MSCC: Multi-Scale Transformers for Camera Calibration

Xu Song[1] Hao Kang[1] Atsunori Moteki [2] Genta Suzuki[2] Yoshie Kobayashi[2] Zhiming Tan[1*]
[1]Fujitsu R&D Center Co., Ltd. {songxu, kanghao, zhmtan}@fujitsu.com
[2]Fujitsu Limited {moteki.atsunori, suzuki.genta, k.yoshie}@fujitsu.com

Abstract

Camera calibration is very important for some vision tasks, like rendering 3D scenes, environment reconstruction, and self-localization, etc. In this paper, we propose a framework of multi-scale transformers for camera calibration. With the input of a single image, the multi-scale features output from the model's backbone are utilized to estimate camera parameters. At the same time, we show that the way of coarse-to-fine is effective to locate global structures and detailed features in the image, by studying the attention response of horizon line estimation. Moreover, deep supervision is proven to get more precise results and accelerated training. Our method outperforms all the state-of-the-art methods by objective and subjective experiments on Google Street View dataset and Pano360.

1. Introduction

Camera calibration, mainly aiming to estimate extrinsic and intrinsic parameters of the camera, plays an important role in rendering 3D scenes, environment reconstruction, image rectification, and camera self-localization [9,27]. Intrinsic parameters mainly include focal length, principal point and lens distortion coefficients. Extrinsic parameters include pitch, roll, yaw, and translation vector, etc. In many cases, estimating horizon line and vanishing points is also one of the main tasks in camera calibration.

There are many existing methods for camera calibration. The checkerboard-based methods [45,46] are stable, but it's inconvenient for industrial applications because manual calibration is required in advance. Other traditional methods depend on geometric structures, like Vanishing Points (VPs) and horizon line (hl). Vanishing points are the projection of intersection of the parallel lines [27]. Thus, the vanishing point based methods need to find the cues which rely on the lines in images. The horizon line is a set of horizontal VPs (hVPs) and is orthogonal to zenith (zenith vanishing point, zvp) line [34]. Such horizon line based method is also constrained by the contents of the images. Thus, the performance of these methods may decrease if the contents and textures of the images are not sufficient for detecting lines.

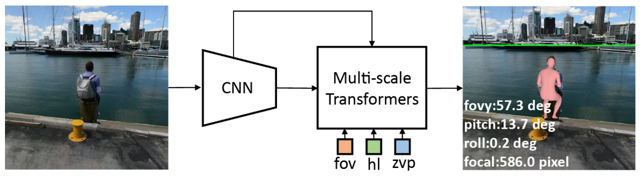

Figure 1. The pipeline of our proposed method, *fov*: field of view; *hl*: horizon line; *zvp*: zenith vanishing point.

With the advancement of deep learning, some methods directly estimate the camera parameters using supervised learning [2, 13, 39, 40]. Workman et al. [39, 40] directly estimated the focal length and the horizon lines from raw pixels using a deep neural network. Bogdan et al. [2] presented the deep learning based method for automatic intrinsic calibration (including focal length and distortion parameter). Hold-Geoffroy et al. [13] proposed a CNN-based model which can estimate the intrinsic and extrinsic parameters. In [20, 21], the authors also utilize geometric cues for camera calibration to achieve better results. Basically, these methods are based on Convolutional Neural Networks (CNNs).

Due to successful applications of transformers in computer vision, CTRL-C [20] debuted as the new State-Of-The-Art (SOTA) results. However, in [20], the authors didn't leverage different scales of the features fully. Inspired by [20, 43], we exploit the multi-scale encoder-decoder within the transformers for single image camera auto-calibration. Fig. 1 shows the pipeline of our proposed method. From a single image, we can estimate the field of view (fov), hl, and zvp concurrently using a CNN backbone coupled with multi-scale transformers. Specifically, we extract features from two blocks of the CNN module and input them to the multi-scale transformers for accurate parameter estimation. As an application example, the estimated 3D human bodies are projected onto the 2D image using the camera parameters, based on the results of calibration. Overall, the proposed method presents a valuable and comprehensive approach to camera parameter estimation in the field of camera calibration, by utilizing multi-scale features combined with the transformer architecture.

Our main contributions are as follows:

1. We utilize the multi-scale image features of the backbone to constitute the multi-stage encoder-decoder structure and apply this architecture to the camera calibration task.

2. With the effective combination of the coarse-to-fine process and the deeply-supervised training strategy, our model achieves more effective and stable results.

3. We surpass the SOTA results in terms of objective and subjective comparison on three datasets: Google Street View (GSV) [20], SYN-Citypark [18], and Flickr [18]. On another dataset of Horizon Lines in-the-Wild (HLW) [40], our model also exhibits good generalization capability.

Other parts are structured as follows. In Sec. 2, we briefly introduce the related work. In Sec. 3, the proposed method is explained in detail. In Sec. 4, the experimental results are analyzed and shown intuitively. Sec. 5 shows application results. Finally, Sec. 6 concludes our paper. It's better to view this paper in color mode.

2. Related work

2.1. Camera Calibration

Camera calibration is a technique to reconstruct the camera space by calculating camera parameters, which establish the mapping relationship between the world coordinate system and the image coordinate system. Some works use image content [20, 28], such as lines and vanishing points, to assist camera calibration. These methods are effective for scenes that conform to Manhattan [5] or Atlanta [32] assumptions. However, they are limited in challenging scenes, such as at night or in wild environments, where it is difficult to determine vanishing points or extract lines. To address this problem, some other methods [18] use panoramic images to generate a labeled dataset, which can improve the performance of models in predicting camera parameters for in-the-wild images. In addition, there are also methods using deep convolutional networks to evaluate the focal of wild images [39]. To improve the accuracy and robustness of camera calibration for different scenes, in this paper, we train our model in a manner that is tailored to different scene types. For scenes that conform to the Manhattan assumption, we use image lines to assist in estimating camera parameters. In the wild, we generate a high-quality and diverse labeled training dataset from panoramic images, and then estimate the camera parameters using a transformer-based network.

