Holistically-Attracted Wireframe Parsing: From Supervised to Self-Supervised Learning

Nan Xue, Tianfu Wu, Song Bai, Fu-Dong Wang, Gui-Song Xia, Liangpei Zhang, Philip H.S. Torr

Abstract—This article presents Holistically-Attracted Wireframe Parsing (HAWP), a method for geometric analysis of 2D images containing wireframes formed by line segments and junctions. HAWP utilizes a parsimonious Holistic Attraction (HAT) field representation that encodes line segments using a closed-form 4D geometric vector field. The proposed HAWP consists of three sequential components empowered by end-to-end and HAT-driven designs: (1) generating a dense set of line segments from HAT fields and endpoint proposals from heatmaps, (2) binding the dense line segments to sparse endpoint proposals to produce initial wireframes, and (3) filtering false positive proposals through a novel endpoint-decoupled line-of-interest aligning (EPD LOIAlign) module that captures the co-occurrence between endpoint propsoals and HAT fields for better verification. Thanks to our novel designs, HAWPv2 shows strong performance in fully supervised learning, while HAWPv3 excels in self-supervised learning, achieving superior repeatability scores and efficient training (24 GPU hours on a single GPU). Furthermore, HAWPv3 exhibits a promising potential for wireframe parsing in out-of-distribution images without providing grountruth labels of wireframes.

Index Terms—Wireframe Parsing, Line Segment Detection, Holistic Attraction Field Representation, Self-Supervised Learning

1 Introduction

Depicting image contents with geometric entities/patterns such as salient points, line segments, and planes/surfaces has been shown as an effective encoding scheme of visual information evolved in primate visual systems, which in turn has long motivated the computer vision community to make tremendous efforts on computing the primal sketch [1] of natural images consisting of different forms including, but not limited to, the blobs [2], [3], [4], corners/junctions [4], [5], [6], [7], edges [8], [9], [10], and line segments [11], [12] since the 1960s. Modeling and computing the primal sketches have remained a long-standing problem, and it plays important roles in many downstream tasks including 3D reconstruction [13], [14], [15], [16], [17] and scene parsing [18], [19], as well as high-level visual recognition tasks [20], [21], [22].

In this paper, our focus lies on modeling, learning, and parsing wireframes [25] from images, which represent a parsimonious form of the primal sketch. As depicted in Fig. 1, wireframes capture line segments and their associated endpoints (primarily junctions) in images, enabling vectorized representations of the underlying boundary structures of objects and generic regions (stuff). Despite line segments being simple geometric patterns/symbols by definition, effectively modeling and computing them from images presents an exceedingly challenging problem due to the inherent uncertainty and ambiguity in grounding line segments to image pixels (referred to as the symbol-to-signal gap).

We present a learning-based approach for wireframe parsing (Fig. 5), aiming to bridge the gap between pixels

(a) HAWPv1 on the Wireframe Dataset (FSL Model)

(b) HAWPv2 on the Wireframe Dataset (FSL Model)

(c) HAWPv3 on the Wireframe Dataset (SSL Model)

(d) HAWPv3 on the BSDS Dataset (SSL Model)

(e) HAWPv3 on the AICrowd Dataset (SSL Model)

Fig. 1. The proposed HAWP models excel in wireframe structure perception using both fully-supervised learning (FSL) and self-supervised learning (SSL). HAWPv1 [23] and improved HAWPv2 are FSL models trained with human-annotated wireframes, primarily in indoor images. HAWPv3, an SSL model built on HAWPv2, enables wireframe parsing in out-of-distribution images such as those from the BSDS-500 dataset [8] and AICrowd dataset [24], without requiring labeled wireframes.

- *N. Xue is with Ant Group.*
- *T. Wu is with the Department of ECE, North Carolina State University.*
- *S. Bai is with ByteDance AI Lab.*
- *F.-D. Wang is with Ant Group.*
- *G.-S. Xia and L. Zhang are with Wuhan University.*
- *P. Torr is with the University of Oxford.*

(2D signals) and symbols (line segments and junctions) progressively. Our method involves three key steps: (1) learning a novel Holistic Attraction (HAT) field representation that characterizes the geometry of line segments, including their endpoints, by incorporating both edge and non-edge pixels; (2) binding densely predicted line segments to a reduced set of junction/endpoint proposals, eliminating the need for complex non-maximum suppression (NMS); and (3) achieving wireframe parsing through a proposal verification module. This paper extends our previous work [23] (HAWPv1, published in CVPR'20) with significant modifications in two aspects. Firstly, we address the effective and robust learning of the proposed HAT fields in the fully supervised learning setting, resulting in the improved HAWPv2. Secondly, we broaden the applicability of our HAT fields through self-supervised learning, leading to HAWPv3 as an extension of HAWPv2 for wireframe parsing in diverse scenarios. Notably, our ablation studies in Appx. D demonstrate that HAWPv1 does not support self-supervised learning well. We summarize and discuss the modifications and development path of the HAWP models as follows.

Our proposed HAT field representation stands out from other methods in the literature due to two novel aspects. Firstly, it incorporates a conceptually simple yet expressive line-segment-to-attraction-region lifting, which aligns with the population coding principle [26] observed in primate visual systems and naturally incorporates visual context awareness. Secondly, it features a rigorously formulated closed-form differentiable HAT field parameterization (see Sec. 3 for detailed formulations), promoting representational parsimony and encouraging inferential consistency among the population, i.e., all (foreground) pixels within an attraction region. These aspects contribute to the stability and efficiency of learning HAWP. In fully supervised learning settings, follow-up studies on wireframe parsing [27], [28], [29] without HAT fields typically require hundreds of training epochs to achieve comparable performances to our HAWPv1 [23], which is trained in just 30 epochs. Intriguingly, the combination of these two aspects facilitates a significantly more efficient self-supervised learning paradigm.

In the development of HAWPv2, we explore novel aspects within the fully supervised learning paradigm. We investigate the closed-form property of the HAT field representation and unveil a simple yet effective differentiable loss function, which penalizes the endpoint fitting error in 2D Euclidean space for densely predicted line segments, thereby reducing invalid proposals. To address the densely predicted line segment proposals and leverage complementary endpoint/junction information, we propose a method for binding line segment proposals and endpoint/junction proposals. This approach significantly reduces the number of joint proposals by incorporating geometry co-occurrence, serving as an effective and efficient replacement for non-maximum suppression (NMS) of line segment pairs. Furthermore, we introduce a novel and lightweight verification module called *endpoint-decoupled LOIAlign* (EPD LOIAlgin). This module captures geometry-aware discriminative features for verification in a lightweight design. Taking advantage of HAT fields, which produce high-quality proposals together with informative point-line co-occurrence patterns by the endpoint predictions from heatmaps, the hand-crafted designs used in L-CNN [30] and HAWPv1 are **no longer necessary to train the verification module**.

Obtaining ground-truth wireframes and other structural annotations for supervised learning is a time-consuming, costly, and often biased process. As a result, fully supervised wireframe parsers often struggle to produce satisfactory parsing results for images that differ significantly from the limited training data, leading to out-of-distribution failures. Motivated by this and aiming for more generalized wireframe parsers, we introduce HAWPv3, empowered by the expressive HAT field representation and inspired by the successes of SuperPoint [31] for keypoint detection and SOLD^2 [32] for self-supervised wireframe parsing. To overcome the limitations of fully supervised learning, we adopt a simplified homographic adaptation pipeline from the self-supervised learning approach of SOLD^2. Using this approach, HAWPv3 achieves remarkable efficiency and effectiveness **, requiring approximately 10 times less synthetic pretraining overall** and can be **trained within 24 GPU hours on a single-GPU workstation**. In particular, it significantly increases the crucial metric of structural repeatability scores for line segments, improving from 61.6% to 75.1% in the wireframe dataset [25], and from 62.9% to 71.1% in the YorkUrban dataset [33], respectively. These results underscore the expressive power of the HAT field representation and the elegantly designed wireframe parsing workflow. Fig. 1 (c)-(e) provide illustrative examples of the impressive HAWPv3 results in three diverse datasets.

In summary, this paper makes three key contributions to the field of wireframe parsing:

- The proposed line-segment-to-attraction-region lifting provides a new paradigm for learning expressive HAT fields for line segment representation, thanks to the richness of learnable information at the regional level. It shows significantly better effectiveness and expressiveness in both fully-supervised learning and self-supervised learning.
- The proposed wireframe parsing frameworks (HAWPv2 and HAWPv3) are elegantly designed and well-cooked. The proposed line segment binding module and the proposed end-point-decoupled LOIAlign facilitate built-in robustness and geometry awareness in wireframe parsing.
- In experiments, we demonstrate the superiority of our proposed HAWPv2 in supervised learning and our HAWPv3 in self-supervised learning with state-of-the-art performance obtained. Our HAWPv3 shows a great potential for general wireframe parsing in images out of the training distributions. In addition, our codes and trained models are publicly available[1].

Paper Organization. The remainder of this paper is organized as follows. Sec. 2 reviews the recent efforts in learning wireframe parsing and line segment detection and then summarizes our contributions. In Sec. 3, we present the HAT field of line segments and discuss it with alternative representations. Sec. 4 and Sec. 5 describe the framework of HAWP and the detail of learning HAWPv2 and HAWPv3 models, respectively. In the experiments, we evaluate the proposed HAWPv2 model in Sec. 6 and HAWPv3 model in Sec. 7. Last but not least, we conclude this paper in Sec. 8.

1. https://github.com/cherubicXN/hawp

2 Related Work

As mentioned earlier, wireframe parsing is a relatively new concept that focuses on modeling and computing line segments and junctions in 2D images, providing a concise representation for robust geometric understanding of the visual world. The roots of wireframe parsing can be traced back to the early days of computer vision, such as Larry Roberts' work on understanding the "Blocks World" [34], [35], as well as the primal sketch concept proposed by David Marr [1]. Over time, significant progress has been made in developing more expressive yet parsimonious representations and powerful yet efficient parsing algorithms to enhance the geometric understanding of images. Notable examples include the alignment-based LSD framework [36] with an a-contrario verification before the era of modern deep learning. In this section, we review recent advancements in learning-based wireframe parsing, which is the domain to which the proposed HAWP models belong.

Inductive and Deductive Wireframe Parsing. Wireframe parsing computation involves two phases: proposal generation and verification. For endpoint/junction proposals in wireframe parsing, methods commonly use heatmap regression. There are two main categories of approaches for line segment modeling in proposal generation: inductive parsing, which leverages line segment biases (e.g., learned line heatmaps) and deductive parsing, which enumerates all pairs of detected endpoints for line segment proposals.

The Deep Wireframe Parsing method and Wireframe dataset [25] introduced an inductive approach, connecting endpoints using learned line heatmaps for line segment proposals. Our previous work on regional attraction fields [37], [38] also falls into the inductive category, extracting line segments from attraction fields using a heuristic squeezing module. Deductive approaches [30], [39] bypass challenges by enabling end-to-end training. L-CNN [30] improves parsing with LOIPooling for proposal verification. Deductive approaches tackle class-imbalance issues caused by exhaustive enumeration, requiring careful sampling strategies during training. Subsequently, both our preliminary HAWPv1 [23] and the LGNN method [40] build on the success of LOIPooling in L-CNN [30] while addressing the inefficiency of line segment proposal generation. These approaches demonstrate that modeling the inductive biases of line segments in a more direct and appropriate manner can further enhance the accuracy and efficiency of wireframe parsing, all while retaining the benefits of end-to-end training. The LGNN method [40] introduces the center-offset representation for line segments, which, as discussed in Sec. 3.4, may exhibit slower convergence during the learning process.

Compared with prior art, the proposed HAWPv2 has the core HAT field representation first proposed in our preliminary HAWPv1, and harnesses the best of both line segment and endpoint proposals (the co-occurrence modeling and the end-point-decoupled LOIAlign as briefly discussed in Sec. 1) in a systematic way. The proposed HAWPv3 further extends the horizon of how wireframes can be learned by developing an effective SSL paradigm.

Line Segment Representations. As one of the most primitive geometric patterns/symbols, line segments are easy to describe and define mathematically, but have been shown to be extremely challenging to induce their conceptually simple inductive biases end-to-end in wireframe parsing. The key question is how to parameterize line segments in a differentiable way in the rasterized image lattice. There are three types of formulations that allow fully end-to-end training with different levels of effectiveness.

The center-offset and query-to-endpoints representations. Unlike the exhaustive enumeration in deductive approaches, alternative inductive representations based on center offset and Transformers [41], [42] have been explored for line segment detection and wireframe parsing. Inspired by the "objects as points" concept in object detection [43], line segments are parameterized by their center point, tangent angle, and Euclidean length in TP-LSD [29], F-Clip [27], M-LSD [44], and E-LSD [45]. Meanwhile, LETR [28] transforms latent queries into the endpoints of line segments using attention mechanisms. These representations, focusing on points along the line segments, are sparse and lack explicit context awareness. Consequently, these methods often require extensive training epochs (e.g., 300 epochs in F-Clip and 800 epochs in LETR).

The HAT field representation. It is first proposed in our preliminary HAWPv1 [23] before the two representations stated above. In contrast to them, the HAT field lifts line segments to non-overlapping attraction regions following our previous work on the regional attraction field representation [37], [38]. It then forms context-aware population coding for a line segment using "foreground" points in the attraction region with points on the line segment excluded. As a result, our HAWP models often use vanilla convolution operations and need only 30 epochs in training with state-of-the-art performance obtained in testing.

