arXiv:2101.01909v1 [cs.CV] 6 Jan 2021

Line Segment Detection Using Transformers without Edges

Yifan Xu[*], Weijian Xu[*], David Cheung, Zhuowen Tu
UC San Diego
{yix081,wex041,d6cheung,ztu}@ucsd.edu

Abstract

In this paper, we present a holistically end-to-end algorithm for line segment detection with transformers that is post-processing and heuristics-guided intermediate processing (edge/junction/region detection) free. Our method, named LinE segment TRansformers (LETR), tackles the three main problems in this domain, namely edge element detection, perceptual grouping, and holistic inference by three highlights in detection transformers (DETR) including tokenized queries with integrated encoding and decoding, self-attention, and joint queries respectively. The transformers learn to progressively refine line segments through layers of self-attention mechanism skipping the heuristic design in the previous line segmentation algorithms. We equip multi-scale encoder/decoder in the transformers to perform fine-grained line segment detection under a direct end-point distance loss that is particularly suitable for entities such as line segments that are not conveniently represented by bounding boxes. In the experiments, we show state-of-the-art results on Wireframe and YorkUrban benchmarks. LETR points to a promising direction for joint end-to-end detection of general entities beyond the standard object bounding box representation.

1. Introduction

Line segment detection is an important mid-level visual process [19] for solving various downstream computer vision tasks including 3D reconstruction, segmentation, image matching and registration, depth estimation, scene understanding, object detection, image editing, and shape analysis. Despite its practical and scientific importance, line segment detection remains an unsolved problem in computer vision.

Although dense pixel-wise edge detection has achieved an impressive performance [28], further extracting line segments of semantic and perceptual significance remains a challenging task. In natural scenes, line segments of interest are often having heterogeneous structures, under the background clutter, locally ambiguous, and partially occluded. Morphological operators [23] operated on detected edges [3] give sub-optimal results. Mid-level representations such as Gestalt law [9] and contextual information [24] play an important role. Deep learning techniques [15, 17, 13, 28] have greatly expedited the development of line segments detection algorithms [33, 29] that become increasingly feasible in real world applications. However, systems like [33, 29, 30] still carry the basic heuristics-guided modules as those in the classical methods [23] such as edge/junction/region detection, line grouping, and post processing, limiting the performance and the further development for these methods.

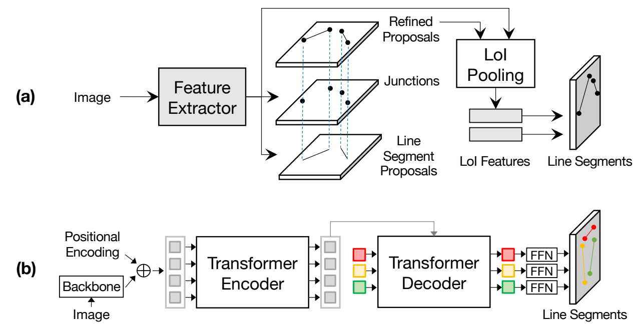

Figure 1. Pipeline comparison between: (a) holistically-attracted wireframe parsing (HAWP) [30] and (b) our LinE segment TRansformers (LETR). Our proposed LETR is a general-purpose approach for line segment detection without heuristics-driven intermediate stages for edge and junction proposal generation.

In this paper, we skip the traditional edge/junction/region detection + proposals + perceptual grouping pipeline by designing a general-purpose holistically end-to-end line segment detection algorithm. We are motivated by the following observations for the transformer framework [25, 4]: tokenized queries with integrated encoding and decoding, self-attention, and joint Hungarian matching implicitly cover the important aspects in line segment detection (edge element detection, perceptual grouping, and holistic estimation) but under a more general-purpose way with greater reasoning capability. Our system, named LinE segment TRsformer (LETR), enjoys the benefit of sharing modeling power with the general-purpose transformer architecture while having

[*]Authors contributed equally.

its enhanced property for line segment detection. LETR is built on top of the seminal work, end-to-end object detector with transformers (DETR) [4]. However, as shown in the ablation study, directly applying the DETR object detector [4] to the line segment detection will no yield satisfactory results since line segments are elongated and not feasible for the bounding box representation.

Our contributions are summarized as follows:

- We cast the line segment detection problem in a holistically end-to-end fashion without explicit edge/junction/region detection and heuristics-guided perceptual grouping processes, which is in distinction to the existing literature in this domain. We achieve state-of-the-art results on the Wireframe [14] and YorkUrban benchmarks [5].
- We perform line segment detection using transformers, specifically based on DETR [4], to realize tokenized entity modeling, perceptual grouping, and holistic (set-based) joint detection via integrated encoder-decoder, self-attention, and joint query inference in transformers.
- We introduce two new algorithmic aspects to DETR [4]: first, a multi-scale encoder/decoder as shown in Figure 2; second, a direct end-point distance loss when training the transformer, allowing entities like line segments to be directly learned and detected that are not feasible in the standard DETR bounding box based representation.

