VPDETR: End-to-End Vanishing Point DEtection TRansformers

Taiyan Chen, Xianghua Ying*, Jinfan Yang, Ruibin Wang, Ruohao Guo, Bowei Xing, Ji Shi
National Key Laboratory of General Artificial Intelligence, School of Intelligence Science and Technology, Peking University
chenty@stu.pku.edu.cn, xhying@pku.edu.cn

Abstract
In the field of vanishing point detection, previous works commonly relied on extracting and clustering straight lines or classifying candidate points as vanishing points. This paper proposes a novel end-to-end framework, called VPDETR (Vanishing Point DEtection TRansformer), that views vanishing point detection as a set prediction problem, applicable to both Manhattan and non-Manhattan world datasets. By using the positional embedding of anchor points as queries in Transformer decoders and dynamically updating them layer by layer, our method is able to directly input images and output their vanishing points without the need for explicit straight line extraction and candidate points sampling. Additionally, we introduce an orthogonal loss and a cross-prediction loss to improve accuracy on the Manhattan world datasets. Experimental results demonstrate that VPDETR achieves competitive performance compared to state-of-the-art methods, without requiring post-processing.

Introduction
Vanishing points are intersection points of parallel lines in the 3D world projected onto the 2D image under the pinhole camera model. Vanishing point detection is a fundamental problem in 3D vision. An accurate vanishing point detection algorithm can benefit many tasks, such as camera calibration (Cipolla, Drummond, and Robertson 1999; Antone and Teller 2000; Grammatikopoulos, Karras, and Petsa 2007), wireframe parsing (Zhou et al. 2019b; Zhou, Qi, and Ma 2019), 3D reconstruction (Guillou et al. 2000), photo forensics (O’Brien and Farid 2012), object detection (Hoiem, Efros, and Hebert 2008), autonomous driving (Lee et al. 2017), and visual SLAM (Davison et al. 2007; Li et al. 2019a).

As shown in Figure 1, traditional vanishing point detection methods (Koegcka and Zhang 2002; Tardif 2009) generally include three steps: line/contour detection, line clustering/classification, and vanishing point regression. In recent years, deep learning approaches (Borji 2016; Zhai, Workman, and Jacobs 2016; Chang, Zhao, and Itti 2018; Zhang et al. 2018; Zhai, Workman, and Jacobs 2016; Kluger et al. 2017; Zhou et al. 2019a; Liu, Zhou, and Zhao 2021; Tong et al. 2022; Lin et al. 2022) have shown great potential in this task. Some of the most recent state-of-the-art works suffer from some problems. NeurVPS (Zhou et al. 2019a) and Lin et al. (Lin et al. 2022) need to sample a large number of candidate points in the Gaussian sphere and classify whether they are vanishing points which is slow during inference. TLC (Tong et al. 2022) uses Transformer to classify straight lines into four categories, requiring additional line category annotations. Post-processing is also required to enhance performance. Although these works are capable of end-to-end training, they still require additional post-processing steps to produce the vanishing point as the final output. In other words, they are not entirely end-to-end because the vanishing point is not directly predicted when an image is inputted.

DETR (Carion et al. 2020) is a highly influential work in the field of object detection. It uses a transformer encoder-decoder architecture for end-to-end object detection. In DETR, object queries are treated as learnable queries, allowing the model to make predictions in a parallel manner, which improves the efficiency. It also uses a set loss function to handle the problem of object permutation, thus eliminates many hand-designed components such as NMS (Non-Maximum Suppression). Some follow-up works such as Deformable DETR (Zhu et al. 2020) and DAB-DETR (Liu et al. 2022) have been proposed to improve its convergence and performance. There are also works on adapting DETR for other applications (Xu et al. 2021; Misra, Girdhar, and Joulin 2021; Prangemeier, Reich, and Koeppl 2020). Drawing inspiration from these studies, we modified DETR and its variations to address the problem of vanishing point detection.