Self-Supervised Learning of Wireframe Parsing. Seeking proper pre-text tasks [46], [47] or self-consistency prediction/matching is one of the keys to self-supervised learning of modern deep vision models. In terms of the wireframe parsing and the related geometric tasks, pipelines that explore the simulation-to-reality workflow has also been studied. By utilizing the simulation-to-reality workflow, the SuperPoint [31] presented the first self-supervised learning method for interest point detection and description with the proposed Homography Adaptation approach. Inspired by the SuperPoint, SOLD^2 [32] presents the first self-supervised learning method for line detection using the deductive wireframe parsing framework similar to L-CNN [30].

SOLD^2 requires a heavy synthetic pretraining using 160k synthetic data. Due to its deductive nature, it also suffers from the severe class-imbalance issue in the proposal verification, but cannot use the carefully-designed sampling strategy proposed in L-CNN in SSL. It thus resorts to the edge maps as the primal cues for verifying proposals in training with real unlabeled images, and achieves better performance in terms of detection repeatability across viewpoints, thus facilitating the learned detectors in downstream tasks such as SLAM and SfM. Our proposed HAWPv3 adopts the overall SSL pipeline used by SOLD^2. The expressiveness and effectiveness of the core HAT field representation result in a much more efficient SSL (*e.g.*, 10x less synthetic pretraining cost) while being more accurate in terms of detection repeatability.

3 HAT Fields of Line Segments

In this section, we present details of the proposed end-to-end trainable HAT field representation that introduces a unified parsimonious closed-form 4D geometric vector space to encode line segments in a wireframe. We also discuss two alternative field representation schema for line segments and give a high-level explanation of why the proposed HAT field representation is better.

3.1 Notations

A wireframe is composed of a set of line segments. These line segments are represented using real coordinates (i.e., sub-pixels) within the continuous image domain $D \subset \mathbb{R}^2$. We also utilize the representation of line segments over the discrete image lattice $\Lambda \subset D$, which consists of integer pixel coordinates (for example, 512×512 pixels) in rasterization. We denote a line segment as $\ddot{l} = (\mathbf{x}1, \mathbf{x}2) \in D \times D$. Here, the symbol $\ddot{}$ is used to highlight the two endpoints of a line segment. The line that passes through the line segment $\ddot{l}$ in $\mathbb{R}^2$ is denoted by $\boldsymbol{l} = (\mathbf{a}_{\ddot{l}}, b_{\ddot{l}})$. In this case, $\mathbf{a}_{\ddot{l}} \in \mathbb{R}^2$ and $b_{\ddot{l}} \in \mathbb{R}$ represent the coefficients (slope and intercept) of the line equation, which are uniquely determined by $(\mathbf{x}_1, \mathbf{x}_2)$.

3.2 The Attraction Region of a Line Segment

Utilizing the method from our previous work [37], [38], we introduce an attraction region for each line segment to counter its inherent locality. As depicted by the gray region in Fig. 2 (a), each point $\mathbf{p} \in \Lambda$ is assigned to the line segment $\ddot{l}$ to which it is closest. Distance is computed by projecting $\mathbf{p}$ onto the line $\boldsymbol{l}$ of $\ddot{l}$ to get $\mathbf{p}' \in D$. If $\mathbf{p}'$ is not on $\ddot{l}$, it's reassigned to the nearest endpoint. The distance between $\mathbf{p}$ and $\ddot{l}$ is the Euclidean distance between $\mathbf{p}$ and $\mathbf{p}'$. Attraction regions of different line segments are disjoint, enabling regional representation of line segments and facilitating more effective learning.

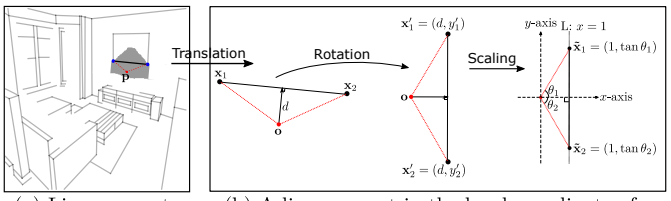
Once the attraction region of a line segment is computed, any pixel within this region can "recover" the line segment, given a suitable encoding scheme. This approach effectively counters the inherent locality and corresponding ambiguity of line segments. However, the question arises: **What is the suitable encoding scheme that enables this representational effectiveness?** In the following sections, we will first introduce the proposed parsimonious closed-form 4D geometric vector field. Subsequently, we will discuss alternative representations that are less effective in experimental comparisons, as shown in Fig. 4 and Sec. 6.

3.3 The HAT Field of a Wireframe

The HAT field of a wireframe can be defined either in the same image domain D or in a downsampled one to comply with the design of ConvNets, denoted by $\mathcal{A}$. For each point $\mathbf{p} \in \Lambda$, $\mathcal{A}(\mathbf{p})$ is a 4D geometric vector derived as follows.

As shown in the right columns in Fig. 2, for a line segment $\ddot{l}$, our derivation undergoes a simple affine transformation for each distant point $\mathbf{p}$ in its attraction region. Let d be the distance between $\mathbf{p}$ and $\ddot{l}$, *i.e.*, $d = |\mathbf{a}_{\ddot{l}}^\top \cdot \mathbf{p}' + b_{\ddot{l}}| > 0$. We have,

i) *Translation*: The point $\mathbf{p}$ is then used as the new coordinate origin.



(a) Line segments (b) A line segment in the local coordinate of $\mathbf{p}$

Fig. 2. Illustration of constructing the proposed HAT field representation for wireframes. (a) displays an example of wireframes, highlighting one line segment in black with its two endpoints in blue, and the corresponding attraction region shaded in solid gray. (b) demonstrates the derivation of the 4D geometric vector representation for a "foreground" point/pixel $\mathbf{p}$ within the attraction region. See text for details.

ii) *Rotation*: The line segment is then aligned with the vertical y-axis with the end-point $\mathbf{x}_1$ on the top and the point $\mathbf{p}$ (the new origin) to the left. The rotation angle is denoted by $\theta \in [-\pi, \pi)$.

iii) *Scaling*: The distance d is used as the unit length to normalize the x- / y-axis in the new coordinate system.

In the new coordinate system after the affine transformation, let $\theta_1 \in (0, \frac{\pi}{2})$ and $\theta_2 \in (-\frac{\pi}{2}, 0]$ be the two angles as illustrated in Fig. 2. So, a point $\mathbf{p}$ in the attraction region of a line segment $\ddot{l}$ is reparameterized as,

$$\mathbf{p}(\ddot{l}) = (d, \theta, \theta_1, \theta_2), \tag{1}$$

which can exactly recover the line segment in a closed form in the ideal case using,

$$\ddot{l} = d \cdot \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 1 \\ \tan\theta_1 & \tan\theta_2 \end{pmatrix} + (\mathbf{p}^\top, \mathbf{p}^\top), \tag{2}$$

where an estimated 4D vector $(\hat{d}, \hat{\theta}, \hat{\theta_1}, \hat{\theta_2})$ will generate a line segment proposal.

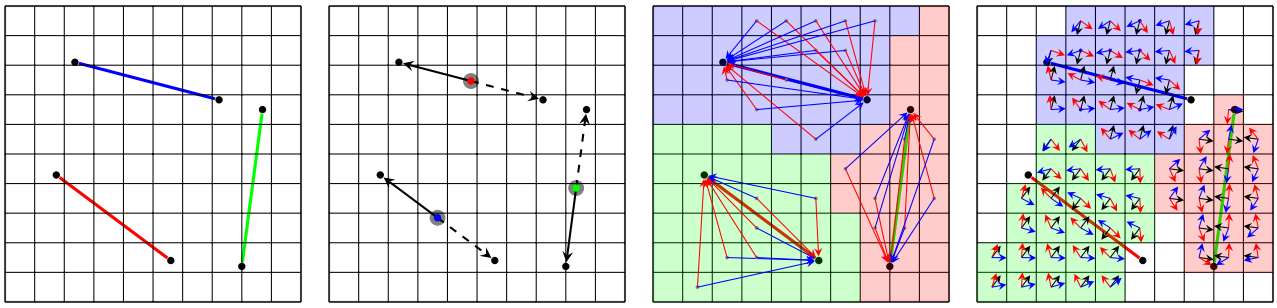
Differentiability of the Proposed Line Segment Representation. Based on Eqn.(2), a significant merit of the proposed holistic attraction field representation is its differentiability at every point in the attraction region. This 4D geometric parameterization of line segments leads to a new loss function, as proposed in Eqn.(7), for training.

Foreground and Background Mask of a Holistic Attraction Field. Certain points (pixels), such as those on any line segments, should not be reparameterized to prevent degradation and are thus classified as the "background". Additionally, we create "shrunk" attraction regions for line segments by excluding points whose distances exceed a predefined threshold τ_d (defined later). This approach allows the model to concentrate more on the immediate surrounding context. We denote the mask by $\mathbf{M}(\mathbf{p}) = 1$ for the foreground and $\mathbf{M}(\mathbf{p}) = 0$ for the background. Background points not attracted by any line segment, as per our specification, are encoded using a 4-D null vector.

So, computing line segment proposals for a wireframe is posed as a map-to-map (*i.e.*, image-to-attraction-field) regression problem, which enables us to exploit many encoder-decoder neural networks in learning.

3.4 Alternative Attraction Field Representations

For comparisons, we discuss two alternative encoding schema of vector field representations for line segments as shown in Fig. 3 using a toy example.



(a) A toy example (b) Center-offset Vectors (c) 4D Attraction Field (d) Holistic Attraction Field

Fig. 3. Illustrative comparisons of different representations of a toy example (a). In (b), the center offset representation parameterizes three line segments by three center points and the corresponding offset vectors. In (c), the 4D attraction field uses all pixels to explicitly characterize line segments as two offset vectors. In (d), three angles θ (in black arrows), θ_1 (in red arrows), and θ_2 (in blue arrows) characterize the two endpoints of the corresponding line segment in a "normalized" coordinate frame and transform the "normalized" coordinate representation to the image coordinate by the distance component. The distance component is hidden to clearly show the angle components.

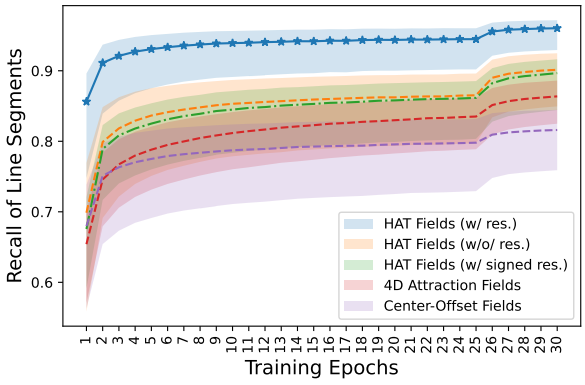


Fig. 4. Comparison of training convergence in terms of line segment recall rates among different field representations. Representations compared include our proposed HAT fields (with and without unsigned residual learning) and an alternative to signed residual learning, the 4D attraction fields, and the center-offset fields. Within each colored region, we compute the recall rates at various training epochs using different thresholds $\vartheta \in [5, 15]$ as defined in Eq. (23), with the results for $\vartheta = 10$ marked in line patterns.

The Regional Attraction Field Representation. It is a straightforward extension of our previous regional attraction work [37], [38]. Similarly, it also utilizes the attraction region lifting for line segments. As illustrated in Fig. 2(a) and Fig. 3(c), consider a distant pixel point $\mathbf{p}$ outside a line segment $\ddot{l}$, with the projection point $\mathbf{p}'$ being on the line segment. The conventional regional attraction method reparameterizes $\mathbf{p}$ as $\mathbf{p} - \mathbf{p}'$, i.e., the displacement vector in the image domain. This results in a 2-D vector field for a wireframe by reparameterizing all the pixel points in the image lattice Λ. If we use the two displacement vectors between $\mathbf{p}$ and the two endpoints of the line segment, we can reparameterize $\mathbf{p}$ by its 4D displacement vector, which can completely determine the line segment (i.e., an exact dual representation). By including the displacement vector $\mathbf{p}-\mathbf{p}'$, we can obtain a 4D/6D field representation, depending on whether the perpendicular vectors are encoded.

In the process of developing the proposed HAT field representation, we initially used the 4D/6D regional attraction field representation. However, we found that these representations could not be accurately and reliably learned during training with deep neural networks (DNNs). Despite the 4D/6D regional attraction field's capability to capture all the necessary information for recovering line segments in a closed-form way, it contains large structural variations that pose challenges for learning. Our observations suggest that DNNs are not sufficiently effective in learning and predicting this type of information.

In contrast, the proposed HAT field representation incorporates both distance and angles, defined in a line-segment-aware local coordinate system (via the aforementioned affine transformation), making it holistic. The combination of distance and angle information helps address the learnability issue associated with solely fitting distance or displacement. It's important to note that even with the holistic attraction field representation, we still need to employ a residual learning method to predict the distance component. In Fig. 4, we compare the different strategies of learning unsigned and the signed distance residuals, showing that the signed distance residual learning will lead to a lower recall rate even than the model without residual learning.