2.2. Multi-scale features

Over time, multi-scale backbones have been proposed, such as GoogLeNet [36], ResNet [10], DenseNet [15], U-Net++ [47], Deep Layer Aggregation (DLA) [44], and Res2Net strategy [7]. Additionally, multi-scale features also have been adopted in many different vision tasks [11, 16, 23, 31, 33, 35, 47]. FPN [23] exploited feature pyramid network, a top-down architecture to build feature maps at all scales, for object detection. Song et al. [35] utilized the multi-scale features in the encoder network to get sufficient features to achieve better fusion results. Huynh et al. [16] presented a multi-scale framework that resolves local ambiguity by looking at the image at multiple magnification levels.

Features at different scales can represent information of the image at different levels. High-level features can help the model to capture the global information because of the bigger receptive fields. In camera calibration tasks, we think the global information represents the image structure, such as the relationship of ground-buildings-sky in outdoor scenes, and structures of ground-furniture-walls-ceiling in indoor scenes. At the same time, low-level features use more image details to improve the model's calibration capabilities, because the camera parameters, the horizon line, and the vanishing points are influenced by image textures. For example, we think that the detailed lines and textures of the buildings, ground, and furniture are very helpful to find the horizon line and the vanishing points.

2.3. Coarse-to-fine

Coarse-to-fine is a way that combines features across multiple levels in a deep learning model. It usually processes the high-level (low-resolution) features of the image, and then utilizes the low-level (increased-resolution) features and propagates the final results.

There are many methods following coarse-to-fine strategy to process information across multiple levels [4, 12, 14, 22, 26, 29, 30, 43]. In [14], the authors proposed a recombinator network that aggregates information by feeding coarser branches into finer branches, allowing the finer resolutions to learn upon the features of coarser branches. Cho et al. [4] presented a novel coarse-to-fine manner, a single U-Net that enables more stable and effective deblurring.

In our work, we think that the coarse branch can help to find the important part of the image structure as the first step, and then the fine branch can locate the detailed attention in that found part as the second step. Accordingly, we can find the horizon line and vanishing points, and even estimate camera parameters in a more precise way.

2.4. Transformer

In recent years, the transformer has been successfully applied to computer vision tasks. Dosovitskiy et al. [6]

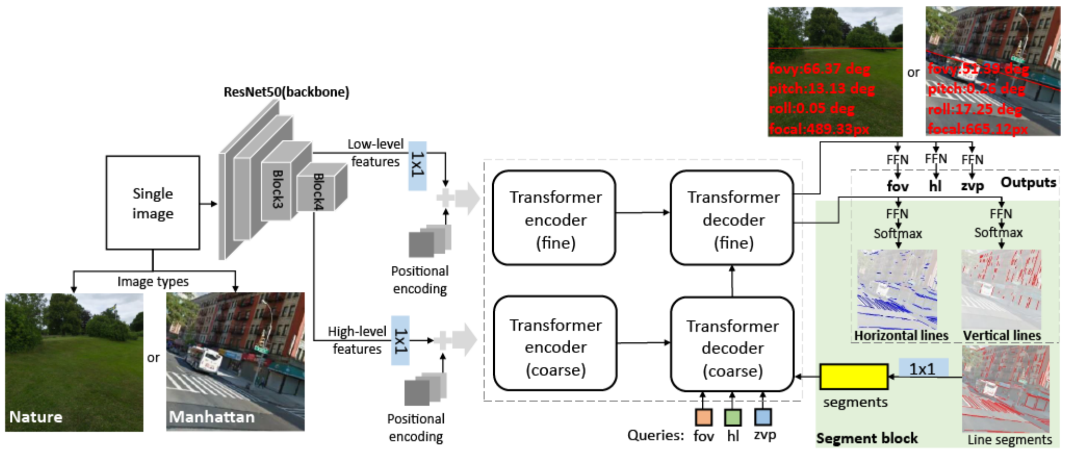

Figure 2. MSCC framework overview. The input of the MSCC is a single image. Segment block: if the input image complies with Manhattan world assumptions, the line segments queries are utilized to classify the horizontal and vertical lines towards the horizontal VPs and the zenith VP. 1×1: a 1×1 convolution. Horizon line is shown as red line, and estimated results (vertical fov, pitch, roll, and focal) are attached to the output image. The structures of the encoder and decoder are described in Sec. 3.3.

used the encoder of the transformer with the input of image patches, named ViT, for image recognition. Following the ViT, many vision transformers have been proposed to improve the performance of vision tasks [8, 24, 42]. Swin Transformer [24] whose representation is computed with shifted windows has been presented to achieve better results on several vision tasks. For object detection, Carion et al. [3] proposed a model called DETR based on transformer encoder-decoder structure to get accurate results. LERT [43] which is based on multi-Transformer has been proposed for line detection. Transformer mainly utilizes the self-attention mechanism to extract intrinsic features and shows great potential for extensive use in AI applications [8, 37].

In the camera calibration area, CTRL-C [20] fed multimodal inputs, including the set of image features and the line segments of the image, to the transformer architecture. Following the decoder transformer, feed-forward networks (FFNs) are used to predict all the outputs.

Although CTRL-C achieves SOTA results, it still has two main drawbacks: 1) the encoder transformer just uses block4 of ResNet [10] as image features, and useful information of the other blocks is lost; 2) in this framework, salient features, such as the global structures, are not used sufficiently without coarse-to-fine step.

To solve these drawbacks, we propose a new method that adopts multi-scale architecture as a fundamental infrastructure. We process multi-block image features with two encoder transformers and concatenate the two independent transformers through the decoder. With this operation, our framework can preserve and process global and detailed information to generate more accurate attention maps, which will benefit the camera calibration.