In this paper, we propose VPDETR, an end-to-end framework for vanishing point detection. Our modification focuses on the decoder part, utilizing anchor vanishing point positional encoding as queries and updating point coordinates layer by layer. Our method is based on set prediction, therefore overcomes the limitations of the Manhattan world assumption, i.e. there are three orthogonal vanishing points in each image, enabling it to predict vanishing points for non-Manhattan world data as well. Specifically, for Manhattan world data, we introduce cross-prediction loss and orthogonal loss to improve the results. Moreover, our method achieves fast inference speed of 16 FPS, outperforming state-of-the-art (SOTA) method (Zhou et al. 2019a).
that rely on sampling candidate points and classifying them one by one (0.5 FPS).

Our work presents the following contributions: (1) We introduce VPDETR, a novel fully end-to-end framework for vanishing point detection, which is capable of handling both Manhattan and non-Manhattan world datasets by treating vanishing point detection as a set prediction problem. (2) For the Manhattan world assumption, we propose a cross-prediction loss and orthogonal loss to improve accuracy. (3) Our experiments demonstrate that VPDETR achieves a good balance of accuracy and inference speed, comparable to state-of-the-art models. And we provide ablation experiments to showcase the effectiveness of the proposed components.

Related Work

Vanishing Point Detection. Vanishing point detection is a fundamental problem in computer vision that locates the point of convergence of parallel lines in an image. The problem was first introduced in (Barnard 1983), and has since been tackled using various approaches. The dominant approach for vanishing point detection is line-based, which generally consist of three steps. Firstly, line detection (Canny 1986; Von Gioi et al. 2008) or contour detection (Arbelaez et al. 2010) is performed. Secondly, the parametric lines are clustered using various algorithms such as RANSAC (Bolles and Fischler 1981; Wu et al. 2021), Hough transform (Lutton, Maitre, and Lopez-Krahe 1994), J-Linkage (Tardif 2009), EM (Kogecka and Zhang 2002; Kosecka and Zhang 2002), and dual space (Lezama et al. 2014). Finally, geometry is used to estimate the vanishing point.

Recently, learning-based methods (Borji 2016; Chang, Zhao, and Itti 2018; Zhang et al. 2018; Shi et al. 2019; Zhai, Workman, and Jacobs 2016; Kluger et al. 2017; Li et al. 2021; Kluger et al. 2020) have been shown to be effective for estimating the vanishing point in images with complex backgrounds and cluttered scenes. NeurVPS (Zhou et al. 2019a) performs best, which introduces conic convolution to enhance the detection accuracy, albeit at the expense of slow inference speed. To address this issue, Liu et al. (Liu, Zhou, and Zhao 2021) proposes an efficient conic convolution in VaPiD. Lin et al. (Lin et al. 2022) incorporates the Hough transform and the Gaussian sphere into deep neural networks to improve model generalization. TLC (Tong et al. 2022) is the first to apply Transformer (Vaswani et al. 2017) to vanishing point detection, utilizing both the geometric and contextual features of lines to improve the accuracy.

DETR and Its Variants. Carion et al. (Carion et al. 2020) introduced DETR (DEtection TRansformer), a Transformer-based end-to-end object detector that eliminates the need for hand-designed components such as anchor design and NMS. Several studies delve into a deeper understanding of the decoder queries in DETR. Many papers associate queries with spatial position from different angles. Deformable DETR (Zhu et al. 2020) predicts 2D anchor points and employs a deformable attention mechanism that focuses on specific sampling points around a reference point. Efficient DETR (Yao et al. 2021) selects the top K positions from the dense prediction of the encoder to improve decoder queries. DAB-DETR (Liu et al. 2022) extends the representation of queries from 2D anchor points to 4D anchor box coordinates, and enabling dynamic updates of boxes in each layer of the decoder. More recently, DN-DETR (Li et al. 2022) introduced a denoising training method to accelerate DETR training. And DINO (Zhang et al. 2022) further introduces three tricks to get better performance.