The Center-Offset Field Representation. In recent literature, the center-offset field representation has been extensively used to represent annotated line segment maps ([27], [29], to cite a few). Given a line segment $\ddot{l} = (\mathbf{x}_1, \mathbf{x}_2)$, it is encoded by the center point $\mathbf{x}_c = (\mathbf{x}_1 + \mathbf{x}_2)/2$, the tangent angle α (or its cos counterpart) and the length $l = \|\mathbf{x}_1 - \mathbf{x}_2\|_2$. Although the center-offset field representation enjoys a simple formulation, it is a sparse field and lacks the richness contributed by lifting line segments to attraction regions in our HAT field representation. Consequently, learning the center-offset field representation often suffers from slow convergence. For example, the best-performing detector, F-Clip [27], entails a very long learning schedule (of 300 epochs) to achieve promising results to overcome the difficulty of predicting the length component accurately and reliably.

We present the comparisons between our HAT field representation and the two alternatives in Fig. 4, which shows that our HAT field representation enables a much faster learning convergence in terms of the recall of line segments (see Sec. 5.1 and Sec. 6 for experimental settings). But, why? We provide a high-level explanation as follows.

Why is the HAT field an "Attractive" Representation? In addition to the computational merit of being differentiable (Eqn. 2), the proposed holistic attraction field also has a strong intuition that has not been fully exploited by the

alternatives. As a toy example in Fig. 3(a) that has 3 line segments defined on the image grid, the center-offset representation (and its variant) has to face two challenging cases to be solved: (1) how to accurately detect the center locations and (2) how to cope with the large variation for the small-length and large-length line segments that are presented in the same image to accurately regress the length of line segments. As a result, the methods built on the center-offset representations usually have long training schedules with hundreds of epochs. To further improve the learning ability, some hand-crafted neural modules based on [48], [49] were studied [27], [29], [45] for feature aggregation.

Different from center-offset representations, which are strictly defined on center pixels, the regional representations in Fig. 3(c) and Fig. 3(d) have a more relaxed definition by incorporating non-edge pixels. With the involvement of more pixels, there is no need to precisely localize the foreground pixels during learning; instead, the focus can solely be on the regression of the fields. However, the 4D attraction field in Fig. 3(c), which utilizes long-range vectors to depict line segments, is susceptible to large structural variations during the learning process. By contrast, our HAT field representation normalizes the two long-range vectors using three angles (or unit vectors) in the local coordinate system, and incorporates the distance to transform the local line segments into the image coordinate system. This type of "representation normalization" eliminate many nuisance factors in the data, thereby facilitating more effective learning. Moreover, the joint encoding that exploits displacement distance and angle effectively decouples the attraction field with respect to complementary spanning dimensions.

4 The Proposed HAWP Framework

In this section, we present details of the proposed HAWP framework that is built on the HAT field representation of a wireframe. As illustrated in Fig. 5, the proposed HAWP consists of three components: line segment and proposal generation (Sec. 4.2 and Sec. 4.3), line segment and binding (Sec. 4.4), and endpoint-decoupled line-of-interest verification (Sec 4.5).

4.1 Notations

Let I represent an image defined on the image lattice Λ with dimensions $H \times W$ pixels (e.g., 512×512). For the wireframe associated with image I, we define $\ddot{\mathbf{L}}$ as the set of annotated line segments in I, and $\dot{\mathbf{J}}$ as the set of unique endpoints belonging to all line segments in $\ddot{\mathbf{L}}$. It is worth noting that many of these endpoints are junction points formed by multiple line segments.

We denote $f_b(\cdot;\Omega_b)$ as the deep neural network feature backbone with parameters Ω_b. Given an input image I, the feature backbone produces an output feature map $F = f_b(I)$ with dimension C, height $H_s = \frac{H}{s}$, and width $W_s = \frac{W}{s}$. The output resolution depends on the overall stride s of the backbone, and the resulting lattice is Λ', a sub-sampled version of the original lattice Λ. On top of the feature map F, we employ lightweight head sub-networks to regress both the 4D HAT field for line segments and the endpoint heatmap. *Detailed network architectures can be found in Appx. A*. In this section, we refer to them as general mapping functions in defining our HAWP.

The predicted HAT field is computed at the resolution of $H_s \times W_s$, we prepare the ground-truth field maps at the same resolution for computing the loss of the predicted holistic attraction field in training accordingly. Since a wireframe is a vectorized representation, the mapping from the original image lattice Λ to the (sub-sampled) lattice Λ' is straightforward. Without loss of generality, we directly use the lattice Λ' when referring to line segments and junction points in the formulations hereafter.

4.2 Learning the HAT Field of Line Segments

We use two separate head sub-networks in learning the distance and the three angles in our proposed 4D holistic attraction field $\mathcal{A}$.

$$\text{The distance map:} \quad \mathcal{A}_d = f_d(F;\Omega_d) \in \mathbb{R}^{1\times\Lambda'}, \tag{3}$$

$$\text{The angle field:} \quad \mathcal{A}_a = f_a(F;\Omega_a) \in \mathbb{R}^{3\times\Lambda'}. \tag{4}$$

Considering an annotated line segment $\ddot{l} \in \ddot{\mathbf{L}}$, a foreground point $\mathbf{p}' \in \Lambda'$ within the attraction region of $\ddot{l}$ is reparameterized as $\mathbf{p}'(\ddot{l}) = (d, \theta, \theta_1, \theta_2)$ using Eq. (1). Here, $\mathbf{p}' = \lfloor \mathbf{p}/s \rfloor \in \Lambda'$ is the mapped point from $\mathbf{p} \in \Lambda$. The ground-truth distance map and angle field are denoted as $\mathcal{A}_d^{gt}$ and $\mathcal{A}_a^{gt}$, respectively. To facilitate neural network training, we normalize the distance map by $\min(\max(d/\tau_d, 0), 1)$, where τ_d controls the foreground pixels. The angles are normalized as $(\frac{\theta}{2\pi} + \frac{1}{2}, \frac{\theta_1}{\pi/2}, \frac{\theta_2}{\pi/2} + 1)$. This normalization ensures that the predicted values of the HAT field fall within the range of $[0, 1]$, and they can be unnormalized to obtain line segment proposals.

Residual Learning of the Distance Map. As aforementioned, we observe that the distance map is more difficult to learn than the angle fields in our experiments. To address this, we propose a residual learning method by introducing another light-weight sub-network to predict the distance residual,

$$\mathcal{A}_{\Delta d} = f_{\Delta d}(F;\Omega_{\Delta d}) \in \mathbb{R}^{1\times\Lambda'}. \tag{5}$$

The ground-truth of the distance residual map is computed on the fly, $\mathcal{A}_{\Delta d}^{gt}(\mathbf{p}') = |\mathcal{A}_d^{gt}(\mathbf{p}') - \mathcal{A}_d(\mathbf{p}')|$. Fig. 6 shows some examples of the learned distance maps and residual maps.

Note that we learn the unsigned residual instead of the signed one, which is mainly due to that the distance field and its residual take the same feature map as input to avoid unnecessary increase of model complexity. Furthermore, the learning of unsigned residual can be viewed as a kind of uncertainty estimation for the distance prediction, thus facilitating the learning of HAT fields as shown in our experiments. Given by this, we enumerate their signs and modulate the scales to get a set $K = \{-k, \ldots, 0, \ldots, k\}$ (k is a hyperparameter, to be specified in experiments), and eventually yield a set of rectified distance maps by,

$$\mathcal{A}_d^{(i)}(\mathbf{p}') = \mathcal{A}_d(\mathbf{p}') + i \cdot \mathcal{A}_{\Delta d}(\mathbf{p}'), \ \ \forall i \in K. \tag{6}$$

With the rectified distance maps, $2k+1$ line segment proposals will be generated at each foreground point and refined by the line-of-interest verification step.

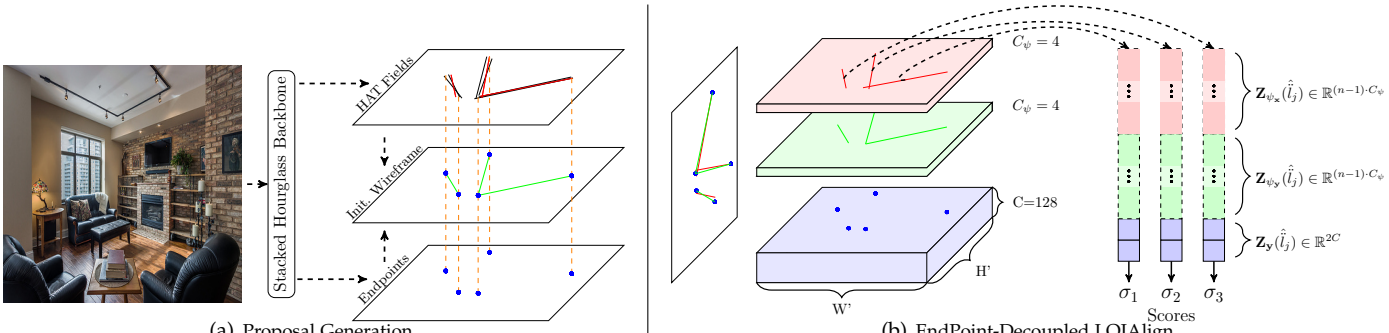


Fig. 5. The overall architecture of the proposed HAWP framework. In (a), the HAT Fields and Junctions are predicted from the shared backbone network to yield initial wireframe proposals by binding the line segments (from the predicted HAT field) and junctions (from the predicted heatmap) together into the green line segments; In (b), we present an EPD LOIAlign module to extract the endpoint features from the C-dim feature map, and the HAT-augmented line segment features from two thin feature maps with the C_ψ-dim for the HAT line segment (in red) and the junction-refined line segment (in green). With the HAT-sourced line segment as an augmentation, the co-occurrence patterns between the HAT fields and the junction predictions are captured for better verification, facilitating the learning of HAWPv2 and HAWPv3 models.

Fig. 6. Some illustrative examples for the learning of distance and residual maps. For each input image, the predicted distance (Dis. Pred.) and the absolute distance residuals (Res. Pred.) are displayed. The darker color is corresponding to the smaller values.

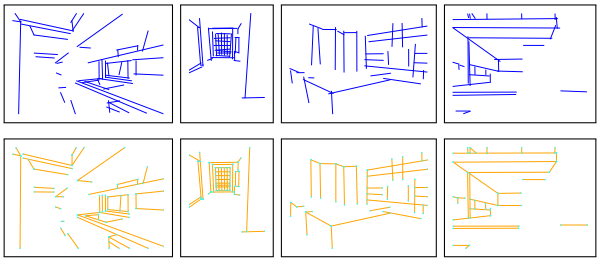


Fig. 7. Some illustrative examples for the learned line segments by HAT fields (top) and the corresponding wireframes (bottom) that are refined by the endpoint binding. Best viewed in magnification.

Loss Functions. We use the ℓ_1 loss function between the ground-truth maps $\mathcal{A}_d^{gt}$, $\mathcal{A}_a^{gt}$, $\mathcal{A}_{\Delta d}^{gt}$ and the corresponding predicted maps $\mathcal{A}_d$, $\mathcal{A}_a$, $\mathcal{A}_{\Delta d}$. The loss is computed across the foreground points only based on the mask $\mathbf{M}(\mathbf{p}')$.

Due to the multiple line segment proposals generated by the residual learning of the distance map at each foreground point, we propose a loss function to directly minimize the endpoint error, $\mathcal{L}_{epe}$, between the line segment proposal and the ground one, thanks to the differentiability of our holistic attraction field (Eqn. 2). As the unsigned distance residual will generate $2k+1$ line proposals for each foreground pixel $\mathbf{p}'$, we encourage all the $2k+1$ proposals to be close to the line segment $\ddot{l}^{gt}(\mathbf{p}')$, and get the loss $\mathcal{L}_{epe}$ by

$$\mathcal{L}_{epe} = \sum_{i=-k}^{k} \sum_{\mathbf{p}' \in \Lambda'} \frac{\mathbf{M}(\mathbf{p}')}{\left\| \ddot{l}^{gt}(\mathbf{p}') \right\|} \ell_1(\ddot{l}^{(i)}(\mathbf{p}'), \ddot{l}^{gt}(\mathbf{p}')), \quad (7)$$

where $\ddot{l}^{(i)}(\mathbf{p}')$ is the i-th line segment proposal in the residual learning at the point $\mathbf{p}'$ and the $\ddot{l}^{gt}(\mathbf{p}')$ the ground-truth line segment. $\|\cdot\|$ represents the length of a line segment.

The total loss $\mathcal{L}_{\ddot{\mathbf{L}}}$ of learning line segments via the HAT field is simply the sum of the different ℓ_1 loss terms (without trade-off tuning parameters).

Fig. 7 shows some examples of generated line segment proposals via learning the HAT field. We can see that the richness of our HAT field also leads to dense proposals of line segments that need to be refined for the final wireframe parsing. One straightforward way is to directly learn to score the line segment proposals followed by the Intersection-over-Union (IoU) based NMS, similar in spirit to object detection systems in [43], [50], [51], however, it is not only time-consuming but also ill-defined (unlike that of bounding boxes). We address this issue by learning endpoints and then binding them with line segment proposals for pruning false positive line segment proposals.