2. Related Works

2.1. Line Segment Detection

Traditional Approaches. Line detection has a long history in computer vision. Early pioneer works rely on low-level cues from pre-defined features (e.g. image gradients). Typically, line segment detection performs edge detection [3, 7, 28], followed by a perceptual grouping [12, 23, 9] process. Classic *perceptual grouping* frameworks [2, 1, 21, 18, 26] aggregate the low-level cues to form line segments in a bottom-up fashion: An image is partitioned into line-support regions by grouping similar pixel-wise features. Line segments are then approximated from line-support regions and filtered by a validation step to remove false positives. Another popular series of line segment detection approaches are based on *Hough transform* [8, 12, 20, 11] to gather local cues: The pixel-wise feature map of the image is converted to a parameter map, in which each position is associated with a unique line. The points on the parameter map that accumulate most votes from the input pixels are identified as line predictions. However, due to the limitation of the local and fixed features, these traditional approaches are sub-optimal on complex and unseen scenes.

Deep Learning Based Approaches. The recent surge of deep learning based approaches has achieved remarkable performance on the line segment detection problem [14, 29, 33, 32, 30] with the use of learnable features to capture extensive context information. One line of approaches detects line segments with *junction-based pipelines*: Deep Wireframe Parser (DWP) [14] creates two parallel branches to predict the junction heatmap and the line heatmap, followed by a merging procedure. Motivated by [22], L-CNN [33] simplifies [14] into a unified network: First, a junction proposal module produces the junction heatmap and then converts detected junctions into line proposals. Second, a line verification module classifies proposals and removes unwanted false positive lines. Methods like [33] are end-to-end but they are at the instance-level (for detecting the individual line segments). Our LETR, like DETR [4], has a general-purpose architecture that is trained in a holistically (set-based) end-to-end fashion. PPGNet [32] proposes to create a point-set graph with junctions as vertices and model line segments as edges. However, the aforementioned approaches are heavily dependent on the high quality junction detection, which is error-prone to various imaging conditions and complex scenarios.

Another line of approaches employs *dense prediction* to obtain a surrogate representation map, and applies a post-process procedure to extract line segments: AFM [29] proposes an attraction field map as an intermediate representation that contains 2-D projection vectors pointing to associated lines. A squeeze module then recovers vectorized line segments from the attraction field map. Despite of a relatively simpler design, [29] demonstrates inferior performance compared with junction-based approaches. Recently, HAWP [30] builds a hybrid model of AFM [29] and L-CNN [33] by computing line segment proposals from the attraction field map and then refining proposals with junctions before further line verification.

In contrast, as shown in Figure 1, our approach differs from previous methods by removing heuristics-driven intermediate stages such as edges, junctions, regions, proposals and surrogate dense prediction maps. Instead, our approach is able to directly predict vectorized line segments while keeping superb performance under a general purpose framework.

2.2. Transformer Architecture

Transformers [25] have achieved great success in the natural language processing field and become a *de facto* standard backbone architecture for language models [6]. It introduces self-attention and cross-attention modules as basic building blocks for modeling dense relations among elements of input sequence. These attention-based mechanisms also benefit many vision tasks such as video classification [27], semantic segmentation [10], image generation [31], etc. Recently, end-to-end object detection with transformers (DETR) [4] reformulates the object detection pipeline with transformers by eliminating the need of hand-crafted anchor boxes and non-maximum suppression steps. Instead, [4] proposes to

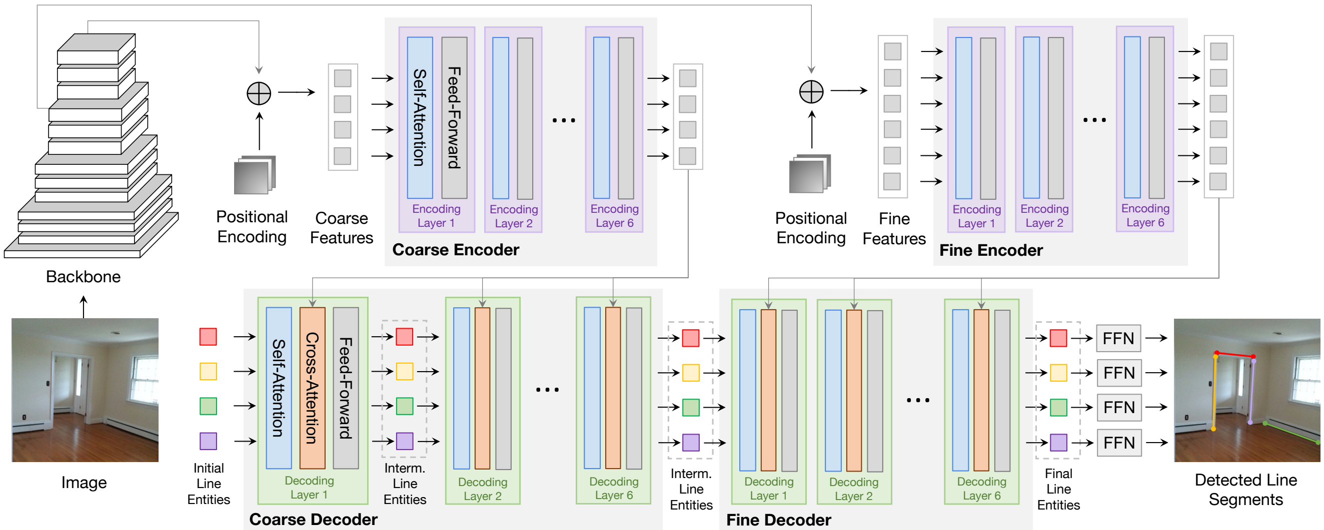

Figure 2. Schematic illustration of our LETR pipeline: An image is fed into a backbone network and generates two feature maps, which are then used by the coarse and the fine encoder respectively. Afterwards, initial line entities are first refined by the coarse decoder with interaction of the coarse encoder output, and then the intermediate line entities from coarse decoder are further refined by the fine decoder attending to the fine encoder. Finally, line segments are detected by feed-forward networks (FFNs) on top of line entities.

feed a set of object queries into the encoder-decoder architecture with interaction from the image feature sequence and generate a final set of predictions. A bipartite matching objective is then optimized to force unique assignments between predictions and targets.