3. Methodology

3.1. Camera model

We first describe the geometric camera model utilized in this paper. A pinhole camera model converts 3D world coordinates $W \in \mathbb{R}^3$ to 2D image pixel coordinates: $(\lambda u, \lambda v, \lambda)^T = K[R|T][W|1]^T$, where K is the intrinsic matrix parameterized by focal length (f_x, f_y, commonly f_x=f_y) and principal points (u_x, u_y). We adopt common assumptions, including no skew, square pixels, and the principal point at the center of the image, to simplify our model for intrinsic matrix K. R and T represent camera rotation and translation in the world coordinate system. R is given by pitch θ and roll ψ (In our paper, yaw $= 0$ [13]).

Since focal length is influenced by image size, so we utilize vertical field of view $vfov$ to calculate the focal length f_y, like Eq. (1). For pitch θ and roll ψ, we utilize the horizon line midpoint hl_m and two endpoints (hl_{left} and hl_{right}) to present them intuitively, as Eq. (2)-Eq. (4) [48]:

$$f_y = (h/2)/\tan(vfov/2) \tag{1}$$

$$\tan\theta = (h/2 - hl_m)/f_y \tag{2}$$

$$hl_{left} = hl_m - (w/2) \times \tan\psi \tag{3}$$

$$hl_{right} = hl_m + (w/2) \times \tan\psi \tag{4}$$

where h and w are image height and width in pixels, and hl_m, hl_{left}, and hl_{right} in pixels are in the vertical axis of the image coordinate system. From top to bottom of the image frame, the coordinates are 0 and h, respectively.

3.2. Proposed method

In this section, we will explain in detail our proposed method, Multi-Scale transformers for Camera Calibration

(MSCC for short). MSCC framework overview is shown in Fig. 2.

Input. For the input image that meets widely used assumptions [5, 32], we utilize a Line Segment Detector (LSD [38]) to extract line segments, which is necessary for subsequent tasks, such as geometric cue estimation. Therefore, the queries will be composed of field of view (fov), horizon line (hl), zenith vanishing point (zvp), and line segments. However, if the input image is from a natural scene, the queries only consist of fov, hl, and zvp.

Multi-scale transformers. To obtain multi-scale image features for robust and accurate camera calibration, we employ ResNet50 [10] as the backbone network for our model. The input image is fed into ResNet50 network, which extracts feature representations at different scales. To leverage the multi-scale features in a coarse-to-fine manner, we utilize a cascaded transformer framework, as shown in Fig. 2. Specifically, the *block4* and *block3* image features from ResNet50 are used as inputs to the coarse and fine encoders of the transformers, respectively. The inputs to the coarse decoder include the output from the coarse encoder and the queries for the camera parameters. The fine decoder takes the queries from the coarse decoder and the outputs of the fine encoder to generate the final camera parameters.

Output. Finally, the outputs of the fine decoder are processed by Feed-Forward Networks (FFNs) to estimate fov, horizon line, and zvp. Moreover, for images that adhere to the Manhattan world assumptions, our model uses FFNs following softmax operators to classify segment tokens as horizontal lines or vertical lines at the end of the fine decoder. These estimated geometric cues are visualized in Fig. 2 as a segment block.

3.3. Encoder & Decoder

Both inputs of the coarse and fine encoder come from ResNet50. For a given input image of size $3 \times H \times W$, *block4* features (low-resolution) of size $C_1 \times H/32 \times W/32$ are fed into the coarse encoder and *block3* features (high-resolution) of size size $C_2 \times H/16 \times W/16$ are fed into the fine encoder. In our experiments, we set H, W, C_1, and C_2 as 512, 512, 2048, and 1024, respectively.

Both of the encoder and decoder contain six layers. As shown in Fig. 3, each encoding layer consists of a Multi-head Self-Attention (MSA), a feed-forward network (FFN), residual connections, and layer normalization. At each decoding layer, after the MSA block, the embeddings enter the Multi-head Cross-Attention (MCA) with the corresponding encoder outputs, then the output of it goes into the FFN.

In the transformer encoder, the input features filtered by a 1×1 convolution and then added with positional encodings enter a self-attention block with 8 attention heads.

In the transformer decoder, we combine the fov, horizon line, and zvp as query embeddings, a tensor of 3×256.

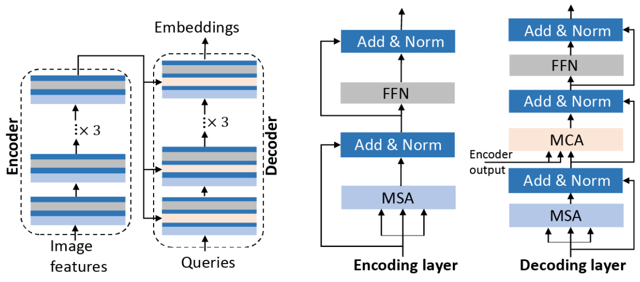

Figure 3. The structures of encoder and decoder, and encoding layer and decoding layer. An encoder consists of six encoding layers, and a decoder consists of six decoding layers.

If queries contain line segments, we sample 512 line segments as query embeddings. Therefore, there are 515 (3 for camera parameters, 512 for segments) embeddings in total. Then, the total queries are fed to the self-attention and cross-attention blocks.

3.4. Loss function

In the final prediction layers, the network produces five types of outputs that correspond to five losses: zvp loss, horizon line loss (*hl loss*), fov loss, vertical line loss (*verL loss*), and horizontal line loss (*horL loss*). To train our model, we adopt a deeply-supervised training strategy [19, 47]. Assume that the outputs of the coarse and fine decoders are denoted by O_1 and O_2, respectively. The methods without deep supervision only use O_2 for training and may cause the problem of gradient vanishing. If we use both O_1 and O_2 for deeply supervised training, the model can be optimized in both of the coarse and fine stages acceleratedly and more friendly.