Meanwhile, there have also been studies exploring the adaptation of DETR for other applications such as line segment detection (Xu et al. 2021), 3D object detection (Prangemeier, Reich, and Koepl 2020), and instance segmentation (Prangemeier, Reich, and Koepl 2020). In this work, we have sensibly adapted DETR to make it appropriate for vanishing point detection tasks, resulting in precise predictions and a truly end-to-end output.

Method

Overview
Our model includes a CNN backbone, Transformer encoders and decoders, and multilayer perceptron (MLP) prediction heads for vanishing points and confidence score. The main improvement of our model is in the decoder part, as illustrated in Figure 2.
Given an image, the features are extracted using a CNN backbone and refined by the Transformer encoders. The decoder then employs dual queries, which consist of both positional queries (anchor points) and content queries (decoder embeddings), to determine the vanishing points that correspond to the anchor points. The dual queries are iteratively refined across multiple layers, slowly approaching the target ground-truth vanishing points. Finally, the outputs of the final decoder layer are utilized to predict confidence scores and vanishing points through prediction heads. A bipartite graph matching is then conducted to calculate the loss, following the procedure used in DETR.

**Anchor Point Learning and Iterative Updating**

Inspired by (Liu et al. 2022), we propose to directly learn anchor points and derive positional queries from these anchors.

We use the Gaussian sphere representation of the vanishing point, the details of which can be found in Section 3.2 of NeurVPS (Zhou et al. 2019a). We denote $A_q = (x_q, y_q, z_q)$ as the $q$-th anchor point, where $x_q, y_q, z_q \in \mathbb{R}$, take values in $[-1, 1]$. The positional query (Meng et al. 2021; Liu et al. 2022) $P_q$ associated with it can be obtained from

$$P_q = \text{MLP}^p(\text{Cat}(\text{PE}(x_q), \text{PE}(y_q), \text{PE}(z_q)))$$

(1)

where $P_q \in \mathbb{R}^D$, Cat means concatenate, and PE means sinusoidal positional encoding:

$$\text{PE}(x)_{2i} = \sin \left( \frac{x}{T^{2i/D}} \right), \quad \text{PE}(x)_{2i+1} = \cos \left( \frac{x}{T^{2i/D}} \right)$$

(2)

where $T$ is temperature, a hyper-parameter, and $i$ denotes the index in the vector. In our implementations, PE maps a float to a vector with $D/2$ dimensions as $\text{PE} : \mathbb{R} \rightarrow \mathbb{R}^{D/2}$, and $\text{MLP}^p$ projects a $3D/2$ dimensional vector into $D$ dimensions $\text{MLP}^p : \mathbb{R}^{3D/2} \rightarrow \mathbb{R}^D$. Following (Liu et al. 2022), we learn a $\text{MLP}^* : \mathbb{R}^D \rightarrow \mathbb{R}^D$ to obtain a scale vector conditional on the content query $C_q$ and multiply it with the positional encoding element by element:

$$\hat{P}_q = P_q \cdot \text{MLP}^*(C_q)$$

(3)

In the self-attention module, queries, keys, and values are formed as:

$$Q_q = C_q + \hat{P}_q, \quad K_q = C_q + \hat{P}_q, \quad V_q = C_q$$

(4)

after the self-attention operation, skip-connection and normalization, we can get the new content information. Next, we use the deformable attention module proposed in Deformable DETR (Zhu et al. 2020) to perform cross-attention operations, as illustrated in Figure 2. Finally, the learned embeddings $E_i^l$ are fed into the score branch and the prediction branch, i.e., two MLP, to obtain the confidence scores and the residuals of vanishing points, where the subscript $t$ take values between 1 and number of queries, and the superscript $l$ means that the feature is output by the $l$-th decoder layer.