We introduce two aspects to DETR [4] when realizing our LETR: 1) multi-scale encoder and decoder; 2) direct distance loss for the line segments.

3. Line Segment Detection with Transformers

3.1. Motivation

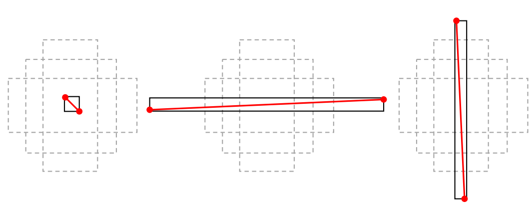

Figure 3. **Bounding Box Representation.** Three hard cases to represent the line segment as a diagonal of bounding box. Red lines are line segments, black boxes are corresponding bounding boxes and gray dotted boxes are anchors.

Despite the exceptional performance achieved by the recent deep learning based approaches [33, 29, 30] on line segment detection, their pipelines still involve heuristics-driven intermediate representations such as junctions and attraction field maps, raising an interesting question: *Can we directly model all the vectorized line segments with a neural network?* A naive solution could be simply regarding the line segments as *objects* and building a pipeline following the standard object detection approaches [22]. Since the location of 2-D objects is typically parameterized as a bounding box, the vectorized line segment can be directly read from a diagonal of the bounding box associated with the line segment object. However, the limited choices of anchors make it difficult for standard two-stage object detectors to predict very short line segments or line segments nearly parallel to the axes (see Figure 3). The recently appeared DETR [4] eliminates the anchors and the non-maximum suppression, perfectly meets the need of line segment detection. However, the vanilla DETR still focuses on bounding box representation with a GIoU loss. We further convert the box predictor in DETR into a vectorized line segment predictor by adapting the losses and enhancing the use of multi-scale features in our designed model.

3.2. Overview

In a line segment detection task, a parser aims to predict a set of line segments from given images. Performing line segmentation detection with transformer removes the need of explicit edge/junction/region detection [33, 30] (see Figure 1). Our LETR is built purely based on the transformer encoder-decoder structure. The proposed line segment detection process consists of four stages:
(1) *Image Feature Extraction*: Given an image input, we obtain the image feature map $\mathbf{x} \in \mathbb{R}^{H \times W \times c}$ from a CNN backbone with reduced dimension. The image feature is concatenated with positional embeddings to obtain spatial relations. (2) *Image Feature Encoding:* The flattened feature map $\mathbf{x} \in \mathbb{R}^{HW \times c}$ is then encoded to $\mathbf{x}' \in \mathbb{R}^{HW \times c}$ by

Figure 4. **Line Entity Representation.** For each row, we show how the same line entity predicts line segments with same property in three different indoor/outdoor scene. The top line entity is specialized for horizontal line segments in the middle of the figure, and the bottom one prefers to predict vertical line segments with a various range of lengths.

multi-head self-attention module and feed forward networks following the standard transformer encoding architecture. (3) *Line Segment Detection:* In the transformer decoder networks, N learnable line entities $\mathbf{l} \in \mathbb{R}^{N\times c}$ interact with encoder output via the cross attention module. (4) *Line Segment Prediction:* Line entities make line segment predictions with two prediction heads built on top of the transformer decoder. The line coordinates are predicted by a multi-layer perceptron (MLP), and the prediction confidence are scored by a linear layer.

Self-Attention and Cross-Attention: We first visit the scaled dot-product attention popularized by Transformer architectures [25]. The basic scaled dot-product attention consists of a set of m query $Q \in \mathbb{R}^{m\times d}$, a set of n key-value pairs notated as a key matrix $K \in \mathbb{R}^{n\times d}$ and a value matrix $V \in \mathbb{R}^{n\times d}$. Here we set Q, K, V to have same feature dimension d.

For each $q \in \mathbb{R}^{1\times d}$ belongs to Q, we calculate its dot product with all keys K, and apply a softmax after divide it by $\sqrt{d}$ to obtain the attention weights on each value V. The attention operation F is defined as:

$$F = \text{Att}(q, K, V) = \text{softmax}(\frac{qK^T}{\sqrt{d}})V \tag{1}$$

In practice, multi-head attention is adopted to model the complex feature relation which consists of h paralleled attention operation F described above:

$$\begin{aligned} F_{\text{MHA}} &= \text{MultiHeadAtt}(q, K, V) = [F_1, ..., F_H]W^o \\ F_h &= \text{Att}(qW_h^Q, KW_h^K, VW_h^V) \end{aligned} \tag{2}$$

Where W_h^Q, W_h^K, $W_h^V \in \mathbb{R}^{d\times d'}$ and $W^o \in \mathbb{R}^{d\times d}$ are the projection matrices adopted in each head. The output dimension of each output feature d' is set to d/h in practice to keep the dimension consistent. For a set of m query $Q \in \mathbb{R}^{m\times d}$, we obtain $F_{\text{MHA}} \in \mathbb{R}^{m\times d}$ following the similar manner of Eq.(2).