The training loss is calculated in Eq. (5):

$$L_{total} = \frac{1}{n}\sum_{i=1}^{n}(L_{O_i}) \tag{5}$$

where n represents the number of decoders; O_i are outputs of the i-th decoder; L_{O_i} are the losses between O_i and ground truth. If outputs are zvp, hl, fov, and vertical and horizontal lines, the respective losses are zvp loss, horizon line loss (*hl loss*), fov loss, vertical lines loss (*verL loss*) and horizontal lines loss (*horL loss*), as shown in Eq. (6). If outputs only include zvp, fov, and hl, the L_{O_i} is calculated by Eq. (7).

$$L_{O_i} = L_{O_i^{zvp}} + L_{O_i^{hl}} + L_{O_i^{fov}} + L_{O_i^{verL}} + L_{O_i^{horL}} \tag{6}$$

$$L_{O_i} = L_{O_i^{zvp}} + L_{O_i^{hl}} + L_{O_i^{fov}} \tag{7}$$

where *zvp loss* ($L_{O_i^{zvp}}$) is calculated by minimizing the angular distance between the GT points and the prediction points; *hl loss* ($L_{O_i^{hl}}$) and *fov loss* ($L_{O_i^{fov}}$) are calculated

Methods	Up(↓)		Pitch(↓)		Roll(↓)		Fov(↓)		HL(↓)	
	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.
	Google Street View									
DeepHorizon [40]	-	-	-	-	-	-	-	-	0.1023	0.0742
UprightNet [41]	88.2337	89.9712	23.4713	19.8758	73.8171	74.4170	-	-	-	-
GPNet [21]	3.33	2.52	2.94	2.10	1.17	0.77	8.79	6.75	0.0672	0.0468
CTRL-C [20]	1.8582	1.5959	1.6254	1.3499	0.6786	0.5313	3.6803	2.8113	0.0321	0.0249
ParamNet [17]	-	-	**1.4155**	**1.2073**	0.6811	0.5305	3.2531	2.4380	-	-
Our: MSCC	**1.7155**	**1.4200**	1.5003	1.2163	**0.6152**	**0.4945**	**3.2110**	**2.2968**	**0.0305**	**0.0232**
	SYN-Citypark									
DeepHorizon [40]	-	-	-	-	-	-	-	-	0.2872	0.2190
UprightNet [41]	90.7437	90.9487	9.7092	7.2401	80.5249	82.1179	-	-	-	-
GPNet [21]	15.34	9.94	13.50	7.86	5.32	3.16	19.39	15.61	0.2729	0.1571
CTRL-C [20]	4.4125	2.9771	3.5044	2.2126	**2.0186**	**1.1401**	8.7922	7.7762	0.0929	0.0554
ParamNet [17]	-	-	24.5867	20.9082	11.5721	6.3776	9.4344	7.6991	-	-
Our: MSCC	**3.1313**	**2.1863**	**1.4536**	**0.9723**	2.5161	1.4863	**1.5029**	**1.0318**	**0.0553**	**0.0329**
	Flickr									
DeepHorizon [40]	-	-	-	-	-	-	-	-	0.4339	0.2545
UprightNet [41]	88.7126	88.7900	13.4263	10.5193	79.3308	80.7965	-	-	-	-
GPNet [21]	14.98	8.72	11.88	6.66	6.10	2.56	21.33	15.60	0.5314	0.1623
CTRL-C [20]	5.3660	3.5930	4.2396	2.5997	2.5464	1.4594	8.5284	6.8844	0.1162	0.0742
ParamNet [17]	-	-	24.2530	19.8243	10.8571	5.9863	8.6635	6.9038	-	-
Our: MSCC	**2.9613**	**1.9929**	**2.3762**	**1.5076**	**1.2836**	**0.6013**	**3.5658**	**2.6263**	**0.0697**	**0.0436**

Table 1. The evaluation results on GSV, SYN-Citypark, and Flickr. Bold means the best value.

from distance metric; *verL loss* ($L_{O_i^{verL}}$) and *horL loss* ($L_{O_i^{horL}}$) are based on binary cross entropy. For the details of the five sub-losses, please refer to [20].

4. Experiments

Our framework is trained and tested on the datasets of Google Street View (GSV) [1, 20], SYN-Citypark [18], and Flickr [18]. It is also tested on the Horizon Lines in-the-Wild (HLW) [40] test sets for generalization purposes. In the training process, batch size is 8, and learning rate is set as 0.0001.

4.1. Datasets

Google Street View dataset. Google Street View (GSV) [20] consists of images of streets, buildings and landmarks that conform to the Manhattan world assumption. The dataset provides Ground Truth (GT) values for the focal length, pitch, and roll, enabling the calculation of the GT values for the up vector, zvp, and horizon line based on the focal length and rotation matrix. Moreover, the dataset includes GT values for horizontal vanishing points. Thus, the GSV dataset can be used to predict the vertical and horizontal lines and estimate the focal length and 2 degrees of freedom (DoF) (pitch and roll) rotation by MSCC. The numbers of training, validation, and test samples from GSV are 12679, 535, and 1333, respectively.

Pano360 dataset. The Pano360 dataset, presented by Kocabas et al. [18], consists of equirectangular panorama images divided into two collections: the Flickr dataset and the SYN-Citypark dataset. The Flickr dataset is a diverse collection with multiple sets grouped by rich tags, while SYN-Citypark is a photorealistic synthetic dataset focusing on city park scenes. These datasets mainly feature natural scenes, such as plants, seas, skies, and everyday human activities in natural settings. Since obtaining GT values for horizontal vanishing points is challenging, both datasets provide only GT values for the focal length, pitch, and roll. Therefore, the MSCC model can only estimate 2-DoF rotation and focal length. The Flickr dataset contains 171,299 training images, 7,877 validation images, and 17,720 test images, while the SYN-Citypark dataset has 6,295 training images, 290 validation images, and 651 test images. We also trained the MSCC model on each dataset.

Horizon Lines in-the-Wild dataset. Horizon Lines in-the-Wild (HLW) [40] provides the ground truth of positions of horizon lines. We just use this dataset to test our model's ability to detect the position of the horizon line. We randomly choose 150 images (10 images for 15 scenes) as the test set.