4.3 Learning the Heatmap-Offset Field of Endpoints

The heatmap-offset field representation is proposed in the L-CNN [30]. The heatmap regression is a widely used method in different types of keypoint detection (*e.g.*, in human pose estimation). The inclusion of an offset field aims to enhance the accuracy of sub-pixel localization for endpoints. In this approach, the heatmaps for endpoint regression and the offset field are computed by two separate head sub-networks on top of the feature backbone output F:

$$\text{The endpoint heatmap:} \quad \dot{\mathbf{J}}_{pt} = f_{pt}(F; \Omega_{pt}) \in \mathbb{R}^{1 \times \Lambda'}, \quad (8)$$

$$\text{The offset field:} \quad \dot{\mathbf{J}}_{o} = f_{o}(F; \Omega_{o}) \in \mathbb{R}^{2 \times \Lambda'}. \quad (9)$$

The corresponding ground-truth maps are straightforwardly defined. Based on the set of annotated endpoints

$\dot{\mathbf{J}}$ in an image, we use the "hard" (binary) heatmap as the ground truth,

$$\dot{\mathbf{J}}^{gt}_{pt}(\mathbf{p}') = 1, \forall \mathbf{p} \in \dot{\mathbf{J}}, \tag{10}$$

where $\mathbf{p}' = \lfloor \mathbf{p}/s \rfloor \in \Lambda'$ and $\mathbf{p} \in \Lambda$. The associated offset is then defined by,

$$\dot{\mathbf{J}}^{gt}_{o}(\mathbf{p}') = \mathbf{p}/s - \mathbf{p}' \tag{11}$$

Loss Functions. We use the binary cross-entropy loss $\text{BCE}(\cdot,\cdot)$ for the endpoint heatmap regression, and the ℓ_1 loss for the offset field,

$$\mathcal{L}_{pt} = \text{BCE}(\dot{\mathbf{J}}_{pt}, \dot{\mathbf{J}}^{gt}_{pt}), \tag{12}$$

$$\mathcal{L}_{o} = \sum_{\mathbf{p}'} \ell_1(\dot{\mathbf{J}}_{o}(\mathbf{p}'), \dot{\mathbf{J}}^{gt}_{o}(\mathbf{p}')) \cdot \dot{\mathbf{J}}^{gt}_{pt}(\mathbf{p}'), \tag{13}$$

The total loss of learning endpoints is $\mathcal{L}_{\dot{\mathbf{J}}} = \beta_{pt} \cdot \mathcal{L}_{pt} + \beta_0 \cdot \mathcal{L}_o$, where β_{pt} and β_o are trade-off hyperparameters. In our experiments, β_{pt} and β_o are set to 8.0 and 0.25, exactly same to L-CNN [30] and our preliminary version, HAWPv1 [23].

Extracting Endpoints from the Dense Predictions. With the predicted heatmap scores, we first apply the local NMS using a 3×3 window, and then keep the top-N endpoints using a sufficiently large number N. In our experiments, we use $N = \max(2 \times N_{gt}, 300)$ in the training phase where N_{gt} the number of junctions in an image. In testing, we use $N = \max(N_{pred}, 300)$ with $N_{pred} = \sum_{\mathbf{p}'} \mathbf{1}_{\dot{\mathbf{J}}_{pt}(\mathbf{p}') \geq \tau_j}$, where τ_j is set to 0.008, same as L-CNN [30] and our HAWPv1 [23]. For the top-N endpoints, their (sub-pixel) locations are updated based on the predicted offsets in $\dot{\mathbf{J}}_o$.

4.4 Binding Line Segment and Endpoint Proposals

Denote by $\hat{\dot{\mathbf{L}}}$ the set of line segment proposals from the HAT field prediction and by $\hat{\dot{\mathbf{J}}}$ the set of endpoint proposals. Binding them captures their co-occurrence and improves the fidelity of line segment proposals while significantly reducing the computational cost to handle the total of $|\hat{\dot{\mathbf{L}}}| = (2k+1) \cdot H_s \cdot W_s$ proposals in $\hat{\dot{\mathbf{L}}}$, particularly those strongly supported by endpoint proposals.

Specifically, we first find the nearest endpoint proposals in $\hat{\dot{\mathbf{J}}}$ for the two endpoints of a line segment proposal in $\hat{\dot{\mathbf{L}}}$. Without loss of generality, consider a line segment proposal $\hat{\dot{l}} = (\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2)$, denote by $\hat{\mathbf{y}}_1 \in \hat{\dot{\mathbf{J}}}$ the nearest endpoint proposal for $\hat{\mathbf{x}}_1$ and the squared Euclidean distance between them is δ_1. The same is done for $\hat{\mathbf{x}}_2$ and we obtain $\hat{\mathbf{y}}_2$ and δ_2. The figure of merit of the binding is defined by the maximum distance $\delta = \max(\delta_1, \delta_2)$, and the smaller the distance, the higher the quality of the line segment $\hat{\dot{l}}$ (*i.e.*, the likelihood of the co-occurrence is high). A threshold τ_δ is used to select proposals of high-quality line segments whose binding costs δ are smaller than the threshold ($\tau_\delta = 10$ in our experiments). Based on this, the $|\hat{\dot{\mathbf{L}}}|$ number of line segments $\hat{\dot{\mathbf{L}}}$ in are anchored by the set $\hat{\dot{\mathbf{J}}}$ in a sparse set without incurring any non-maximal suppression schema of line segments, and eventually yield a new set of **endpoint-augmented line segment** proposals consisting of line segments $\hat{\dot{l}}_j = (\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2; \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2)$, where $\hat{\dot{l}}_j$ is used to indicate that the line segment is formed by the binding process, and the endpoints from both $\hat{\dot{\mathbf{L}}}$ and $\hat{\dot{\mathbf{J}}}$ are retained for preserving the instrinsic uncertainty of proposals. Eventually, the endpoint-augmented line segment proposals generate a new set $\hat{\dot{\mathbf{L}}}_{\dot{\mathbf{J}}}$.

4.5 Line Segment Verification

Recall that in the proposed HAT field, points on line segments are treated as "background" points. Even after the binding, those points have not been verified. Thus, verifying line segment proposals in $\hat{\dot{\mathbf{L}}}_{\dot{\mathbf{J}}}$ entails grounding the entirety of a line segment proposal to the data evidence, *i.e.*, Line-of-Interest (LOI) Pooling, to compute the final score for wireframe parsing, where the entirety is defined with respect to a point-sampling strategy. We give details of our proposed endpoint-decoupled (EPD) LOIAlign method, which is built on the vanilla LOIPooling method [30].

To enable an efficient design of the verification head MLP classifier, we ask the question: *Do we need to encode every sampled point in the same high-dim space as done in the vanilla LOIPooling?* By definition, the two endpoints of a line segment play a critical role. So, we propose to encode the two endpoints and the sampled intermediate points of a line segment differently in a high-dim feature space and a low-dim feature space, respectively, that is, to develop the EPD LOIAlign.

Denoted by a linear-sampling function $\psi_t(\ddot{l})$ that maps a line segment $\ddot{l} = (\mathbf{x}_1, \mathbf{x}_2)$ to a point in the line segment by

$$\psi_t(\mathbf{x}_1, \mathbf{x}_2) = (1-t) \cdot \mathbf{x}_1 + t \cdot \mathbf{x}_2,\ t \in [0,1], \tag{14}$$

where a predefined number of evenly-sampled points $t_i = \frac{i}{n}$ is typically used (*e.g.*, $n = 31$ and $i \in \{0, 1, \cdots n\}$).

Consider an endpoint-augmented line segment $\hat{\ddot{l}}_j = (\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2; \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2)$, for the LOI verification, we maintain three subsets of sampled points:

- The two endpoints: $\{(\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2)\}$;
- The intermediate points between $\hat{\mathbf{y}}_1$ and $\hat{\mathbf{y}}_2$: $\mathcal{Y} = \{\psi^y_{t_i} = \psi_{t_i}(\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2)); i = 1, 2, \cdots, n-1\}$;
- The intermediate points between $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$: $\mathcal{X} = \{\psi^x_{t_i} = \psi_{t_i}(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2)), i = 1, 2, \cdots, n-1\}$.

By decoupling endpoints and intermediate points, the model will be geometrically aware for line segments in learning to verify the proposals. We note that our proposed EPD LOIAlign captures the co-occurrence between the HAT-drive inductive line segment proposals and the junction-guided deductive line segment proposals for better verification.

The Verification Head Classifier. To exploit the EPD LOIAlign and to enrich the information flow for the low-dimensional feature extractor f_ψ, we utilize a parallel branch design of the verification head classifier. As shown in Fig. 5(b), we use three convolution layers to transform the backbone network to $F_J \in \mathbb{R}^{C \times H_s \times W_s}$ for the two endpoints, as well as $F_{\mathcal{Y}} \in \mathbb{R}^{C_\psi \times H_s \times W_s}$ and $F_{\mathcal{X}} \in \mathbb{R}^{C_\psi \times H_s \times W_s}$ for the sampled point set $\mathcal{Y}$ and $\mathcal{X}$. Note that the feature maps $F_{\mathcal{Y}}$ and $F_{\mathcal{X}}$ have fewer channels than F_J to deal with the sample points $(n-1)$ for each proposal. The bilinear interpolation is used to sample all features, and yield three feature vectors for each proposal, $\mathbf{Z}_{\mathbf{y}}(\hat{\ddot{l}}_j)$ for the

two endpoints, as well as $\mathbf{Z}_{\psi_\mathbf{y}}$ and $\mathbf{Z}_{\psi_\mathbf{x}}$ for the sampled point set $\mathcal{Y}$ and $\mathcal{X}$ for the line verification, denoted by

$$\mathbf{Z}_\mathbf{y}(\hat{\hat{l}}_j) = [F_J(\mathbf{y}_1), F_J(\mathbf{y}_2)] \in \mathbb{R}^{2C}, \quad (15)$$

$$\mathbf{Z}_{\psi_\mathbf{y}}(\hat{\hat{l}}_j) = [F_\mathcal{Y}(\psi^y_{t_1}), \dots, F_\mathcal{Y}(\psi^y_{t_{n-1}})] \in \mathbb{R}^{(n-1)\cdot C_\psi}, \quad (16)$$

$$\mathbf{Z}_{\psi_\mathbf{x}}(\hat{\hat{l}}_j) = [F_\mathcal{X}(\psi^x_{t_1}), \dots, F_\mathcal{X}(\psi^x_{t_{n-1}})] \in \mathbb{R}^{(n-1)\cdot C_\psi}. \quad (17)$$

Let $\mathbf{Z}_\psi(\hat{\hat{l}}_j) = [\mathbf{Z}_{\psi_\mathbf{y}}(\hat{\hat{l}}_j), \mathbf{Z}_{\psi_\mathbf{x}}(\hat{\hat{l}}_j)] \in \mathbb{R}^{2\cdot(n-1)\cdot C_\psi}$ be the concatenated features of the intermediated sample points and $\mathbf{Z}(\hat{\hat{l}}_j) = [\mathbf{Z}_\mathbf{y}(\hat{\hat{l}}_j), \mathbf{Z}_\psi(\hat{\hat{l}}_j)] \in \mathbb{R}^{2\cdot(n-1)\cdot C_\psi + 2\cdot C}$ be the concatenated features of all sample points, we first apply two separate MLPs to transform $\mathbf{Z}_\psi$ and $\mathbf{Z}$ into the D-dim feature space (*e.g.*, $D = 128$) respectively, and then sum them, followed by appling a linear transformation to compute the final score (or logit) of the line segment. For simplicity of notation, we will omit "$(\hat{\hat{l}}_j)$" and instead use $\mathbf{Z}_\psi$ and $\mathbf{Z}$.

$$\text{Score}(\hat{\hat{l}}_j) = \text{Linear}(\text{MLP}(\mathbf{Z}_\psi) + \text{MLP}(\mathbf{Z})). \quad (18)$$

To further enhance the learning of the low-dimensional feature extractor f_ψ, we use an auxiliary verification layer (linear transformation) that is used in training only,

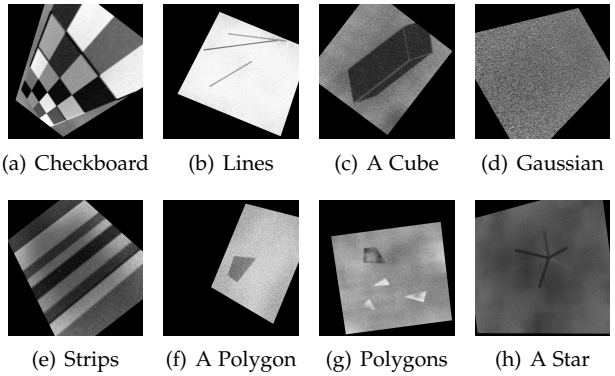
$$\text{AuxScore}(\hat{\hat{l}}_j) = \text{Linear}(\mathbf{Z}_\psi). \quad (19)$$

The Ground-Truth Assignment in Training. A line segment proposal $\hat{\hat{l}}_j = (\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2; \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2)$ is assigned as positive if and only if there exists a ground-truth line segment $\ddot{l}$ within the close proximity of $\hat{\hat{l}}_j$. Similar to the binding between line segment proposals and endpoint proposals, the proximity is defined by the maximum distance between the two endpoints of the ground-truth line segment and $(\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2)$. By close, it means that the maximum distance needs to be smaller than a predefined threshold, τ_{ver} (*e.g.*, $\tau_{ver} = 1.5$ used in our experiments). This ground-truth assignment method facilitates learning the negatives on the fly and end-to-end, unlike previous work [23], [30] that exploit static negative line segment samples (*e.g.*, by sampling two s that do not come from any ground-truth line segments). As we shall show, this on-the-fly and end-to-end generation of negative samples enables much more effective self-supervised learning of our HAWP model.

Loss Functions. The verification is posed as the binary classification problem. We utilize the BCE loss in training both the verification head classifier and the auxiliary head classifier. Denote by $\mathcal{L}_{ver}$ and $\mathcal{L}_{aux}$ the BCE loss functions for training Eq. (18) and Eq. (19) respectively.