$$\text{score}^l_i = \text{MLP}^{\text{score}}(E_i^l)$$

(5)

$$(\Delta x_i, \Delta y_i, \Delta z_i) = \text{MLP}^{\text{pred}}(E_i^l)$$

(6)

$$(x_{i}^l, y_{i}^l, z_{i}^l) = (x_{i}^{l-1} + \Delta x_i, y_{i}^{l-1} + \Delta y_i, z_{i}^{l-1} + \Delta z_i)$$

(7)

where $(x_{i}^l, y_{i}^l, z_{i}^l)$ is the $i$-th prediction of the $l$-th decoder layer, and for initialization, $(x_{i}^0, y_{i}^0, z_{i}^0)$ are randomly sampled in $(-1, 1)$. $\text{score}^l_i$ represents the prediction quality, the smaller the angle between the prediction and the ground truth, the higher the score.

**Bipartite Matching**

Our VPDETR predicts a set of $N$ vanishing points $\{\hat{\text{VP}}_i = (x_{i}^l, y_{i}^l, z_{i}^l); i = 1, \ldots, N\}$ in one pass through the $l$-th decoder layer, with $N$ set to 20 in this study. For simplicity, we omit the superscript $l$. We conduct a set-based bipartite matching between the predicted vanishing points and ground-truth targets to determine whether a prediction is associated with a real vanishing point and will be involved in the calculation of the loss function during the training stage.
Assume there are $M$ target vanishing points $\{\text{VP}_j; j = 1, \ldots, M\}$, we optimize a bipartite matching objective on a permutation function $\sigma(\cdot): \mathbb{Z}_+ \to \mathbb{Z}_+$, which maps prediction indices $\{1, \ldots, N\}$ to potential target indices $\{1, \ldots, N\}$, including $\{1, \ldots, M\}$ for matched predictions, and $\{M + 1, \ldots, N\}$ for unmatched predictions:

$$L_{\text{match}} = \sum_{i=1}^{N} 1_{\{\sigma(i) \leq M\}} \left[ \lambda_1 d \left( \text{VP}_{\sigma(i)}, \text{VP}_i \right) + \lambda_2 a \left( \text{VP}_{\sigma(i)}, \text{VP}_i \right) \right]$$

(8)

where $d(\cdot, \cdot)$ represents $L_1$ distance between vectors, and $a(\cdot, \cdot)$ represents the angle between vectors. $1_{\{\cdot\}}$ is an indicator function. The optimal permutation $\sigma^*$ is computed using a Hungarian algorithm, mapping $M$ positive prediction indices to ground-truth indices $\{1, \ldots, M\}$.

**Losses**

We compute losses based on the optimal permutation $\sigma^*$ from the bipartite matching procedure introduced in Sec, in which $\{i; \sigma^*(i) \leq M\}$ represents positive predictions indices.

**Regression Loss.** We use a simple $L_1$ distance loss as a baseline of regression loss:

$$L_{\text{reg}} = \sum_{i=1}^{M} d \left( \text{VP}_{\sigma^*(i)}, \text{VP}_i \right)$$

(10)

where $d(\cdot, \cdot)$ represents $L_1$ distance between vectors.

**Cross-prediction Loss.** For datasets that fits the Manhattan world assumption, we design a novel cross-prediction loss. Denote the normalized predicted vanishing point $\text{VP}_{\sigma^*(i)}$ that matches the ground-truth $\text{VP}_i$ as $\hat{\text{VP}}_i$:

$$\hat{\text{VP}}_i = \frac{\text{VP}_{\sigma^*(i)}}{\|\text{VP}_{\sigma^*(i)}\|}$$

(11)

then the cross-prediction loss is formulated as:

$$L_{\text{cross}} = \text{dist} \left( \text{VP}_1, \hat{\text{VP}}_2 \times \hat{\text{VP}}_3 \right) + \text{dist} \left( \text{VP}_2, \hat{\text{VP}}_3 \times \hat{\text{VP}}_1 \right) + \text{dist} \left( \text{VP}_3, \hat{\text{VP}}_1 \times \hat{\text{VP}}_2 \right)$$

(12)

the intuition is that according to the orthogonality of the vanishing points in Manhattan world assumption, any two predicted vanishing points should be able to obtain another vanishing point by cross product. The order of the two elements participating in the cross product will cause the result to have different directions, therefore:

$$\text{dist}(a, b) = \min(d(a, b), d(-a, b))$$

(13)

where $a$ and $b$ are two vectors. Eq. 13 solves the wrong direction problem caused by cross product.