In our encoder-decoder transformer architecture, we adopt two attention modules based on the multi-head attention, namely the self-attention module (SA) and cross-attention (CA) module (see Figure 2). The SA module takes in a set of input embeddings notated as $\mathbf{x} = [x_1, ..., x_i] \in \mathbb{R}^{i\times d}$, and outputs a weighted summation $\mathbf{x}' = [x'_1, ..., x'_i] \in \mathbb{R}^{i\times d}$ of input embeddings within $\mathbf{x}$ following Eq.2 where $F = \text{Att}(Q = \mathbf{x}, K = \mathbf{x}, V = \mathbf{x})$. The CA module takes in two sets of input embeddings notated as $\mathbf{x} = [x_1, ..., x_i] \in \mathbb{R}^{i\times d}$, $\mathbf{z} = [x_1, ..., x_j] \in \mathbb{R}^{j\times d}$ following Eq.2 where $F = \text{Att}(Q = \mathbf{z}, K = \mathbf{x}, V = \mathbf{x})$.

Transformer Encoder is stacked with multiple encoder layers. Each encoder layer takes in image features $\mathbf{x} \in \mathbb{R}^{HW\times c}$ from its predecessor encoder layer, and processes it with a SA module to learn the pairwise relation. The output features from SA module are passed into a feed-forward layer (FC) with activation and dropout layer followed by another feed-forward (FC) layer. Layer norm is applied between SA module and first FC layer and after second FC layer. To facilitate optimization of deep layers, residual connection is added before the first FC layer and after the second FC layer.

Transformer Decoder is stacked with multiple decoder layers. Each decoder layer takes in a set of image features $\mathbf{x}' \in \mathbb{R}^{HW\times c}$ from the last encoder layer, and a set of line entities $\mathbf{l} \in \mathbb{R}^{N\times c}$ from its predecessor decoder layer. The line entities are first processed with a SA module, each line entity $l \in \mathbb{R}^{1\times c}$ in $\mathbf{l}$ attends to different regions of image feature embeddings $\mathbf{x}'$ via the CA module. FC layers and other modules are added into the pipeline similar to the Encoder setting above.

Line Entity Interpretation. The *line entities* are analogous with the *object queries* in DETR [4]. All line entities $\mathbf{l}$ are learnable embeddings and work together to build line segments holistically. We found each line entity has its own preferred existing region, length, and rotation degree of potential line segment after the training process (shown in Figure 4). We find the holistic predictions from line entities are more accurate compared with edge/junction/region detection + proposals + perceptual grouping pipelines when encountering heterogeneous line segment structures in Section 4.4 and Figure 5.

3.3. Coarse-to-Fine Strategy

Different from object detection, line segment detection requires the detector to consider the local fine-grained details of line segments with the global indoor/outdoor structural together. In our LETR architecture, we propose a coarse-to-fine strategy to predict line segments in a refinement process. The process allows line entities to make precise prediction with the interaction of multi-scale encoded features, while having the capability to aware holistic architecture with the communication to other line entities. During the coarse decoding stage, our line entities attend to potential line segment regions, often unevenly distributed, with a low resolution. During the fine decoding stage, our line entities produce detailed line segment predictions with a high resolution (see Figure 2). After each decoding layer at both coarse and fine decoding stages, we require line entities to make predictions through two shared prediction heads to progressively improve the prediction quality.

Coarse Decoding. During the coarse decoding stage, we pass image features and line entities into an encoder-decoder transformer architecture. The encoder receives coarse features from the output of Conv5 (C5) from ResNet with $\frac{1}{32}$ original resolution. Then, line entity embeddings attend to coarse features from the output of the encoder in the cross-attention module at each layer. After multiple decoding stages, the line entities obtain holistic awareness of line segment density in the scene. The coarse decoding stage is necessary for success at fine decoding stage and its high efficiency with less memory and less computation cost.

Fine Decoding. Fine decoder inherits line entities from coarse decoder and high resolution features from the fine encoder. The features to the fine encoder come from output of Conv4 (C4) from ResNet with $\frac{1}{16}$ original resolution. The line entity embeddings decode feature information in the same manner as the coarse decoding stage.

3.4. Line Segment Prediction

In the previous decoding procedure, our multi-scale decoders progressively refine N initial line entities to produce same amount final line entities. In the prediction stage, each final entity l will be fed into feed-forward networks (FFN), which consist of a classifier module to predict the confidence $\hat{p}$ of being a line segment, and a regression module to predict the coordinates of two end points $\hat{\mathbf{p}}_1 = (\hat{x}_1, \hat{y}_1)$, $\hat{\mathbf{p}}_2 = (\hat{x}_2, \hat{y}_2)$ that parameterizes the associated line segment $\hat{\mathbf{L}} = (\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2)$.