5 Learning HAWP Models

In this section, we present details of training and inference of both HAWPv2 and HAWPv3. Due to the space limit, we present network architectures used in our experiments in the Appx. A.



(a) Checkboard (b) Lines (c) A Cube (d) Gaussian

(e) Strips (f) A Polygon (g) Polygons (h) A Star

Fig. 8. Some training examples generated in the synthetic dataset for the initialization of self-supervised learning. There are 8 primitives in total used in the dataset and we show one random synthesized image for each primitive.

5.1 HAWPv2 with Fully-Supervised Learning

Training Details Our proposed HAWPv2 is trained on the Wireframe dataset [25] with their official annotations. In the training, a total of 5000 training samples with precisely annotated wireframes are augmented by the flipping operations along the horizontal, vertical, and diagonal directions, as well as the rotation operations[2] by 90° and -90°. Finally, the augmented dataset with 30k samples is used to train HAWPv2 with a total of 30 epochs. The ADAM optimizer [52] is used for training, and the learning rate is initially set at 4e-4 for the first 25 epochs and then reduced to 4e-5 for the last 5 epochs. The image resolution of the training samples is set to 512×512. The total loss is the sum of the loss functions defined in Sec. 4.

Inference Details Given an input image, we forward the trained HAWPv2 model to get the junctions and line segments. We only keep the line segments of which the classification score c_i is greater than a given threshold ε as the final prediction. The input image is also resized to 512×512 tensors for the forward to obtain the wireframes that are then scaled to their original resolution according to the scaling factors in both horizontal and vertical directions.

5.2 HAWPv3 with Self-Supervised Learning

We adopt the simulation-to-reality pipeline for SSL as in the SuperPoint [31] and the SOLD^2 [32]. In the synthetic pretraining using simulated data consisting of simple primitives (Fig. 8), ground-truth wireframes are naturally available. We can thus train our HAWPv2 model. The challenge is how to leverage the synthetically pretrained HAWPv2 to "annotate" wireframes in real images. If we directly run the synthetically trained HAWPv2 on an input unlabeled image. The putative wireframe is often quite noisy due to the gap between synthetic data and real images. To address this challenge, we need to introduce additional inductive biases of line segments that are more transferrable from simulation to reality, such that we can exploit them to "clean up" the putative wireframe to get the pseudo wireframe labels for

2. After 2020, the more complicated data augmentation techniques including the rotational augmentation used in F-Clip [27] and ELSD [45], the random cropping and color jittering used in LETR [28] were explored to boost the final performance.

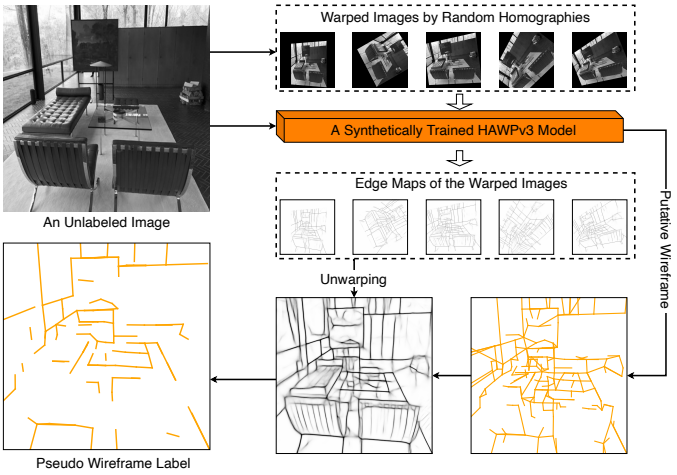


Fig. 9. The homographic adaptation scheme for the HAWPv3 training. Different from SOLD^2 [32], in our HAWPv3, the warped images are only used to predict the edge maps that are then unwarped back to the original view and yield an averaged edge prediction. The edge predictions are treated as additional evidence to prune out the outliers of line segments in the putative wireframe computed from the original unlabeled image.

the input image. The resulted pseudo wireframe labels will be used in training a HAWP model from scratch. We utilize the Homography Adaptation method [31] (Fig. 9) and resort to learn the edge map as cues in the "cleaning-up" process. This leads to our design of HAWPv3.

Our HAWPv3 is built on the HAWPv2 with a new sub-network added for high resolution predictions of edge and junction heatmaps as well as the junction offsets, denoted by $f_{\text{edge}}(F;\Omega_{\text{edge}})$, $f_{\text{jhm}}(F;\Omega_{\text{jhm}})$ and $f_{\text{joff}}(F;\Omega_{\text{joff}})$,

$$f_{\text{edge}}: \quad \text{ReLU} \circ \text{Conv}_{1\times1}^{\frac{C}{s^2}\to1} \circ \text{PixelShuffle}, \tag{20}$$

$$f_{\text{jhm}}: \quad \text{ReLU} \circ \text{Conv}_{1\times1}^{\frac{C}{s^2}\to1} \circ \text{PixelShuffle}, \tag{21}$$

$$f_{\text{joff}}: \quad \text{ReLU} \circ \text{Conv}_{1\times1}^{\frac{C}{s^2}\to2} \circ \text{PixelShuffle}, \tag{22}$$

where the PixelShuffle [53] operation rearranges elements in the feature map F of shape (C, H_s, W_s) to a feature map of shape $(\frac{C}{s^2}, H, W)$. Recall that s is the overall stride used in computing F, the predictions are at the same resolution as the input image (*e.g.*, 512×512). With the high resolution prediction of junctions, our HAWPv3 could handle localization errors better in practice.

The proposed HAWPv3 is first trained using synthetic data. For the edge branch, we use the balanced cross-entropy loss function in training, together with other loss functions defined in training HAWPv2.

Synthetic Pretraining of HAWPv3. We generate 2,000 images for each of the eight primitives (Fig. 8) and obtain 16,000 images in total. We train our HAWPv3 with 10 epochs. The learning rate is set to 4e-4 constantly over the entire training schedule. Compared to SOLD^2 [32] that used 10 times more synthetic images (*i.e.*, 160k images) and 200 epochs of training, HAWPv3 is much more efficient.

"Annotating" Real Images using HAWPv3 with Homography Adaptation. In Fig. 9, we use the synthetically trained HAWPv3 to annotate real images. Given an unlabeled real image I, we randomly warp it using N_h homographies ($N_h = 10$). The warped images are used to compute edge maps, which are then unwarped and averaged to generate the predicted edge map for I. From the original image I, we compute the putative wireframe. During verification, we combine the EPD LOIAlign score (c_i) and the edge map score (c_i') for each line segment (i). The edge map score is computed by sampling and averaging local maximal edge scores for 64 points on the line segment. The final SSL score for a line segment is the harmonic average, $c_i^{SSL} = \sqrt{c_i \cdot c'i}$. We prune line segments with scores below the threshold $\tau_{SSL} = 0.75$ in the putative wireframe. Unlike SOLD^2 [32], which uses adapted edge maps and end-point heatmaps, we only use the former. We employ 20k training images for SSL, generated from the original wireframe dataset [25] by flipping horizontally, vertically, and diagonally.

Training HAWPv3 with Pseudo Wireframe Labels. With the pseudo wireframe labels, training HAWPv3 is supervised learning again, as done in training our HAWPv2. We train a new HAWPv3 model from scratch with 30 epochs following the same settings as HAWPv2. We do not use the model weights from the synthetic pretraining due to the semantic gap between the synthetic data and real images. We notice that the process of "annotating" real images can be applied in a cascade manner, that is to use the pseudo-wireframe trained HAWPv3 to re-compute putative wireframes and edge maps to update/refine pseudo wireframes for a new round of SSL (see Sec. 7 and the Appx. C for the experimental results and detailed analyses).

6 Experiments of HAWPv2

In this section, we test the proposed HAWPv2 and compare it with state-of-the-art line segment detectors and wireframe parsers in the fully supervised learning (FSL) setting. All methods are trained in the Wireframe training dataset [25], and tested with the testing samples (462 images in total) of the Wireframe dataset [25] and the entire YorkUrban dataset (102 images in total) [33]. We also present ablation studies to verify the design of the proposed HAWPv2.

6.1 Evaluation Metrics

Structural Average Precision (sAP). This is motivated by the typical AP metric used in evaluating object detection systems, and has been the most challenging metric for wireframe parsing and line segment detection (LSD). A counterpart of the Intersection-over-Union (IoU) overlap is used. For each ground-truth line segment $\ddot{l} = (\mathbf{x}_1, \mathbf{x}_2)$, we first find the set of parsed line segments each of which, $\hat{\ddot{l}} = (\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2)$, satisfies the "overlap",

$$\min_{(i,j)} \|\mathbf{x}_1 - \hat{\mathbf{x}}_i\|^2 + \|\mathbf{x}_2 - \hat{\mathbf{x}}_j\|^2 \le \vartheta_L, \tag{23}$$

where (i, j) can be either $(1, 2)$ or $(2, 1)$, and ϑ_L is a predefined threshold. If no overlap is found, it is a False Negative (FN). If multiple candidates exist, the one with the highest verification score is a True Positive (TP), and the rest are False Positives (FP). Unmatched parsed line segments are also counted as FPs. We follow the convention used in

TABLE 1
Quantitative results and comparisons. Our proposed HAWPv2 sets new records in the most challenging metrics of structural correctness. For heatmap-based evaluation results, our proposed HAWPv2 obtains better average precision and performance comparable to heatmap-based F scores. In the last column, the inference speed is compared for all learning-based approaches. The numbers with * are extracted from the original paper. The best scores are highlighted in **bold fonts**.

	Image Size	# Epochs	*Wireframe Dataset*						*YorkUrban Dataset*						FPS
			sAP^5	sAP^{10}	sAP^{15}	mAP^J	AP^H	F^H	sAP^5	sAP^{10}	sAP^{15}	mAP^J	AP^H	F^H	
DWP [25]	320×320	200	3.7	5.1	5.9	40.9	67.8	72.2	1.5	2.1	2.6	13.4	51.0	61.6	2.24
AFM [37]	320×320	200	18.5	24.4	27.5	23.3	69.2	77.2	7.3	9.4	11.1	12.4	48.2	63.3	13.5
AFM++ [38]	512×512	200	27.7	32.4	34.8	30.8	74.8	**82.8**	9.5	11.6	13.2	TBD	50.5	**66.8**	5.2
L-CNN [30]	512× 512	30	59.7	63.6	65.3	60.2	81.6	77.9	25.0	27.1	28.3	31.5	58.3	62.2	15.6
F-Clip (HG2-LB) [27]	512× 512	300	62.6	66.8	68.7	48.7	85.1	80.9	27.6	29.9	31.3	28.3	62.3	64.5	28.3
LETR (R101) [28]	800 (short)	825	59.2	65.2	67.7	44.1	85.5	79.8	23.9	27.6	29.7	24.5	59.6	62.0	5.25
LETR (R50) [28]	800 (short)	825	58.5	64.6	67.3	44.2	84.7	79.1	25.7	29.6	32.0	25.9	61.7	63.4	5.25
ELSD (HG) [45]	512× 512	170	62.7*	67.2*	69.0*	N/A	84.7*	80.3*	23.9*	26.3*	27.9*	N/A	57.8*	62.1*	47*
ELSD (Res34) [45]		170	64.3*	68.9*	70.9*	N/A	87.2*	82.3*	27.6*	30.2*	31.8*	N/A	62.0*	63.6*	42.6*
HAWPv1 (Ours) [23]	512× 512	30	62.5	66.5	68.2	60.2	84.5	80.3	26.1	28.5	29.7	31.6	60.6	64.8	29.5
HAWPv2 (Ours)		30	**65.7**	**69.7**	**71.3**	**61.8**	**88.0**	81.4	**28.8**	**31.2**	**32.6**	**32.5**	**64.6**	64.5	40.8

previous methods to resize predictions and grountruth wireframes to 128×128, and report sAP scores with thresholds ϑ of $5, 10, 15$, *i.e.*, sAP^5, sAP^{10}, and sAP^{15}, respectively.

Heatmap based F score and Average Precision. These are traditional metrics used in LSD and wireframe parsing [25]. Instead of directly using the vectorized representation of line segments, heatmaps are used, which are generated by rasterizing line segments for both parsing results and the ground truth. The pixel-level evaluation is used to calculate the precision and recall curves with which the heatmap F score, indicated by F^H and the heatmap average precision, indicated by AP^H are computed. Unlike the evaluation protocol that computes the F scores and AP by averaging the per-image evaluation result in [25], [37], [38], we follow L-CNN [30] to first calculate the true positive and false negative edge pixels over the entire dataset and then compute the F scores and AP.

Vectorized Junction Mean AP. It is calculated in a similar way to the sAP of line segments. Let ϑ_J be the threshold for the distance between a predicted junction and a ground truth one. The mAP^J is computed *w.r.t.* $\vartheta_J = 0.5, 1.0, 2.0$. For the line segment detection approaches compared that did not yield junctions, we take the endpoints as the detected junctions for evaluation as in L-CNN [30].