**Orthogonal Loss.** For datasets in the Manhattan world, we additionally propose an orthogonal loss:

$$L_{\text{orth}} = \|\text{VP}_1 \cdot \text{VP}_2\| + \|\text{VP}_2 \cdot \text{VP}_3\| + \|\text{VP}_3 \cdot \text{VP}_1\|$$

(14)

where $\cdot$ is inner-product. That is, the three predicted vanishing points should be orthogonal, the larger the inner product, the larger the loss value.

**Classification Loss.** The classification loss is based on binary cross-entropy loss (BCE), the input logits $(\text{score}_i, \text{conf}_i)$ are predicted in Eq. 5, and the target is determined by the angle between predicted vanishing points and the ground-truth. Denote the ground-truth value of the minimum angle with the predicted vanishing point $\text{VP}_i$ as $\text{VP}_{\rho(i)}$. The corresponding targets are designed as:

$$\text{score}_i^1 = \begin{cases} 0.01 \times (\text{degree}_i - 10^\circ)^2 & \text{if } \text{degree}_i < 10^\circ \\ 0 & \text{otherwise} \end{cases}$$

(15)

where

$$\text{degree}_i = a \left( \text{VP}_{\rho(i)}, \hat{\text{VP}}_i \right)$$

(16)

The classification loss is formed as:

$$L_{\text{cls}} = \sum_{i=1}^{N} \text{BCE} \left( \text{score}_i, \text{score}_i^1 \right)$$

(17)

The final loss function is

$$L_{\text{total}} = \lambda_{\text{reg}} L_{\text{reg}} + \lambda_{\text{cls}} L_{\text{cls}}$$

(18)

as a baseline for general datasets, and

$$L_{\text{total}} = \lambda_{\text{cross}} L_{\text{cross}} + \lambda_{\text{orth}} L_{\text{orth}} + \lambda_{\text{reg}} L_{\text{reg}} + \lambda_{\text{cls}} L_{\text{cls}}$$

(19)

for Manhattan world datasets.

**Experiments**

**Datasets.** Since our method is based on set predictions, it can be applied both to datasets that following the Manhattan world assumption such as SU3 (Zhou et al. 2019b), ScanNet (Dai et al. 2017), YUD (Denis, Elder, and Estrada 2008), and SVVP (Tong et al. 2022), and to non-Manhattan world dataset such as NYU Depth (Silberman et al. 2012). The SU3 dataset comprises 23k synthetic outdoor images generated by a procedural photo-realistic building generator, with vanishing points computed directly from the CAD models of the generated buildings. The ScanNet dataset comprises 189,916 RGB-D images of real-world indoor scenes for training and 53,193 for validation. Vanishing points in ScanNet are estimated from surface normals following (Zhou et al. 2019a), which makes them less accurate than other datasets. YUD (Denis, Elder, and Estrada 2008) contains 102 images of indoor and outdoor scenes. SVVP (Tong et al. 2022) is a real-world street view dataset that contains 500 images. We also evaluate our model on the non-Manhattan
Table 1: Comparison with SOTA on SU3 and ScanNet. Bold is highest, underlined is second highest, italics is third highest. * in the column of "End-to-End" represents end-to-end trainable, but needs candidate points to be classified in the inference stage.