4.2. Comparisons

Our proposed method outperforms SOTA methods both subjectively and objectively. Fig. 4 illustrates that our horizon line estimation closely matches the ground truth in a wide range of indoor and outdoor scenes. Moreover, Fig. 6 showcases the performance of our model in estimating the geometric position and relationship of the horizon line/zenith line, horizontal lines, and vertical lines in various image examples.

We compare our method to ParamNet [17], UprightNet [41], GPNet [21], CTRL-C [20], and DeepHorizon [40] on the GSV, Flickr, and SYN-Citypark datasets. Tab. 1 re-

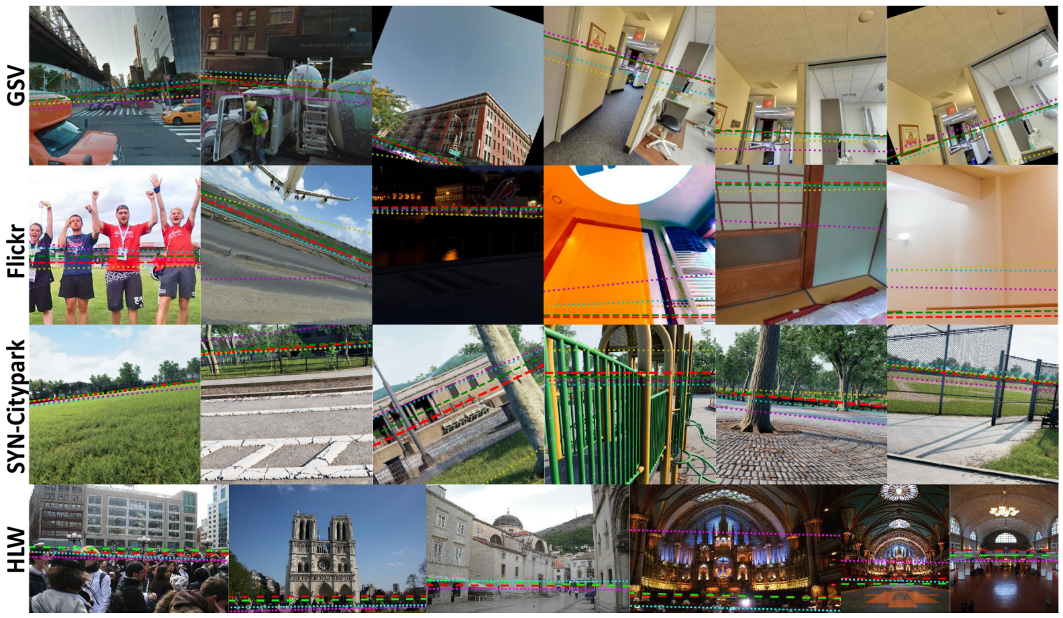

Figure 4. Comparison of horizon line on GSV, Flickr, SYN-Citypark, and HLW: Red dashed lines: GT; Green dashed lines: our method; Cyan dotted lines: CTRL-C; Magenta dotted lines: GPNet; Yellow dotted lines: DeepHorizon.

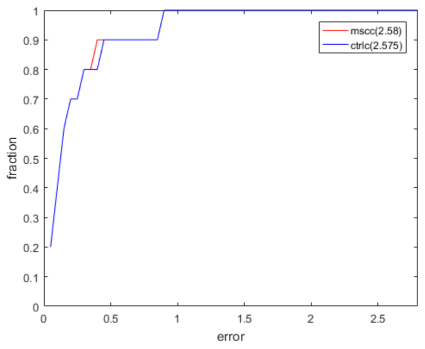

Figure 5. Comparison of Cumulative Error Distribution (CED) curves on HLW.

ports the mean and median errors, and we note that Upright-Net uses a pre-trained model on another dataset due to the lack of required supervision in the GSV and Pano360 [21]. The estimation results of the model trained on the large Flickr dataset are more stable than those trained on the smaller GSV and SYN-Citypark datasets. This is because the models trained on the smaller datasets have slightly higher pitch and roll error results than ParamNet and CTRL-C, respectively. On Flickr, our method surpasses all the SOTA methods. On the HLW dataset, we utilize Cumulative Error Distribution (CED) as the performance evaluation indicator and present the results of our method and CTRL-C in Fig. 5, as GPNet's error exceeds 2.8 and falls outside the range. The Area Under the Curve (AUC) value of our method is similar to CTRL-C. Based on the subjective results in Fig. 4 and objective results in Fig. 5, we conclude that our method has good generalization capability.

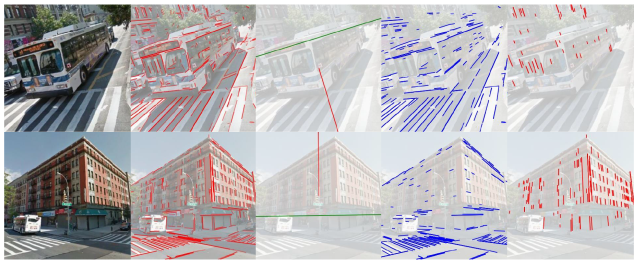

Figure 6. Visual results on GSV obtained by our method. From left to right: input images, line segments extracted by LSD [38], horizon line (*green*) and zenith line through the image center (*red*), horizontal lines toward the horizontal VPs, and vertical lines toward the zenith VP.