6.2 Main Comparisons with State of the Arts

6.2.1 Baselines

We compare with the recent deep learning based approaches which are summarized as follows:

1) *In 2018*, the Wireframe dataset [25] was proposed with a baseline wireframe parser, DWP, which groups the learned junctions and line heatmaps into vectorized wireframe graphs. We use the DWP as the earliest baseline for wireframe parsing.
2) *In 2019*, there are several line segment detectors including AFM [37] and AFM++ [38], PPG-Net [39] and L-CNN [30]. The AFM approaches [37], [38] are not fully end-to-end, while PPG-Net [39] and L-CNN [30] use neural networks to obtain the final wireframe graphs end-to-end. We compare our HAWPv2 with the AFM approaches [37], [38] and L-CNN [30].
3) *After 2019*, many works focused on the direct regression of line segments by learning center-based representations or exploiting Attention mechanisms in line segment detection. All these approaches require long learning schedules with hundreds of training epochs. We choose the best performing approaches, ELSD [45], F-Clip [27] and LETR [28] for comparison.

6.2.2 The Results

We present the quantitative evaluation results using metrics including sAP^5, sAP^{10}, sAP^{15}, AP^H, and F^H for detected line segments, as well as mAP^J for detected junctions (Tab. 1). Precision-recall curves for sAP^5 and heatmap-based metrics are plotted in Fig. 10.

Compared to L-CNN [30] and HAWPv1 [23], our proposed HAWPv2 achieves significant improvements in challenging metrics such as sAP with varying strictness. HAWPv2 outperforms HAWPv1 by 3.2 and 2.7 points in sAP^5 on the Wireframe dataset and the YorkUrban dataset, respectively. Compared to F-Clip [27] and ELSD [45], both employing direct regression methods with the Stacked Hourglass Network backbone, HAWPv2 outperforms them by at least 3 points in sAP^5, the strictest metric. Despite F-Clip and ELSD using larger backbones or advanced design techniques, HAWPv2 still surpasses them. Moreover, HAWPv2 requires significantly fewer training epochs, being 5.67 times less than ELSD and 10 times less than F-Clip.

In terms of heatmap-based evaluation, HAWPv2 demonstrates superior performance on the Wireframe and YorkUrban datasets compared to other approaches. However, AFM++ [38] remains the best approach with F^H scores of 82.8 and 66.8 on these two datasets.

We also conducted a qualitative comparison between HAWPv2, AFM++ [38], HAWPv1 [23], F-Clip [27], and LETR [28]. As shown in Fig. 11, HAWPv2 achieves overall better results compared to other methods. F-Clip occasionally produces erroneous line segments (*e.g.*, second row), while HAWPv1 and HAWPv2 exhibit greater stability. HAWPv2, with improved visualization accuracy, outperforms HAWPv1 in sAP metrics. While AFM++ does not achieve competitive sAP scores due to its heuristic post-processing, it still produces good visualization results, albeit with localization issues in line segment endpoints.

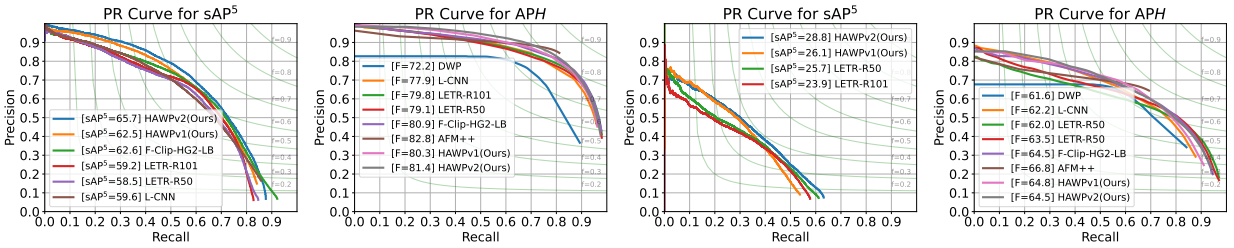


Fig. 10. Precision-Recall Curves for sAP^5 and AP^H on the Wireframe dataset (in the top row) and the YorkUrban dataset (in the bottom row)

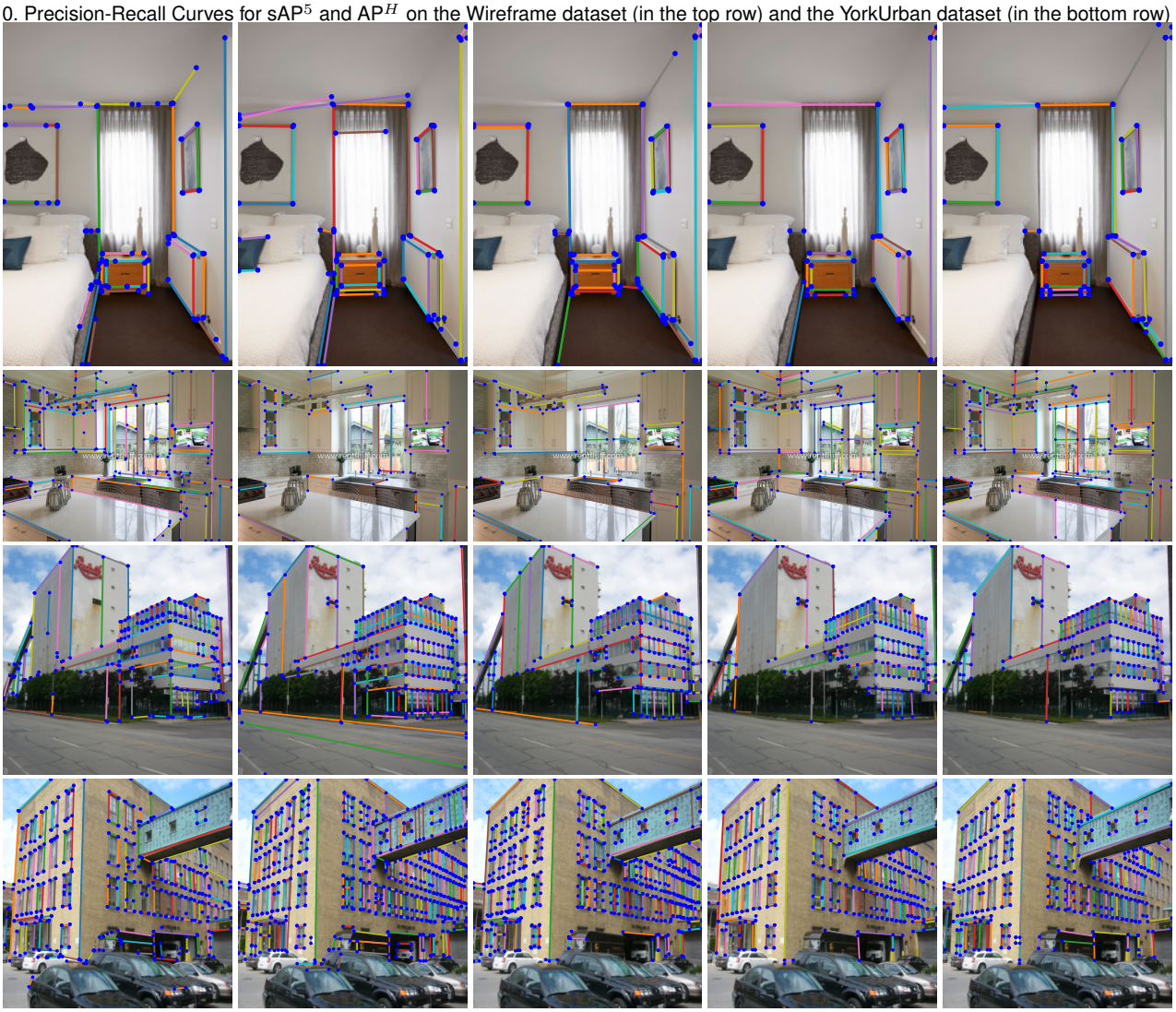


Fig. 11. Qualitative comparisons between AFM++ [38], F-Clip [27], LETR [28], our HAWPv1 [23] and HAWPv2 on the Wireframe dataset [25] (the top two rows) and the YorkUrban dataset [33] (the last two rows).

Regarding inference speed, HAWPv2 achieves 40.8 FPS on a single NVIDIA V100 GPU. The improved inference speed is primarily attributed to our HAT-driven design for proposal generation and verification, which replaces the computationally heavier LOIPooling on the high-dimensional feature map for a fewer number of high-quality line segment proposals.

6.3 Ablation Studies

In this section, we comprehensively evaluate the designs of the line segment proposal generation component (HAT field learning in Sec. 4.2 and endpoint binding in Sec. 4.4), as well as the line segment proposal verification component (EPD LOIAlign module) in Sec. 4.5. All ablation experiments utilize the Wireframe dataset for training and testing, following the settings described in Sec. 5.1. The challenging metrics of sAP^5, sAP^{10}, and sAP^{15} are used for evaluation.

Overall, the EPD LOIAlign module plays the major role in achieving the final performance gain of our HAWPv2 compared to the baselines. Endpoint error loss (Eqn. 7) in the HAT field learning and the binding threshold choices (τ_δ in Sec. 4.4) play a minor role in terms of final accuracy. However, they do play the main role in achieving the accuracy of the proposal, which significantly reduces

TABLE 2
The ablation study for the proposed EPD LOIAlign module.

End-Point Decoupled	Features of $\mathbf{Z}_{\psi_{\mathbf{x}}}(\hat{\hat{l}}_j)$ Initial Lines	AuxScore (Eqn. 19) BCE Loss	sAP^5	sAP^{10}	sAP^{15}
No	No	No	64.0	68.0	69.8
Yes	No	No	65.3	69.1	70.8
Yes	Yes	No	65.4	69.2	70.8
Yes	Yes	Yes	**65.7**	**69.7**	**71.3**

the number of proposals for verification by 20%, and thus boosts the inference speed. The expressive power of both EPD LOIAlign and the differentiable HAT field end-point error loss ensures overall effectiveness and efficiency in a disentangled way.

6.3.1 The Design of the End-Point Decoupled LOIAlign

We evaluate the effectiveness of the proposed EPD LOIAlign by comparing it to the baseline, vanilla LOIPooling [30]. We analyze three aspects: (1) endpoint-decoupled feature aggregation versus uniform feature aggregation; (2) inclusion or exclusion of features from intermediate points on the HAT field proposed line segments ($\mathbf{Z}_{\psi_{\mathbf{x}}}(\hat{\hat{l}}_j)$); and (3) presence or absence of the auxiliary score loss (Eqn. 19).

The results in Tab. 2 show that the EPD LOIAlign improves performance by 1.3, 1.1, and 1.0 points for sAP^5, sAP^{10}, and sAP^{15}, respectively. Additionally, incorporating features $\mathbf{Z}_{\psi_{\mathbf{x}}}(\hat{\hat{l}}_j)$ in line segment verification yields positive effects. The auxiliary classification task aids in learning better features $\mathbf{Z}_{\psi_{\mathbf{x}}}(\hat{\hat{l}}_j)$, resulting in performance improvements of 0.3, 0.5, and 0.5 points for sAP^5, sAP^{10}, and sAP^{15}, respectively. In App. B, we compare different feature map dimension C_ψ used for representing intermediate points in the EPD LOIAlign, and thus set $C_\psi = 4$ in our final models.

6.3.2 The Designs of Learning Line Segment Proposals

Line segment proposal quality is crucial for both accuracy and efficiency in wireframe parsing. We conduct detailed experiments on four aspects: (1) EPE loss (Eqn. 7) and binding threshold τ_δ (Sec. 4.4) in (2) training and (3) testing, respectively. Additionally, we verify (4) the effectiveness of residual learning for distance maps and rectified distance maps (Eqn. 6). Throughout these experiments, the EPD LOIAlign component remains unchanged. We compare overall accuracy performance and the average number of line segment proposals per image during inference. Additionally, we provide references to the average inference latency and breakdown profiling for binding and scoring, allowing further insight into the performance of the system.

Regarding the EPE loss and τ_δ settings, we observe that they do not significantly affect the final performance, as shown in Row 6 of Tab. 3. This is due to the expressive power of our EPD LOIAlign verification. However, the efficiency drops significantly due to the increased number of line segment proposals (6.19k vs. 2.24k in Row 1). This indicates that HAWPv2 maintains consistently high recall rates of line segments at the proposal generation stage and achieves significant precision improvements with EPE loss.

Residual Learning of Distance Maps. In Tab. 4, not learning distance residuals (Row 1) yields sAP scores of $\{61.1, 65.2, 67.1\}$ for different evaluation strictnesses. However, learning distance residuals as auxiliary supervision signals improves sAP scores by an average of 1.8 points across evaluation thresholds (Row 2), indicating enhanced accuracy in distance map learning. Using the learned distance residuals to generate three rectified distance maps (Eqn. 6) significantly improves performance by an average of 2.17 points (Row 3). For the most strict metric, sAP^5, learned distance residuals boost performance by 2.3 points. Modulating the residual scales by adding the modulated distance residuals with $2\times$ scales (-2 and +2) leads to the best precision for our HAWPv2 (Row 4).

Additionally, we compare our strategy for learning unsigned distance residuals with the signed version. We find that signed residual learning only marginally affects performance for different k values, while unsigned residual learning accurately captures the residual distance predictions despite the unknown sign. Unsigned residual learning can be seen as a form of uncertainty estimation rather than regression, while signed residual learning aligns with distance field prediction.

TABLE 3
The ablation study for the EPE loss and the specification of the line segment binding in training and testing.