Bipartite Matching. Generally, there are much more line entities provided than the actual line segments in the image. Thus, during *training* stage, we conduct a set-based bipartite matching between line segment predictions and ground-truth targets to determine whether the prediction is associated with an existing line segment or not: Assume there are N line segment predictions $\{(p^{(i)}, \hat{\mathbf{L}}^{(i)}); i = 1, ..., N\}$ and M targets $\{\mathbf{L}^{(j)}; j = 1, ..., M\}$, we optimize a bipartite matching objective on a permutation function $\sigma(\cdot)$ of indices $\{1, ..., N\}$:

$$\mathcal{L}_{\text{match}} = \sum_{i=1}^{N} \mathbb{1}_{\{\sigma(i) \leq M\}} \left[\lambda_1 d(\hat{\mathbf{L}}^{(i)}, \mathbf{L}^{(\sigma(i))}) - \lambda_2 \hat{p}^{(i)}\right] \quad (3)$$

where $d(\cdot, \cdot)$ represents L_1 distance between coordinates and $\mathbb{1}_{\{\cdot\}}$ is an indicator function. $\mathcal{L}_{\text{match}}$ takes both distance and confidence into account with balancing coefficients λ_1, λ_2. The optimal permutation σ^* is computed using a Hungarian algorithm, mapping M positive prediction indices to target indices $\{1, ..., M\}$. During the *inference* stage, we filter the N line segment predictions by setting a fixed threshold on the confidence $\hat{p}^{(i)}$ if needed due to no ground-truth provided.

3.5. Line Segment Losses

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

Classification Loss. Based on a binary cross-entropy loss, we observe that hard examples are less optimized after learning rate decay and decide to apply adaptive coefficients inspired by focal loss [16] to the classification loss term $\mathcal{L}_{\text{cls}}$:

$$\mathcal{L}_{\text{cls}}^{(i)} = -\mathbb{1}_{\{\sigma^*(i) \leq M\}} \alpha_1 (1 - p^{(i)})^{\gamma} \log p^{(i)} \quad (4)$$
$$- \mathbb{1}_{\{\sigma^*(i) > M\}} \alpha_2 {p^{(i)}}^{\gamma} \log(1 - p^{(i)}) \quad (5)$$

Distance Loss. We compute a simple L_1-based distance loss for line segment end-point regression:

$$\mathcal{L}_{\text{dist}}^{(i)} = \mathbb{1}_{\{\sigma^*(i) \leq M\}} d(\hat{\mathbf{L}}^{(i)}, \mathbf{L}^{(\sigma^*(i))}) \quad (6)$$

where $d(\cdot, \cdot)$ represents the sum of L_1 distances between prediction and target coordinates. The distance loss is only applied to the positive predictions. Note that we remove the GIoU loss from [4] since GIoU is mainly designed for the similarity between bounding boxes instead of line segments. Thus, the final loss $\mathcal{L}$ of our model is formulated as:

$$\mathcal{L} = \sum_{i=1}^{N} \lambda_{\text{cls}} \mathcal{L}_{\text{cls}}^{(i)} + \lambda_{\text{dist}} \mathcal{L}_{\text{dist}}^{(i)} \quad (7)$$

4. Experiments

4.1. Datasets

We train and evaluate our model on the ShanghaiTech *Wireframe* dataset [14], which consists of 5000 training images and 462 testing images. We also evaluate our model on the *YorkUrban* dataset [5] with 102 testing images from both indoor scenes and outdoor scenes.

Through all experiments, we conduct data augmentations for the training set, including random horizontal/vertical flip, random resize, random crop, and image color jittering. At the training stage, we resize the image to ensure the shortest size is at least 480 and at most 800 pixels while the longest size is at most 1333. At evaluation stage, we resize the image with the shortest side at least 1100 pixels.

4.2. Implementation

Networks. We experience both ResNet-50 and ResNet-101 as our feature backbone. For an input image $X \in \mathbb{R}^{W_0 \times H_0 \times 3}$, the coarse encoder takes in the feature map from the Conv5 (C5) layer of ResNet backbone with resolution $\mathbf{x} \in \mathbb{R}^{W \times H \times c}$ where $c = 2048, H = \frac{H_0}{32}, W = \frac{W_0}{32}$.

(a) AFM [29] (b) LCNN [33] (c) HAWP [30] (d) LETR (ours) (e) Ground-Truth

Figure 5. **Qualitative Evaluation of Line Detection Methods. From left to right:** the columns are the results from AFM [29], LCNN [33], HAWP [30], LETR (ours) and the ground-truth. **From top to bottom:** the top two rows are the results from Wireframe test set, and the bottom two rows are the results from YorkUrban test set.

The fine encoder takes in higher resolution feature map ($c = 1024, H = \frac{H_0}{16}, W = \frac{W_0}{16}$) from the from the Conv4 (C4) layer of ResNet. Feature maps are reduced to 256 channels by a 1x1 convolution and are fed into the transformer along with the *sine/cosine* positional encoding. Our coarse-to-fine strategy consists two independent encoder-decoder structures processing multi-scale image features. Each encoder-decoder structure is constructed with 6 encoder and 6 decoder layers with 256 hidden dimension and 8 attention heads.

Optimization. We train our model using 4 Titan RTX GPUs through all our experiments. We first train the coarse encoder-decoder until optimal. Then, we freeze the weights in the coarse transformer and initial fine transformer with coarse transformer weights. FFN weights are shared through all decoder layers. We use AdamW as the model optimizer, and set weight decay as 10^{-4}. The initial learning rate is set to 10^{-4}, and is reduced by a factor of 10 every 200 epochs for coarse decoding stage and every 120 epochs for fine prediction stage. We found the large number of line entities is essential to the line segment detection task by experimenting on a wide range of number of line entities (See Figure 8 (Right)), and use 1000 line entities in all reported benchmarks unless specified elsewhere. To mitigate the class imbalance issue, we also reduce the classification weight for background/no-object instances (i.e. unmatched line entities) by a factor of 10.