4.3. Attention map

To check the effectiveness of the coarse-to-fine strategy and study feature representation for camera calibration, we exhibit attention maps of horizon line responses in the decoding process in Fig. 7. The baseline method CTRL-C [20] pays more attention to the ground textures and edges other than the whole structure of buildings, sky, walls, ceilings, etc. Such attention can not help to find accurate horizon lines. In our method, in the coarse stage, the model captures the intersection parts between the outdoor land and buildings/sky, and the intersection parts between the indoor

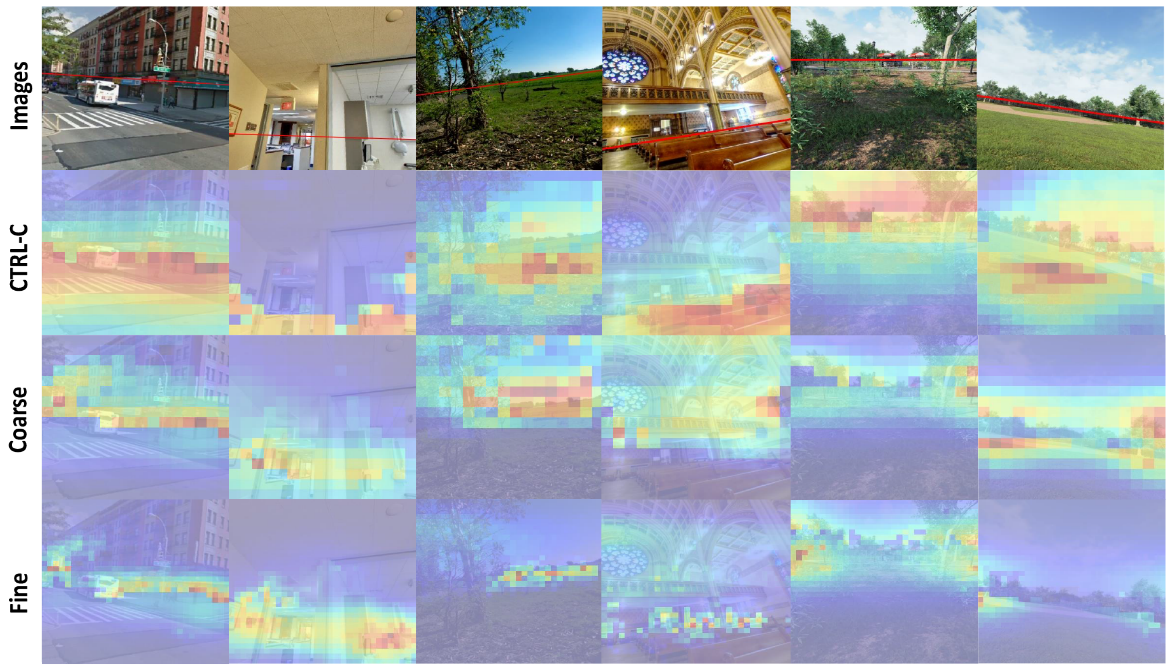

Figure 7. Visualization of attention maps in the decoding process displayed as a jet overlay. From top to down: the input images with the ground truth of the horizon lines (shown in *red*); the attention maps of CTRL-C; the attention maps of the coarse stage of our MSCC; the attention maps of the fine stage of our MSCC. From left to right, two columns as a group are from GSV, Flickr, and SYN-Citypark, respectively.

low part and ceilings. These intersection parts show the rough areas of the horizon lines with correct positions. In the fine stage, the model focuses on refined areas based on the rough areas. Moreover, the weights of the coarse stage are also optimized by the flow from the fine stage in the back propagation. In a word, we think this benefit comes from the multi-scale features, coarse-to-fine strategy, and deep supervision.

4.4. Ablation study

In this section, we will discuss in detail whether it is effective to utilize multi-scale features, deep supervision, and the coarse-to-fine strategy in our proposed method.

The capability of multi-scale features. The decoder consists of 6 decoding layers. Each layer appending the last FFNs gets the output results. In our method, there are 2 decoders, and 12 layers in total. We compare the errors of each layer in the testing phase, as shown in Fig. 8. From shallow layers to deep layers, the errors gradually decrease. The fine decoder results (7th layer∼12th layer) are based on the coarse decoder outputs (1th∼6th layer) and the high-resolution features. Because the errors in the fine stage are lower than the ones in the coarse stage, we find that the multi-scale features help to reduce the errors from shallow (global features) to deep (detailed features).

The influence of deep supervision. In addition, we analyze the influence of deep supervision on our training process. We only utilize the fine decoder results to optimize our model without deep supervision. The results are shown in Fig. 9. In the same way, from shallow layers to deep layers, the errors decrease. However, compared with Fig. 8, we find that error curves jitter violently in the shallow layers. Moreover, in Fig. 8, the error curves go down more smoothly. It means the model using deep supervision is easier to be optimized. In Tab. 2, we also compare the errors of the final layer between MSCC without and with deep supervision. We can see that the latter can get better results than the former.

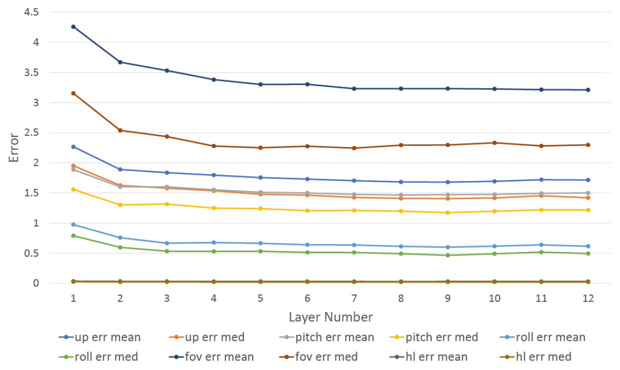

Figure 8. The errors of decoding layers of our MSCC on GSV.

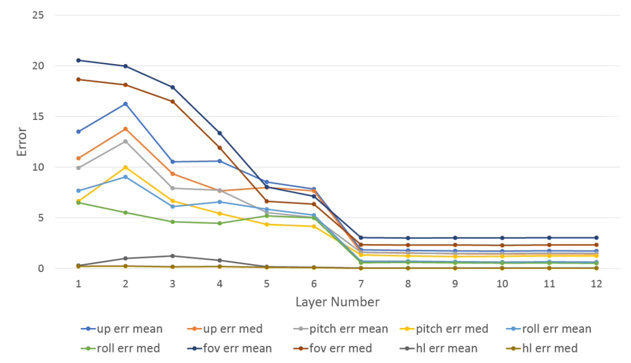

Figure 9. The errors of decoding layers of our MSCC without deep supervision on GSV.

