohli, and Victor Lempitsky. Geometric Image Parsing in Man-Made Environments. *International Journal of Computer Vision*, 97:305–321, 2012.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Proc. NIPS*, pages 6000–6010, 2017.

[36] Rafael Grompone von Gioi, Jérémie Jakubowicz, Jean-Michel Morel, and Gregory Randall. LSD: A Fast Line Segment Detector with a False Detection Control. *IEEE Trans. Pattern Analysis Machine Intelligence*, 32(4):722–732, 2010.

[37] Scott Workman, Menghua Zhai, and Nathan Jacobs. Horizon Lines in the Wild. In *Proc. BMVC*, pages 20.1–20.12, 2016.

[38] Wenqi Xian, Zhengqi Li, Matthew Fisher, Jonathan Eisenmann, Eli Shechtman, and Noah Snavely. UprightNet: Geometry-Aware Camera Orientation Estimation From Single Images. In *Proc. ICCV*, pages 9974–9983, 2019.

[39] Jianxiong Xiao, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Recognizing Scene Viewpoint using Panoramic Place Representation. In *Proc. CVPR*, pages 2695–2702, 2012.

[40] Yifan Xu, Weijian Xu, David Cheung, and Zhuowen Tu. Line Segment Detection Using Transformers without Edges. In *Proc. CVPR*, pages 4257–4266, 2021.

[41] Menghua Zhai, Scott Workman, and Nathan Jacobs. Detecting Vanishing Points using Global Image Context in a Non-Manhattan World. In *Proc. CVPR*, pages 5657–5665, 2016.

[42] Rui Zhu, Xingyi Yang, Yannick Hold-Geoffroy, Federico Perazzi, Jonathan Eisenmann, Kalyan Sunkavalli, and Manmohan Chandraker. Single View Metrology in the Wild. In *Proc. ECCV*, pages 316–333, 2020.

[43] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable Transformers for End-to-End Object Detection. In *Proc. ICLR*, 2021.