4.3. Evaluation Metric

We evaluate our results based on two heatmap-based metrics, AP^H and F^H, which are widely used in previous LSD

Method	*Wireframe Dataset*						*YorkUrban Dataset*					
	sAP^{10}	sAP^{15}	sF^{10}	sF^{15}	AP^{H}	F^{H}	sAP^{10}	sAP^{15}	sF^{10}	sF^{15}	AP^{H}	F^{H}
LSD [26]	/	/	/	/	55.2	62.5	/	/	/	/	50.9	60.1
DWP [14]	5.1	5.9	/	/	67.8	72.2	2.1	2.6	/	/	51.0	61.6
AFM [29]	24.4	27.5	/	/	69.2	77.2	9.4	11.1	/	/	48.2	63.3
L-CNN [33]	62.9	64.9	61.3	62.4	82.8	81.3	26.4	27.5	36.9	37.8	59.6	65.3
HAWP [30]	66.5	68.2	64.9	65.9	86.1	83.1	28.5	29.7	39.7	40.5	61.2	66.3
LETR (ours)	65.2	67.7	**65.8**	**66.6**	**86.3**	**83.3**	**29.4**	**31.7**	**40.1**	**41.8**	**62.7**	**66.9**

Table 1. **Comparison to Prior work on Wireframe and YorkUrban benchmarks.** Our proposed LETR reaches state-of-the-art performance except sAP^{10} and sAP^{15} slightly worse than HAWP [30] in Wireframe. Results of prior works are adopted from HAWP paper.

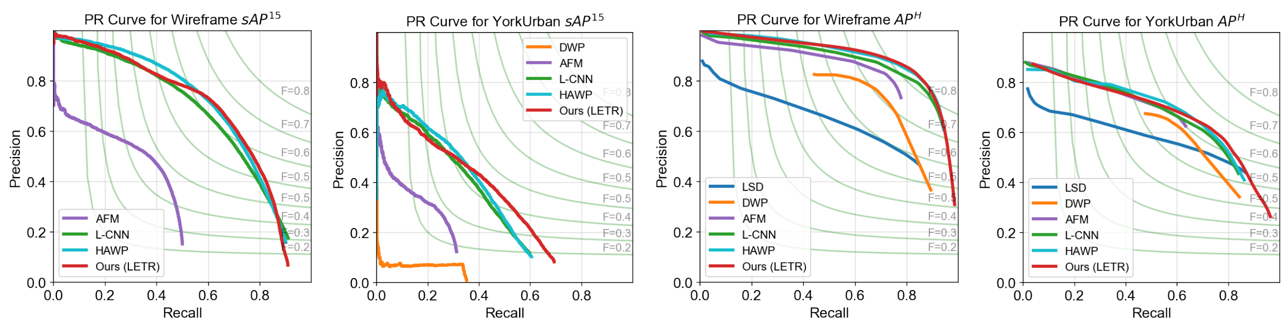

Figure 6. **Precision-Recall (PR) curves.** PR curves of sAP^{15} and AP^{H} for DWP[14], AFM[29], L-CNN[33], HAWP[30] and LETR (ours) on Wireframe and YorkUrban benchmarks.

task[33, 14], and Structural Average Precision(sAP) which is proposed in L-CNN [33]. On the top of that, we evaluate result with a new metric, Structural F-score(sF), for a more comprehensive comparison.

Heatmap-based metrics, AP^H, F^H: Prediction and ground truth lines are first converted to heatmaps by rasterizing the lines, and we generate the precision-recall curve comparing each pixel along with their confidence. Then we can use the curve to calculate F^{H} and AP^{H}.

Structural-based metrics, sAP[33], sF: Given a set of ground truth line and a set of predicted lines, for each ground truth line $\mathbf{L}$, we define a predicted line $\hat{\mathbf{L}}$ to be a match of $\mathbf{L}$ if their L_2 loss is smaller than the pre-defined threshold $\vartheta \in \{10, 15\}$. Over the set of lines matched to $\mathbf{L}$, we select the line with the highest confidence as true-positive, and treat the rest as false-positive. If the set of matching line is empty, we would regard this ground-truth line as false-negative. Each predicted line would be matched to at most one ground truth line, and if a line isn't matched to any ground truth line, then it is considered as false-positive. The matching is recomputed at each confidence level to produce the precision-recall curve, and we consider sAP as the area under this curve. Considering F^{H} as the complementary F-score measurement for AP^{H}, we evaluate the F-score measurement for sAP, denoted as sF, to be the best balanced performance measurement.

4.4. Results and Comparisons

We summarize quantitative comparison results between LETR and previous line segment detection methods in Table 1. We report results for LETR with ResNet-101 backbone for Wireframe dataset, and results with ResNet-50 backbone for York dataset. Our LETR achieves new state-of-the-art for all evaluation metrics on YorkUrban Dataset[5]. In terms of heatmap based evaluation metric, our LETR is consistently better than other models for both benchmarks, and outperforms HAWP [30] for 1.5% for AP^{H} on YorkUrban Dataset. We show PR curve comparison in Figure 6 on sAP^{15} and AP^{H} for both Wireframe[14] and YorkUrban benchmarks. In Figure 6, we notice the current limitation of LETR comes from lower precision prediction when we include fewer predictions compare to HAWP. When we include all set of predictions, LETR predicts slightly better than HAWP and other leading methods, which matches our hypothesis that holistic prediction fashion can guide line entities to refine low confident predictions (usually due to local ambiguous and occlusion) with high confident predictions.