	EPE Loss (Eqn. 7)	τ_δ (train)	τ_δ (test)	sAP^5	sAP^{10}	sAP^{15}	# Proposals
1	Yes	10	10	**65.7**	**69.7**	**71.3**	2.24k
2	Yes	10	∞	65.1	69.2	70.8	4.73k
3	Yes	∞	10	65.5	69.4	71.2	2.22k
4	Yes	∞	∞	65.7	69.6	71.3	4.61k
5	No	∞	10	65.3	69.4	71.1	2.79k
6	No	∞	∞	65.5	69.5	71.2	6.19k
7	No	10	10	65.6	69.5	71.2	2.83k
8	No	10	∞	65.1	69.0	70.7	6.32k

TABLE 4
Ablation study evaluating the impact of residual learning on distance maps and scale modulation during the testing phase, including average inference latency and breakdown profiling on a single V100 GPU for binding and scoring.

Learning Distance Residuals	# Residual Scales	sAP^5	sAP^{10}	sAP^{15}	# Proposals	Overall Latency	Binding	Scoring
No	N/A	61.1	65.2	67.1	894.84	21.821ms	5.948ms	1.876ms
Unsigned	0	62.9	67.1	68.8	931.35	22.096ms	6.046ms	2.016ms
	{-1,0,1}	65.2	69.2	70.9	1704.36	23.261ms	7.117ms	2.335ms
	{-2,-1,0,1,2}	65.7	69.7	71.3	2240.09	24.571ms	8.065ms	2.515ms
Signed	0	62.6	66.3	68.2	979.78	21.915ms	5.979ms	2.036ms
	{0,1}	62.7	66.8	68.6	995.13	22.568ms	6.625ms	2.036ms
	{0,1,2}	62.7	66.9	68.6	1010.23	23.047ms	6.990ms	2.043ms

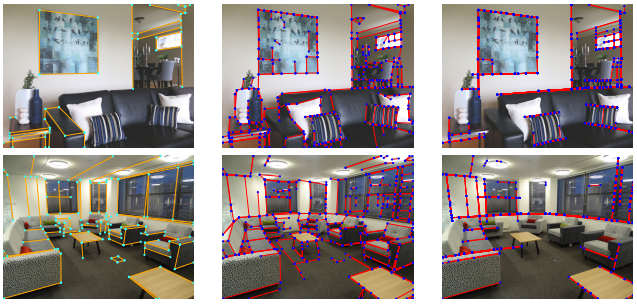
7 Experiments of HAWPv3

In this section, we test our HAWPv3 under the SSL setting. We also show the out-of-distribution (OOD) capability and potential of our HAWPv3.

7.1 Evaluation Protocol and Metrics

We follow the evaluation protocol presented in $SOLD^2$ [32]. In detail, we use the repeatability and the localization error as the main metrics across the original input images and the warped ones by the randomly generated homographies. The images in the Wireframe dataset [25] and YorkUrban dataset [33] are used for evaluation.

Repeatibility Scores and Localization Errors The metrics of repeatability and localization error were extensively used for keypoint detectors and line segment detectors. Given a pair of input images I and $I' = \text{Warp}(I|H)$, where



HAWPv2@0.95 *(26/68)* | HAWPv3@0.05 *(405/404)* | SOLD^2 [32]@0.05 *(642/443)*

Fig. 12. Parsing result comparisons between HAWPv2, HAWPv3 and SOLD^2 [32]. The numbers in the brackets are the number of parsed line segments for the two images, respectively, by each method.

$\text{Warp}(\cdot|H)$ is a homographic image warping function with homography $H \in \mathbb{R}^{3\times3}$, the repeatability score is calculated by checking if a line segment $\ddot{l}$ in the image is successfully detected again in the warped image up to a distance metric. Denoted by the line segment $\ddot{l} = (\mathbf{x}_1, \mathbf{x}_2)$ and the re-detected one $\ddot{l}' = (\mathbf{x}'_1, \mathbf{x}'_2)$, the structural distance (*i.e.*, the Euclidean endpoint distance)

$$d_s(\ddot{l}, \ddot{l}') = \frac{1}{2}\min(\|\mathbf{x}_1 - \mathbf{x}'_1\|_2 + \|\mathbf{x}_2 - \mathbf{x}'_2\|_2, \|\mathbf{x}_1 - \mathbf{x}'_2\|_2 + \|\mathbf{x}_2 - \mathbf{x}'_1\|_2), \tag{24}$$

and the orthogonal distance

$$d_{orth}(\ddot{l}, \ddot{l}') = \frac{1}{2}(\|\mathbf{x}_1 - p_{\ddot{l}'}(\mathbf{x}_1)\|_2 + \|\mathbf{x}_2 - p_{\ddot{l}'}(\mathbf{x}_2)\|_2 + \|\mathbf{x}'_1 - p_{\ddot{l}}(\mathbf{x}_1)\|_2 + \|\mathbf{x}'_2 - p_{\ddot{l}}(\mathbf{x}'_2)\|_2), \tag{25}$$

are used to compute the repeatability scores. In the orthogonal distance metric $d_o(\ddot{l}, \ddot{l}')$, the function $p_{\ddot{l}'}(\mathbf{x}_1)$ orthogonally maps the endpoint $\mathbf{x}_1$ into the line segment $\ddot{l}'$.

Using the distance metrics d_s and d_{orth}, we calculate the repeatability scores by counting the co-detected line segments in the image pair (I, I') and its swapped counterpart (I', I) among all detected line segments in the first image of the input pair. A distance threshold ε of 5 pixels is used. The localization error is then obtained by averaging the distance values over the pairly-detected line segments. We denote the repeatability and localization error for distance threshold ε as Rep_ε and Loc_ε, respectively. In contrast to the fully-supervised learning evaluation protocol that resizes predictions to a 128×128 image domain, Eq. (24) and Eq. (25) employ the ℓ_2 norm (without squaring) on the same domain as the input images for SSL evaluation.

Random Homography Generation We adopt the homography configurations from SOLD^2 to compute repeatability scores and localization errors. The random homography generator takes a patch ratio of 0.85 as input. The perspective displacement, left horizontal displacement, and right horizontal displacement values are obtained using a Gaussian noise generator with a perspective amplitude of 0.2 in both x and y directions, twice the standard deviation. The scaling matrix follows a zero mean Gaussian distribution with a standard deviation of 0.1. Random translations follow a uniform distribution within valid areas, and the rotation matrix follows a uniform distribution with rotation angles ranging from $-\pi/2$ to $\pi/2$.

Datasets Similar to the experiments of the FSL pipeline, we use the 462 test images of the Wireframe dataset and the 102 images of the YorkUrban dataset as image sources. For each original image, we generate two homographies and independently evaluate the repeatability scores and localization errors on these two datasets. So, there are 924 and 204 pair of images in the Wireframe and YorkUrban datasets for evaluation.

7.2 Comparisons with the State of the Arts

We comprehensively compare our proposed HAWPv3 model with the traditional LSD [36], the fully supervised approaches including L-CNN [30], DeepHough [54], TP-LSD [29], HAWPv1 [23] and HAWPv2, as well as the state-of-the-art SSL method, SOLD^2 [32].

Tab. 5 reports the results of the quantitative comparison. In terms of the structural distance metric d_s defined in Eq. (24), our HAWPv3 model improves the Rep-5 repeatability scores by 13.8 points at most while obtaining a localization error of 1.487 pixels on the Wireframe dataset compared to the state-of-the-art SOLD^2 without the candidate selection scheme. For the model SOLD2 with a candidate selection scheme (w/ CS), our HAWPv3 model obtains an improvement by about 20 points.

In terms of the orthogonal distance metric d_{orth}, as it is friendly to the overlapped detection results, the repeatability score of our HAWPv3 is worse than SOLD^2, placing it in the second place among all state-of-the-art approaches. For the localization error with d_{orth}, our HAWPv3 is still the best. Similar conclusions can be drawn from the YorkUrban dataset. Benefitting from the fast convergence speed of our HAWPv3, we are able to train it within 24 hours on a single GPU (Nvidia RTX 3090) from scratch to obtain very competitive performances. For detailed analysis of our HAWPv3, please refer to Appx. C.

Remarks on the FSL and SSL of Wireframe Parsing. For the FSL pipeline, wireframes are typically annotated in a viewpoint-specific manner based on given images, often consisting of long line segments. This poses a challenge for FSL wireframe parsers to achieve high repeatabilities across (warped) views, as viewpoint occlusions can break long line segments into shorter visible ones (see results in Tab. 5). To visually demonstrate this, we compare parsing results in Fig. 12 between our HAWPv2, HAWPv3, and SOLD^2. It is evident that many long line segments detected by HAWPv2 are "split" into several short line segments by HAWPv3 and SOLD^2. This difference highlights two key observations: (1) Applying an FSL wireframe parser directly to multiview tasks requiring correspondences across viewpoints can be more challenging compared to SSL wireframe parsers, and (2) SSL wireframe parsers should explore stronger SSL pretext tasks and more effective inductive biases that are learnable and transferable from simulation to reality, in addition to learned edge maps, to approach human perception of line segments. Between our HAWPv3 and the SOLD^2 [32] at a lower threshold of 0.05, our HAWPv3 can cover more geometry structures in the images than SOLD^2, while using a less number of, but longer, line segments. SOLD^2 often computes many overlapped co-linear line segments with

TABLE 5
The repeatability evaluation results. Numbers with **bold font** and underline indicate the best and second best performance on specific metrics. The image resolutions are fixed to 512×512 for both Wireframe and YorkUrban datasets in evaluation.

Method	Wireframe Dataset					YorkUrban Dataset				
	d_s		d_{orth}		#lines / image	d_s		d_{orth}		#lines / image
	Rep-5 ↑	Loc-5 ↓	Rep-5 ↑	Loc-5 ↓		Rep-5 ↑	Loc-5 ↓	Rep-5 ↑	Loc-5 ↓	
L-CNN [30]@0.98	0.434	2.589	0.570	1.725	76	0.318	2.662	0.449	1.784	103
DeepHough [54]@0.9	0.419	2.576	0.618	1.720	135	0.315	2.695	0.535	1.751	206
TP-LSD [29]TP512	0.547	2.479	0.695	1.474	77	0.447	2.491	0.610	1.491	130
LSD [36]	0.383	2.198	0.719	1.028	441	0.419	2.123	0.723	0.959	591
SOLD^2 [32] w/CS	0.566	2.039	0.805	1.135	116	0.585	1.918	0.824	1.097	196
SOLD^2 [32]	0.613	2.060	**0.921**	0.809	482.6	0.629	1.951	**0.939**	**0.693**	1031
HAWPv1 [23]@0.97	0.451	2.625	0.537	1.738	47	0.295	2.566	0.368	1.757	59
HAWPv2 (Ours)@0.9	0.514	2.375	0.577	1.548	34	0.385	2.205	0.425	1.397	30
HAWPv3 (Ours)@0.5	**0.751**	**1.487**	0.874	**0.841**	145	**0.711**	**1.454**	0.829	0.839	225

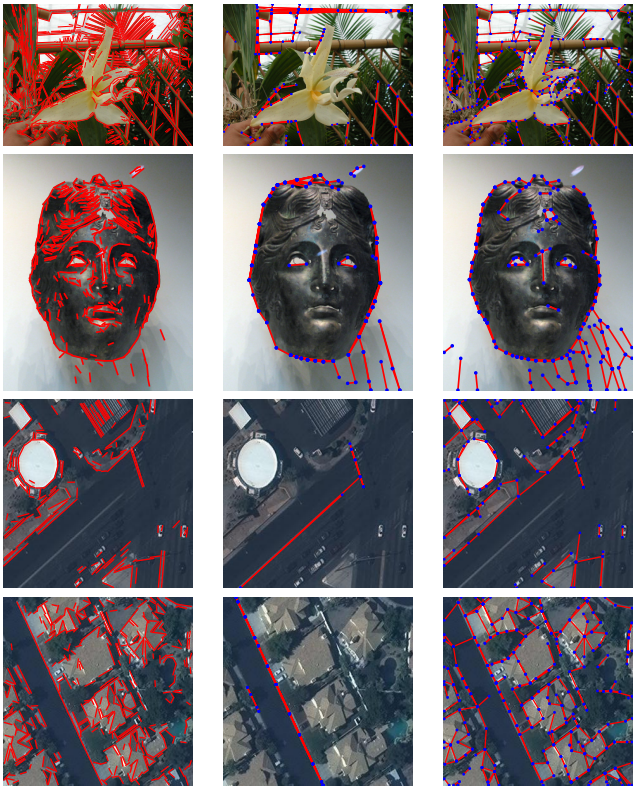


Fig. 13. Qualitative OOD comparisons between the LSD [12], the SOLD^2 [32] and our HAWPv3. The first two rows show results on images in the ImageNet [55] dataset, and the last two rows show results on images in the AICrowd [24] dataset.

different end-points, which contributes to its higher repeatability score, d_{orth}.

7.3 The OOD Potential of our HAWPv3

To achieve robust geometry understanding of the visual world, it is crucial to possess out-of-distribution (OOD) perception generalizability for low-level structures like line segments and junction points. We qualitatively tested this aspect and observed that our HAWPv3 model exhibits remarkable OOD potential (Fig. 13). We compared it with the training-free LSD method [36] and the SOLD^2 approach [32]. By utilizing SSL models trained on the wireframe dataset [25] and evaluating them on images from the ImageNet dataset [55] and the AICrowd dataset [24], which differ significantly from the indoor images in the wireframe dataset, we observed promising OOD results with our HAWPv3 model. These results offer valuable insights for the development of geometry-guided semi-supervised learning (SSL) approaches for downstream tasks such as image classification and object detection, as well as multi-view 3D vision problems including but not limited to Structure from Motion and SLAM.