<table>
<thead>
<tr>
<th>Method</th>
<th>SU3 (Zhou et al. 2019b) A@3°</th>
<th>ScanNet (Dai et al. 2017) A@3°</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>J-linkage</td>
<td>69.2 77.0 84.4</td>
<td>27.8 41.7 57.7</td>
<td>1.2</td>
</tr>
<tr>
<td>Simon et al. (2018)</td>
<td>70.2 77.9 85.1</td>
<td>25.7 39.9 56.6</td>
<td>0.6</td>
</tr>
<tr>
<td>Wu et al. (2021)</td>
<td>74.8 79.5 83.9</td>
<td>22.9 36.8 54.0</td>
<td>23  ×</td>
</tr>
<tr>
<td>Lu et al. (2017)</td>
<td>81.4 87.8 93.0</td>
<td>35.6 53.2 71.6</td>
<td>25  ×</td>
</tr>
<tr>
<td>Li et al. (2019b)</td>
<td>59.1 66.9 74.6</td>
<td>35.0 50.2 66.9</td>
<td>25  ×</td>
</tr>
<tr>
<td>CON SAC (Kluger et al. 2020)</td>
<td>77.9 85.2 91.0</td>
<td>31.1 46.1 62.4</td>
<td>2   ×</td>
</tr>
<tr>
<td>TLC (Tong et al. 2022)</td>
<td>91.3 94.6 97.1</td>
<td>36.2 53.9 72.6</td>
<td>25  ×</td>
</tr>
<tr>
<td>NeurVPS (Zhou et al. 2019a)</td>
<td>94.4 96.5 98.2</td>
<td>36.1 54.9 74.9</td>
<td>0.5  *</td>
</tr>
<tr>
<td>Lin et al. (2022)</td>
<td>84.0 90.3 95.1</td>
<td>32.9 52.0 72.7</td>
<td>5.5  *</td>
</tr>
<tr>
<td>ours</td>
<td>92.8 95.7 97.8</td>
<td>36.8 55.4 75.6</td>
<td>16  √</td>
</tr>
<tr>
<td>ours*</td>
<td>96.4 97.8 98.9</td>
<td>41.7 60.0 78.7</td>
<td>16  √</td>
</tr>
</tbody>
</table>

Table 2: Comparison with SOTA on YUD and SVVP. All learning-based models are pre-trained on SU3.

<table>
<thead>
<tr>
<th>Method</th>
<th>SVVP (Tong et al. 2022) A@3°</th>
<th>YUD (Denis, Elder, and Estrada 2008) A@3°</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>J-linkage</td>
<td>32.8 45.7 60.2</td>
<td>40.2 50.5 64.1</td>
<td>1.2</td>
</tr>
<tr>
<td>Simon et al. (2018)</td>
<td>45.4 59.6 73.2</td>
<td>40.1 58.2 77.5</td>
<td>0.6</td>
</tr>
<tr>
<td>Wu et al. (2021)</td>
<td>39.1 52.4 67.9</td>
<td>44.3 61.4 77.4</td>
<td>23</td>
</tr>
<tr>
<td>Lu et al. (2017)</td>
<td>48.5 64.8 80.0</td>
<td>58.0 73.2 86.2</td>
<td>25</td>
</tr>
<tr>
<td>Li et al. (2019b)</td>
<td>39.3 53.0 66.8</td>
<td>51.1 66.1 80.5</td>
<td>25</td>
</tr>
<tr>
<td>CON SAC (Kluger et al. 2020)</td>
<td>43.8 56.5 69.4</td>
<td>62.1 73.7 84.1</td>
<td>2</td>
</tr>
<tr>
<td>TLC (Tong et al. 2022)</td>
<td>51.6 67.7 82.6</td>
<td>65.5 77.1 87.4</td>
<td>25</td>
</tr>
<tr>
<td>NeurVPS (Zhou et al. 2019a)</td>
<td>27.1 40.4 55.3</td>
<td>39.9 50.3 65.0</td>
<td>0.5</td>
</tr>
<tr>
<td>Lin et al. (2022)</td>
<td>32.9 52.0 72.7</td>
<td>60.7 74.3 86.9</td>
<td>5.5</td>
</tr>
<tr>
<td>ours</td>
<td>41.6 60.3 78.9</td>
<td>42.3 61.4 80.3</td>
<td>16</td>
</tr>
<tr>
<td>ours*</td>
<td>66.9 83.3 91.6</td>
<td>69.1 81.3 90.7</td>
<td>16</td>
</tr>
</tbody>
</table>