We also show both Wireframe and YorkUrban line segment detection qualitative results from LETR and other competing methods in Figure 5. The top two rows are indoor scene detection results from Wireframe dataset while the buttom two rows are outdoor scene detection results from YorkUrban dataset. We see more clear difference between LETR's holistic prediction through self-attention and cross-attention and the one from other methods when line segments

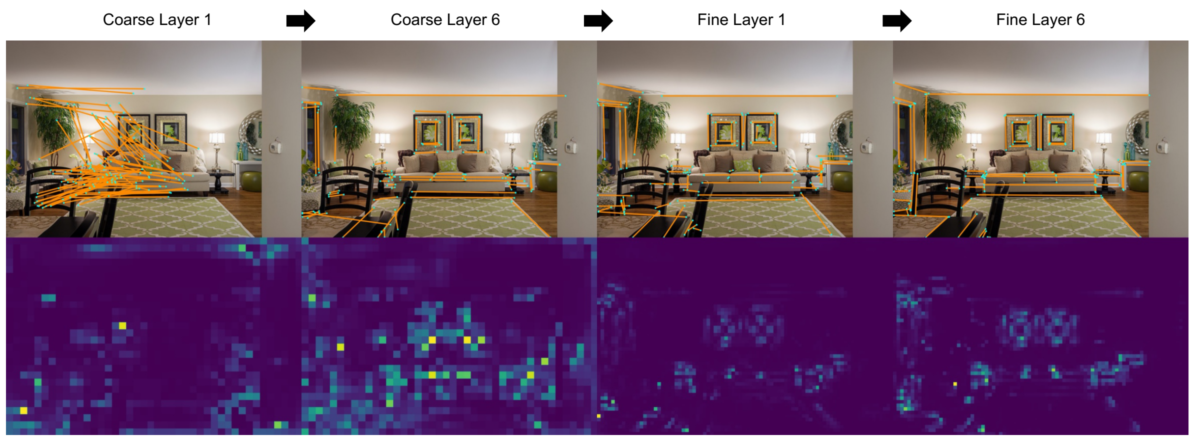

Figure 7. **Visualization of LETR coarse-to-fine decoding process. From top to bottom**: The 1^{st} row shows line segment detection results based on line entities after different layers and the 2^{nd} row shows its corresponding overlay of attention heatmaps. **From left to right**: The 1^{st}, 2^{nd}, 3^{rd}, 4^{th} columns are coarse decoder layer 1, coarse decoder layer 6, fine decoder layer 1, fine decoder layer 6, respectively.

Coarse Decoding	Fine Decoding	Focal Loss	Feature Index	sAP^{10}	sAP^{15}
✓			C5(C)	60.7	63.9
✓			C4(C)	63.8	66.5
✓	✓		C5(C), C4(F)	64.5	67.3
✓	✓	✓	C5(C), C4(F)	65.1	67.7

Table 2. **Effectiveness of modules.** The abalation study of architecture design and learning aspects in the proposed LETR on Wireframe dataset. (C) indicates the indexed feature is used for coarse decoder; (F) indicates the indexed feature is used for fine decoder.

Method	sAP^{10}	sAP^{15}	sF^{10}	sF^{15}
Faster R-CNN	38.4	40.7	51.5	53.0
Vanilla DETR	53.8	57.2	57.2	59.0
LETR (ours)	65.1	67.7	65.2	66.6

Table 3. Comparison with object detection approaches on Wireframe dataset.

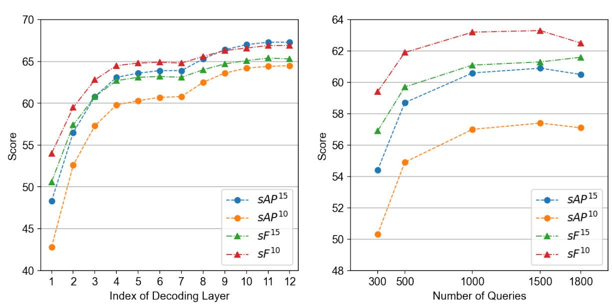

Figure 8. (Left) We evaluate performance for output from each decoding layers. The 1-6 layers are coarse decoder layers and 7-12 layers fine decoder layers. (Right) We test coarse decoder with different number of line entities on Wireframe.

of interest contain **massive predictions** (e.g. building windows in row 3, 4), **paralleled structure** (e.g. eaves and pillars in row 1), and **partially occlusion** (e.g. decorative strip in row 2).

5. Model Analysis

We compare the effectiveness of different modules in LETR in Table 2. During the coarse decoding stage, LETR reaches 60.7 and 63.9 for sAP^{10} and sAP^{15} with encoding features from C5 layer of ResNet backbone, and 63.8 and 66.5 with the one from C4 of ResNet backbone. The fine decoder reaches 64.5 and 67.3 by improving the coarse prediction with fine grained details from high resolution features in row 3. We then address the data imbalance problem with focal loss in row 4.