Methods	Up(↓)		Pitch(↓)		Roll(↓)		Fov(↓)		HL(↓)	
	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.
	Google Street View									
w/o	1.7181	1.4376	**1.4896**	1.2478	0.6319	0.5093	**3.0289**	2.3268	0.0314	0.0240
w/	**1.7155**	**1.4200**	1.5003	**1.2163**	**0.6152**	**0.4945**	3.2110	**2.2968**	**0.0305**	**0.0232**
	Flickr									
w/o	**2.9276**	2.0085	**2.3511**	1.5216	**1.2536**	**0.5798**	**3.4986**	**2.6162**	**0.0675**	**0.0422**
w/	2.9613	**1.9929**	2.3762	**1.5076**	1.2836	0.6013	3.5658	2.6263	0.0697	0.0436
	SYN-Citypark									
w/o	3.1907	2.2840	1.5494	1.1380	2.5494	1.6482	1.5708	**1.0225**	**0.0538**	0.0338
w/	**3.1313**	**2.1863**	**1.4536**	**0.9723**	**2.5161**	**1.4863**	**1.5029**	1.0318	0.0553	**0.0329**

Table 2. The comparison between MSCC without (w/o) and with (w/) deep supervision. Bold means the best results.

Methods	Up(↓)		Pitch(↓)		Roll(↓)		Fov(↓)		HL(↓)	
	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.
	Google Street View									
F2C	1.8019	1.4887	1.5813	1.2340	0.6489	0.5299	3.2367	**2.1701**	0.0330	0.0253
C2F	**1.7155**	**1.4200**	**1.5003**	**1.2163**	**0.6152**	**0.4945**	**3.2110**	2.2968	**0.0305**	**0.0232**
	Flickr									
F2C	3.2787	2.2798	2.6125	1.7135	1.4467	0.7275	4.1958	3.1600	0.0740	0.0469
C2F	**2.9613**	**1.9929**	**2.3762**	**1.5076**	**1.2836**	**0.6013**	**3.5658**	**2.6263**	**0.0697**	**0.0436**
	SYN-Citypark									
F2C	13.5015	12.1408	10.7295	8.5338	5.3695	2.6545	13.7462	11.7831	0.2718	0.2467
C2F	**3.1313**	**2.1863**	**1.4536**	**0.9723**	**2.5161**	**1.4863**	**1.5029**	**1.0318**	**0.0553**	**0.0329**

Table 3. The comparison between strategies of fine-to-coarse (F2C) and coarse-to-fine (C2F) for our proposed method. Bold means the best results.

The advantage of coarse-to-fine strategy. Coarse-to-fine is a very powerful architecture for combining multi-scale features and we choose this strategy as our backbone to process multi-level features. To explain more clearly, we compare our proposed method with different strategies, like our method with coarse-to-fine strategy (C2F) and our method with fine-to-coarse strategy (F2C). The F2C strategy means inputting high-resolution features to the first encoder of the first transformer and low-resolution features to the second encoder of the second transformer, whose input order is opposite to the C2F strategy. The comparison results are shown in Tab. 3. From this table, we can see that C2F outperforms F2C in most cases.

4.5. Line classification

Line classification (horizontal lines or vertical lines) is an auxiliary task for CTRL-C and our model. We compare the results among CTRL-C and our two models (with and without deep supervision), as shown in Tab. 4. The accuracies are calculated based on two types of line segments directed towards the zvp and hvps within $[85°, 88°]$ in vertical and horizontal convergence lines, respectively. With the same conclusion, the model with deep supervision gets the best results.

Methods	Accuracy(↑)	
	Vertical	Horizontal
CTRL-C [20]	99.7333	93.5645
Our: w/o	99.8434	96.0623
Our: w/	**99.8511**	**96.1347**

Table 4. Accuracy results for the horizontal and vertical lines. Bold means the best values.

5. Application

We utilize calibrated results (fov, focal, pitch, and roll) from a single image to project 3D bodies into image space. 3D human parameters (shape and pose) are obtained by the SMPL [25] method. Camera intrinsics K is set as $diag([f, f, 1])$, where f is the focal length in pixels. Camera rotation matrix R is parameterized by roll and pitch angles and camera translation T is set as $[0, 0, 0]$. Body translation is calculated by SPEC [18]. Projection results are shown in Fig. 10, where the green line is the horizon line.

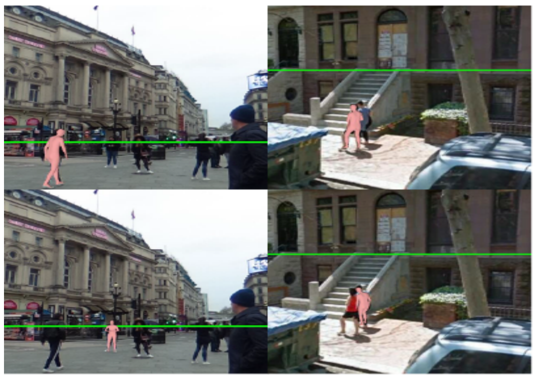

Figure 10. Examples of projected 3D bodies in image space using the estimated camera parameters with our method.

6. Conclusion

In this paper, we propose a camera calibration method based on multi-scale transformers, a coarse-to-fine strategy, and deep supervision. Our method outperforms all the state-of-the-art methods by objective and subjective experiments on Google Street View and Pano360 (Flickr and SYN-Citypark) datasets. The method also shows good generalization capability on Horizon Lines in-the-Wild dataset. For future work, we will study cross correlation among multi-scale features to improve representation capability.

References

[1] Google street view images api. `https://developers.google.com/maps/`.

[2] Oleksandr Bogdan, Viktor Eckstein, Francois Rameau, and Jean-Charles Bazin. Deepcalib: a deep learning approach for automatic intrinsic calibration of wide field-of-view cameras. In *Proceedings of the 15th ACM SIGGRAPH European Conference on Visual Media Production*, pages 1–10, 2018.