Evaluation. Following (Zhou et al. 2019a; Tong et al. 2022), we use 500 images for evaluation on SU3 and ScanNet. And on the SVVP and YUD datasets, since the number of images is too small, we use the model pre-trained on SU3. On these datasets, we calculate the percentage of predictions with an angle difference smaller than a certain threshold and compare the angle accuracy (AA) across different thresholds, as done in previous works (Zhou et al. 2019a; Liu, Zhou, and Zhao 2021; Tong et al. 2022; Lin et al. 2022). For the NYU Depth dataset, we follow (Kluger et al. 2020; Lin et al. 2022), which rank the predicted vanishing points by confidence and then use bipartite matching to compute angular errors for the top-k predictions, then we generate the recall curve and calculate the area under the curve (AUC) up to a specified threshold, such as 10 degrees.

In the inference stage, we first normalize the output value. For the general dataset, we select the prediction with confidence greater than a certain threshold as the output. For the Manhattan World dataset, we select the prediction with the highest score among all predictions as the first result. And as the score decreases, compare the predicted value with the existing results, and select the inner product smaller than a certain threshold such as 0.01 to add to the result until three vanishing points are selected as the final output. In addition, we provide ours* as the results of using 400 queries for prediction and bipartite graph matching for selection.

Implementation details. We implement our model on Nvidia RTX2080Ti for a fair comparison of inference speed. \( \lambda_{reg} \) in Eq. 18 and \( \lambda_{cross} \) in Eq. 19 is set to 5, and \( \lambda_{cls} \) and \( \lambda_{orth} \) is set to 1. ResNet-50 (He et al. 2016) pre-trained on ImageNet (Deng et al. 2009) is used as the backbone. Specifically, we use the C3 layer of the multi-scale feature map as the input of the Transformer, which will be discussed in ablation study. The number of queries is set as 20. By default, models are trained for 220 epochs and the learning rate is decayed at the 200-th epoch by a factor of 0.1. We trained our model using AdamW (Loshchilov and Hutter 2017) with base learning rate of \( 5 \times 10^{-5} \), and weight decay of \( 10^{-4} \).

Comparison with State-of-the-Arts
We compare our model with J-Linkage (Tardif 2009), Simon et al. (Simon, Fond, and Berger 2018), Wu et al. (Wu et al. 2021), Lu et al. (Lu et al. 2017), Li et al. (Li et al. 2021), and others. We find that our model achieves comparable or better results on all datasets, especially on NYU Depth and ScanNet, where it outperforms existing methods.
Datasets NYU Depth (Silberman et al. 2012)

<table>
<thead>
<tr>
<th></th>
<th>top-k=#gt</th>
<th>top-k=#pred</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>@5°</td>
<td>@10°</td>
</tr>
<tr>
<td></td>
<td>@5°</td>
<td>@10°</td>
</tr>
<tr>
<td>J-Linkage</td>
<td>49.30</td>
<td>61.28</td>
</tr>
<tr>
<td></td>
<td>54.48</td>
<td>68.34</td>
</tr>
<tr>
<td>T-Linkage</td>
<td>43.38</td>
<td>58.05</td>
</tr>
<tr>
<td></td>
<td>47.48</td>
<td>64.59</td>
</tr>
<tr>
<td>CONSAC</td>
<td>49.46</td>
<td>65.00</td>
</tr>
<tr>
<td></td>
<td>54.37</td>
<td>69.89</td>
</tr>
<tr>
<td>CONSAC+DLSD</td>
<td>46.78</td>
<td>61.06</td>
</tr>
<tr>
<td></td>
<td>49.94</td>
<td>65.96</td>
</tr>
<tr>
<td>VaPiD</td>
<td>–</td>
<td>69.10</td>
</tr>
<tr>
<td>Lin et al.</td>
<td>55.92</td>
<td>69.57</td>
</tr>
<tr>
<td></td>
<td>57.19</td>
<td>71.62</td>
</tr>
<tr>
<td>Ours</td>
<td>54.56</td>
<td>72.32</td>
</tr>
<tr>
<td></td>
<td>65.52</td>
<td>80.81</td>
</tr>
</tbody>
</table>