We compare LETR results with two object detection baseline where the line segments are treated as 2-D objects within this context in Table 3. We see clear limitations for using bounding box diagonal for both Faster R-CNN and DETR responding to our motivation in Section 3.1.

6. Visualization

We study the dynamics of the LETR's coarse-to-fine decoding process in Figure 7. The first two columns are results from coarse decoder receiving decoded features from C5 ResNet layer. While the global structure of the scene is well captured efficiently, the low resolution features prevents it to make predictions precisely. The last two columns are results from fine decoder receiving decoded features from C4 ResNet layer and line entities from coarse decoder. The overlay of attention heatmaps depicts more detailed relationships in the image space, which is the key for the detector performance. This finding is also shown in Figure 8 (Left), where the decoded output after each layer has consistent improvement with the multi-scale encoder-decoder strategy.

7. Conclusion

In this paper, we presented LETR, a line segment detector based on multi-scale encoder/decoder transformer structure.

By casting the line segment detection problem in a holistically end-to-end fashion, we perform detection without explicit edge/junction/region detection and heuristics-guided perceptual grouping processes. A direct end-point distance loss allows line entities to represent and learn line segments beyond bounding-box based representation.

Acknowledgment. This work is funded by NSF IIS-1618477 and NSF IIS-1717431. We thank Feng Han, Ido Durst, Yuezhou Sun, Haoming Zhang, and Heidi Cheng for valuable feedbacks.

References

[1] Michael Boldt, Richard Weiss, and Edward Riseman. Token-based extraction of straight lines. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1581–1594, 1989. 2

[2] J. Brian Burns, Allen R. Hanson, and Edward M. Riseman. Extracting straight lines. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(4):425–455, 1986. 2

[3] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986. 1, 2

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Eur. Conf. Comput. Vis.*, 2020. 1, 2, 3, 4, 5

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

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2

[7] Piotr Dollár, Zhuowen Tu, and Serge Belongie. Supervised learning of edges and object boundaries. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006. 2

[8] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972. 2

[9] James H Elder and Richard M Goldberg. Ecological statistics of gestalt laws for the perceptual organization of contours. *Journal of Vision*, 2(4):5–5, 2002. 1, 2

[10] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019. 2

[11] Yasutaka Furukawa and Yoshihisa Shinagawa. Accurate and robust line segment extraction by analyzing distribution around peaks in hough space. *Computer Vision and Image Understanding*, 92(1):1–25, 2003. 2

[12] Nicolas Guil, Julio Villalba, and Emilio L Zapata. A fast hough transform for segment detection. *IEEE Transactions on Image Processing*, 4(11):1541–1548, 1995. 2

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[14] Kun Huang, Yifan Wang, Zihan Zhou, Tianjiao Ding, Shenghua Gao, and Yi Ma. Learning to parse wireframes in images of man-made environments. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 626–635, 2018. 2, 5, 7

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. *Adv. Neural Inform. Process. Syst.*, 2012. 1

[16] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Int. Conf. Comput. Vis.*, pages 2999–3007, 2017. 5

[17] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015. 1

[18] Xiaohu Lu, Jian Yao, Kai Li, and Li Li. Cannylines: A parameter-free line segment detector. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 507–511. IEEE, 2015. 2

[19] David Marr. Vision: A computational investigation into the human representation and processing of visual information, henry holt and co. *Inc., New York, NY*, 2(4.2), 1982. 1

[20] Jiri Matas, Charles Galambos, and Josef Kittler. Robust detection of lines using the progressive probabilistic hough transform. *Computer vision and image understanding*, 78(1):119–137, 2000. 2

[21] Marcos Nieto, Carlos Cuevas, Luis Salgado, and Narciso García. Line segment detection using weighted mean shift procedures on a 2d slice sampling strategy. *Pattern Analysis and Applications*, 14(2):149–163, 2011. 2

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 2, 3

[23] Stephen M Smith and J Michael Brady. Susan—a new approach to low level image processing. *International journal of computer vision*, 23(1):45–78, 1997. 1, 2

[24] Zhuowen Tu. Auto-context and its application to high-level vision tasks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 1

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 1, 2, 4

[26] R G von Gioi, J Jakubowicz, J M Morel, and G Randall. LSD: A Fast Line Segment Detector with a False Detection Control. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(4):722–732, 2010. 2, 7

[27] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 2

[28] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015. 1, 2

[29] Nan Xue, Song Bai, Fudong Wang, Gui-Song Xia, Tianfu Wu, and Liangpei Zhang. Learning attraction field representation for robust line segment detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 1, 2, 3, 6, 7

[30] Nan Xue, Tianfu Wu, Song Bai, Fudong Wang, Gui-Song Xia, Liangpei Zhang, and Philip HS Torr. Holistically-attracted wireframe parsing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2788–2797, 2020. 1, 2, 3, 6, 7

[31] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning*, pages 7354–7363. PMLR, 2019. 2

[32] Ziheng Zhang, Zhengxin Li, Ning Bi, Jia Zheng, Jinlei Wang, Kun Huang, Weixin Luo, Yanyu Xu, and Shenghua Gao. Ppgnet: Learning point-pair graph for line segment detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 2

[33] Yichao Zhou, Haozhi Qi, and Yi Ma. End-to-end wireframe parsing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 962–971, 2019. 1, 2, 3, 6, 7