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[4] Sung-Jin Cho, Seo-Won Ji, Jun-Pyo Hong, Seung-Won Jung, and Sung-Jea Ko. Rethinking coarse-to-fine approach in single image deblurring. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4641–4650, 2021.

[5] James Coughlan and Alan L Yuille. The manhattan world assumption: Regularities in scene statistics which enable bayesian inference. *Advances in Neural Information Processing Systems*, 13, 2000.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[7] Shang-Hua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr. Res2net: A new multi-scale backbone architecture. *IEEE transactions on pattern analysis and machine intelligence*, 43(2):652–662, 2019.

[8] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on visual transformer. *arXiv e-prints*, pages arXiv–2012, 2020.

[9] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Moein Heidari, Amirhossein Kazerouni, Milad Soltany, Reza Azad, Ehsan Khodapanah Aghdam, Julien Cohen-Adad, and Dorit Merhof. Hiformer: Hierarchical multi-scale representations using transformers for medical image segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6202–6212, 2023.

[12] Markus Hofinger, Samuel Rota Bulò, Lorenzo Porzi, Arno Knapitsch, Thomas Pock, and Peter Kontschieder. Improving optical flow on a pyramid level. In *European Conference on Computer Vision*, pages 770–786. Springer, 2020.

[13] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2354–2363, 2018.

[14] Sina Honari, Jason Yosinski, Pascal Vincent, and Christopher Pal. Recombinator networks: Learning coarse-to-fine feature aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5743–5752, 2016.

[15] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[16] Chuong Huynh, Anh Tuan Tran, Khoa Luu, and Minh Hoai. Progressive semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16755–16764, 2021.

[17] Linyi Jin, Jianming Zhang, Yannick Hold-Geoffroy, Oliver Wang, Kevin Blackburn-Matzen, Matthew Sticha, and David F Fouhey. Perspective fields for single image camera calibration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17307–17316, 2023.

[18] Muhammed Kocabas, Chun-Hao P Huang, Joachim Tesch, Lea Müller, Otmar Hilliges, and Michael J Black. Spec: Seeing people in the wild with an estimated camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11035–11045, 2021.

[19] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial intelligence and statistics*, pages 562–570. PMLR, 2015.

[20] Jinwoo Lee, Hyunsung Go, Hyunjoon Lee, Sunghyun Cho, Minhyuk Sung, and Junho Kim. Ctrl-c: Camera calibration transformer with line-classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16228–16237, 2021.

[21] Jinwoo Lee, Minhyuk Sung, Hyunjoon Lee, and Junho Kim. Neural geometric parser for single image camera calibration. In *European Conference on Computer Vision*, pages 541–557. Springer, 2020.

[22] Yuxi Li, Weiyao Lin, John See, Ning Xu, Shugong Xu, Ke Yan, and Cong Yang. Cfad: Coarse-to-fine action detector for spatiotemporal action localization. In *European Conference on Computer Vision*, pages 510–527. Springer, 2020.

[23] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[24] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[25] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015.

[26] Zhipeng Luo, Gongjie Zhang, Changqing Zhou, Tianrui Liu, Shijian Lu, and Liang Pan. Transpillars: Coarse-to-fine aggregation for multi-frame 3d object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4230–4239, 2023.

[27] Yi Ma, Stefano Soatto, Jana Košecká, and Shankar Sastry. *An invitation to 3-d vision: from images to geometric models*, volume 26. Springer, 2004.

[28] Radu Orghidan, Joaquim Salvi, Mihaela Gordan, and Bogdan Orza. Camera calibration using two or three vanishing points. In *2012 Federated Conference on Computer science and information systems (FedCSIS)*, pages 123–130. IEEE, 2012.

[29] Rui Qian, Di Hu, Heinrich Dinkel, Mengyue Wu, Ning Xu, and Weiyao Lin. Multiple sound sources localization from coarse to fine. In *European Conference on Computer Vision*, pages 292–308. Springer, 2020.

[30] Niamul Quader, Juwei Lu, Peng Dai, and Wei Li. Towards efficient coarse-to-fine networks for action and gesture recognition. In *European Conference on Computer Vision*, pages 35–51. Springer, 2020.

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[32] 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. IEEE, 2004.

[33] Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3626–3633, 2013.

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

[35] Xu Song, Xiao-Jun Wu, and Hui Li. Msdnet for medical image fusion. In *International conference on image and graphics*, pages 278–288. Springer, 2019.

[36] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[38] Rafael Grompone Von Gioi, Jérémie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: a line segment detector. *Image Processing On Line*, 2:35–55, 2012.

[39] Scott Workman, Connor Greenwell, Menghua Zhai, Ryan Baltenberger, and Nathan Jacobs. Deepfocal: A method for direct focal length estimation. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 1369–1373. IEEE, 2015.

[40] Scott Workman, Menghua Zhai, and Nathan Jacobs. Horizon lines in the wild. *arXiv preprint arXiv:1604.02129*, 2016.

[41] Wenqi Xian, Zhengqi Li, Matthew Fisher, Jonathan Eisenmann, Eli Shechtman, and Noah Snavely. Uprightnet: geometry-aware camera orientation estimation from single images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9974–9983, 2019.

[42] Peng Xu, Xiatian Zhu, and David A Clifton. Multimodal learning with transformers: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[43] Yifan Xu, Weijian Xu, David Cheung, and Zhuowen Tu. Line segment detection using transformers without edges. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4257–4266, 2021.

[44] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2403–2412, 2018.

[45] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the seventh ieee international conference on computer vision*, volume 1, pages 666–673. Ieee, 1999.

[46] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

[47] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 3–11. Springer, 2018.

[48] Rui Zhu, Xingyi Yang, Yannick Hold-Geoffroy, Federico Perazzi, Jonathan Eisenmann, Kalyan Sunkavalli, and Manmohan Chandraker. Single view metrology in the wild. In *European Conference on Computer Vision*, pages 316–333. Springer, 2020.