Table 3: Non-Manhattan scenario. Here “top-k=#gt” indicates the k most confident predictions are used for evaluation, where k is the number of ground-truth vanishing points. For “top-k=#pred” all predictions are used for evaluation.

Comparison results show that our methods achieve comparable or better performance than SOTA methods on SU3 (Zhou et al. 2019b), ScanNet (Dai et al. 2017), SVVP (Tong et al. 2022), and NYU Depth (Silberman et al. 2012) benchmarks, and the inference speed is 16 FPS, achieving a better balance of accuracy and speed. Furthermore, we also show the angle accuracy curves on SU3 and ScanNet for detail comparison in Figure 3 and Figure 4.

Table 4: Network structure and loss function ablation study, using the last layer feature map of ResNet-50.

Ablation Study
To verify the effectiveness of our design, we conduct extensive ablation experiments on the SU3 dataset. As shown in Table 4, our method outperforms vanilla DETR under the same setting. Adding the orthogonal loss and the cross-prediction loss can effectively improve the performance of the model, which is effective for both vanilla DETR and our VPDETR. And in Table 5, we tested the effect of using different levels of features on the model, and finally chose the C3 level of features. As discussed by TLC (Tong et al. 2022), predicting vanishing points requires both semantic and geometric information, the C3 level achieves a better balance. The resolution of C2 and C4 level is too low to lose too much geometric information. And experiments prove that using multi-scale features does not improve the accuracy.

Visualization
In order to study which regions of the input image our method pays more attention to for giving the final prediction, we draw the gradient norm of the final prediction with respect to each pixel in the image. The gradient norm reflects how much the result will change when a pixel is perturbed. So it can show us which pixels the model relies more on to...
As shown in Figure 5, in order to predict vanishing points, our model will pay more attention to places rich in straight line information, such as road markings, edges of road tiles, outlines of buildings, edges of furniture, corners, etc. This is consistent with the focus of traditional methods, which is to use the clues of the straight line to infer the vanishing point, but our method does not need to explicitly extract the straight line, but through training the model to automatically focus on the area with rich straight line information. Through observation, we found that our model may pay attention to some information that we usually don’t notice, such as the texture of the road surface and the table.

In addition, we also show the position relationship between the predicted vanishing point and the ground truth in the Figure 5.

### Conclusion

This paper proposes VPDETR, a novel end-to-end framework for vanishing point detection that is capable of handling both Manhattan and non-Manhattan world datasets. To improve model accuracy for Manhattan world datasets, we introduce an orthogonal loss and a cross-prediction loss. Our method achieves a good balance of accuracy and inference speed. Our method has limitations, such as the general performance on the YUD dataset due to the domain gap problem. We hope that our method can serve as a new paradigm of vanishing point detection, and inspire thinking about how to infer vanishing points of an image end-to-end, accurately, and quickly.

### Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) (No. 62371009 and No. 61971008).
References
Li, H.; Xing, Y.; Zhao, J.; Bazin, J.-C.; Liu, Z.; and Liu, Y.-H. 2019a. Leveraging structural regularity of Atlanta world for monocular SLAM. In 2019 International Conference on Robotics and Automation, 2412–2418. IEEE.