8 Conclusion

This paper comprehensively studies the problem of wireframe parsing from the perspective of line segment representation, the algorithmic design of wireframe parsing, as well as the learning of wireframes in both supervised and self-supervised pipelines. With the proposed novel HAT field representation of line segments that has built-in geometry-awareness, context-awareness and robustness, the presented HAWP models set several new records on challenging benchmarks of Wireframe and YorkUrban while being efficient for training and inference. For the fully-supervised learning with precisely-annotated wireframe labels, the proposed HAWPv2 model is leading the performance on the Wireframe and YorkUrban datasets. For self-supervised learning, the proposed HAWPv3 model shows its advantages of learning the appropriate inductive biases of line segments via the HAT field representation. Besides the strong performance on the structural repeatability, the proposed HAWPv3 model is of great potential for general wireframe parsing in images out of the training distributions.

Acknowledgements

We acknowledge the efforts of the authors of the Wireframe dataset and the YorkUrban dataset. These datasets make accurate line segment detection and wireframe parsing possible. We thank the anonymous reviewers and associate editors for their constructive comments. We also thank Rémi Pautrat for helpful discussions, and Bin Tan for proofreading. This work was supported by the NSFC Grants under the contracts No. 62101390, 62325111 and U22B2011. T. Wu was partly supported by ARO Grant W911NF1810295, NSF IIS-1909644, ARO Grant W911NF2210010, NSF IIS-1822477, NSF CMMI-2024688 and NSF IUSE-2013451. The views presented in this paper are those of the authors and should not be interpreted as representing any funding agencies.

REFERENCES

[1] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 207, no. 1167, pp. 187–217, 1980.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[3] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image Vis. Comput.*, vol. 22, no. 10, pp. 761–767, 2004.

[4] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *Int. J. Comput. Vis.*, vol. 60, no. 1, pp. 63–86, 2004.

[5] C. G. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the Alvey Vision Conference (AVC)*, C. J. Taylor, Ed., 1988, pp. 1–6.

[6] N. Xue, G. Xia, X. Bai, L. Zhang, and W. Shen, "Anisotropic-scale junction detection and matching for indoor images," *IEEE Trans. Image Process.*, vol. 27, no. 1, pp. 78–91, 2018.

[7] G. Xia, J. Delon, and Y. Gousseau, "Accurate junction detection and characterization in natural images," *Int. J. Comput. Vis.*, vol. 106, no. 1, pp. 31–56, 2014.

[8] P. Arbelaez, M. Maire, C. C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, 2011.

[9] S. Xie and Z. Tu, "Holistically-nested edge detection," *Int. J. Comput. Vis.*, vol. 125, no. 1-3, pp. 3–18, 2017.

[10] L. Huan, N. Xue, X. Zheng, W. He, J. Gong, and G. Xia, "Unmixing convolutional features for crisp edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6602–6609, 2022.

[11] D. H. Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[12] 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.*, vol. 32, no. 4, pp. 722–732, 2010.

[13] L. Liu, H. Li, H. Yao, and R. Zha, "Pluckernet: Learn to register 3d line reconstructions," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021, pp. 1842–1852.

[14] W. Ma, B. Tan, N. Xue, T. Wu, X. Zheng, and G. Xia, "How-3d: Holistic 3d wireframe perception from a single image," in *Int. Conf. 3D Vis.*, 2022.

[15] B. Tan, N. Xue, S. Bai, T. Wu, and G. Xia, "Planetr: Structure-guided transformers for 3d plane recovery," in *Int. Conf. Comput. Vis.*, 2021, pp. 4166–4175.

[16] R. Gomez-Ojeda, F. A. Moreno, D. Zuñiga-Noël, D. Scaramuzza, and J. G. Jiménez, "PL-SLAM: A stereo SLAM system through the combination of points and line segments," *IEEE Trans. Robotics*, vol. 35, no. 3, pp. 734–746, 2019.

[17] Y. Li, J. Mao, X. Zhang, B. Freeman, J. Tenenbaum, N. Snavely, and J. Wu, "Multi-plane program induction with 3d box priors," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[18] Y. N. Wu, Z. Si, H. Gong, and S.-C. Zhu, "Learning active basis model for object detection and recognition," *Int. J. Comput. Vis.*, vol. 90, no. 2, pp. 198–235, 2010.

[19] L. Duan and F. Lafarge, "Image partitioning into convex polygons," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, 2015, pp. 3119–3127.

[20] J. Lazarow, W. Xu, and Z. Tu, "Instance segmentation with mask-supervised polygonal boundary transformers," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, 2022, pp. 4372–4381.

[21] G. Xia, J. Huang, N. Xue, Q. Lu, and X. Zhu, "Geosay: A geometric saliency for extracting buildings in remote sensing images," *Comput. Vis. Image Underst.*, vol. 186, pp. 37–47, 2019.

[22] B. Xu, J. Xu, N. Xue, and G. Xia, "Accurate polygonal mapping of buildings in satellite imagery," *CoRR*, vol. abs/2208.00609, 2022.

[23] N. Xue, T. Wu, S. Bai, F. Wang, G. Xia, L. Zhang, and P. H. S. Torr, "Holistically-attracted wireframe parsing," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020, pp. 2785–2794.

[24] S. P. Mohanty, J. Czakon, K. A. Kaczmarek, A. Pyskir, P. Tarasiewicz, S. Kunwar, J. Rohrbach, D. Luo, M. Prasad, S. Fleer, J. P. Göpfert, A. Tandon, G. Mollard, N. Rayaprolu, M. Salathé, and M. Schilling, "Deep learning for understanding satellite imagery: An experimental survey," *Frontiers Artif. Intell.*, vol. 3, p. 534696, 2020.

[25] K. Huang, Y. Wang, Z. Zhou, T. Ding, S. Gao, and Y. Ma, "Learning to parse wireframes in images of man-made environments," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 626–635.

[26] B. B. Averbeck, P. E. Latham, and A. Pouget, "Neural correlations, population coding and computation," *Nature reviews neuroscience*, vol. 7, no. 5, pp. 358–366, 2006.

[27] X. Dai, X. Yuan, H. Gong, and Y. Ma, "Fully convolutional line parsing," *NeuroComputing*, 2022.

[28] Y. Xu, W. Xu, D. Cheung, and Z. Tu, "Line segment detection using transformers without edges," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021, pp. 4257–4266.

[29] S. Huang, F. Qin, P. Xiong, N. Ding, Y. He, and X. Liu, "TP-LSD: tri-points based line segment detector," in *Eur. Conf. Comput. Vis.*, vol. 12372, 2020, pp. 770–785.

[30] Y. Zhou, H. Qi, and Y. Ma, "End-to-end wireframe parsing," in *Int. Conf. Comput. Vis.*, 2019, pp. 962–971.

[31] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, 2018, pp. 224–236.

[32] R. Pautrat, J. Lin, V. Larsson, M. R. Oswald, and M. Pollefeys, "SOLD2: self-supervised occlusion-aware line description and detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021, pp. 11 368–11 378.

[33] P. Denis, J. H. Elder, and F. J. Estrada, "Efficient Edge-Based Methods for Estimating Manhattan Frames in Urban Imagery," in *Eur. Conf. Comput. Vis.*, 2008, pp. 197–210.

[34] L. G. Roberts, "Machine perception of three-dimensional solids," Ph.D. dissertation, Massachusetts Institute of Technology, 1963.

[35] A. Gupta, A. A. Efros, and M. Hebert, "Blocks world revisited: Image understanding using qualitative geometry and mechanics," in *Eur. Conf. Comput. Vis.*, 2010, pp. 482–496.

[36] R. G. von Gioi, J. Jakubowicz, J. Morel, and G. Randall, "LSD: A fast line segment detector with a false detection control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 4, pp. 722–732, 2010.

[37] N. Xue, S. Bai, F. Wang, G. Xia, T. Wu, and L. Zhang, "Learning attraction field representation for robust line segment detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 1595–1603.

[38] N. Xue, S. Bai, F. Wang, G. Xia, T. Wu, L. Zhang, and P. H. S. Torr, "Learning regional attraction for line segment detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 6, pp. 1998–2013, 2021.

[39] Z. Zhang, Z. Li, N. Bi, J. Zheng, J. Wang, K. Huang, W. Luo, Y. Xu, and S. Gao, "Ppgnet: Learning point-pair graph for line segment detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 7105–7114.

[40] Q. Meng, J. Zhang, Q. Hu, X. He, and J. Yu, "LGNN: A context-aware line segment detector," in *ACM Int. Conf. Multimedia*, 2020, pp. 4364–4372.

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Adv. Neural Inform. Process. Syst.*, vol. 30, 2017.

[42] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Eur. Conf. Comput. Vis.*, 2020, pp. 213–229.

[43] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *CoRR*, vol. abs/1904.07850, 2019.

[44] G. Gu, B. Ko, S. Go, S.-H. Lee, J. Lee, and M. Shin, "Towards light-weight and real-time line segment detection," in *AAAI*, 2022.

[45] H. Zhang, Y. Luo, F. Qin, Y. He, and X. Liu, "ELSD: efficient line segment detector and descriptor," in *Int. Conf. Comput. Vis.*, 2021, pp. 2949–2958.

[46] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020, pp. 9729–9738.

[47] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022, pp. 16 000–16 009.

[48] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Int. Conf. Comput. Vis.*, 2017, pp. 764–773.

[49] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2018.

[50] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.

[51] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *Int. Conf. Comput. Vis.*, 2019, pp. 6568–6577.
[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[53] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 1874–1883.
[54] Y. Lin, S. L. Pintea, and J. C. van Gemert, "Deep hough-transform line priors," in *Eur. Conf. Comput. Vis.*, vol. 12367, 2020, pp. 323–340.
[55] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2009, pp. 248–255.
[56] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Eur. Conf. Comput. Vis.*, ser. Lecture Notes in Computer Science, vol. 9912, 2016, pp. 483–499.

APPENDIX A
NETWORK ARCHITECTURES

To facilitate real-time inference speed, we utilize light-weight neural building blocks for different components in the HAWP models. We summarize them as follows. More details are referred to our open-sourced code.

- For the image feature backbone $f_b(\cdot;\Omega_b)$, we use the Stacked Hourglass Networks [56] (2 stacks). The dimension of the output feature map F is $C = 256$, and the overall stride $s = 4$.
- In learning the HAT field (Sec. 4.2), we use a vanilla convolution block configuration for $f_d(F;\Omega_d)$ (Eqn. 3), $f_a(F;\Omega_a)$ (Eqn. 4) and $f_{\Delta d}(F;\Omega_{\Delta d})$ (Eqn. 5).

$$f_d, f_{\Delta d}: \quad \sigma \circ \mathrm{Conv}_{1\times1}^{D\to1} \circ \mathrm{ReLU} \circ \mathrm{Conv}_{3\times3}^{C\to D}, \quad (26)$$
$$f_a: \quad \sigma \circ \mathrm{Conv}_{1\times1}^{D\to3} \circ \mathrm{ReLU} \circ \mathrm{Conv}_{3\times3}^{C\to D}, \quad (27)$$

  where $\sigma(\cdot)$ is the sigmoid function, $\mathrm{Conv}_{k\times k}^{u\to v}$ denotes the convolution layer that transforms a u-channel feature map to a v-channel one using a $k \times k$ kernels. In our experiments, we set $D = 128$.
- In learning the heatmap-offset field of endpoints (Sec. 4.3), $f_{pt}(F;\Omega_{pt})$ (Eqn. 8) and $f_o(F;\Omega_o)$ (Eqn. 9) are implemented by $\mathrm{Conv}_{1\times1}^{C\to1}$ and $\mathrm{Conv}_{1\times1}^{C\to2}$ respectively.
- In computing features for the intermediate points on a line segment proposal in the proposed EPD LOIAlign (Sec. 4.5), $f_\psi(F;\Omega_\psi)$ is implemented by $\mathrm{ReLU} \circ \mathrm{Conv}_{3\times3}^{C\to C_\psi}$, where $C_\psi = 4$ in our experiments.
- The MLPs used in the verification head classifier (Eqn. 18) are implemented by two hidden layers with the ReLU nonlinearity function.

APPENDIX B
THE LOW-DIMENSIONAL FEATURE MAPS F_ψ IN THE EPD LOIALIGN VERIFICATION

In this ablation study following those in Sec. 6.3.1, we show the performance differences in terms of the dimensions C_ψ's of computing the "thin" feature maps F_ψ. As reported in Tab. 6, when the number of feature channels is set to $C_\psi = 4$, our proposed HAWPv2 model obtains the best performance in terms of accuracy. For the model that uses $C_\psi = 1$, the performance of accuracy is degenerated by 1.6% in terms of sAP^5. When we increase the channels from 4 to 8, there is no performance gain in both aspects of accuracy and speed. Overall, since the used feature maps in comparisons are all sufficiently "thin", the inference speed change with respect to the number of feature channels is less apparent.

TABLE 6
The performance change by the different number of feature channels for the thin feature maps.

Feat. Dim. C_ψ	sAP^5	sAP^{10}	sAP^{15}	FPS
1	64.1	68.1	69.8	42.5
2	64.6	68.6	70.2	41.4
4	**65.7**	**69.7**	**71.3**	40.8
8	65.6	69.3	71.0	40.0

APPENDIX C
MORE DETAILED ANALYSES OF HAWPV3

TABLE 7
The quantative evaluation results of the synthetically trained models, SOLD^2 and our HAWPv3. All the evaluation results are obtained with the detection threshold of 0.5 and inlier threshold of 0.75.

		SOLD^2	SOLD^2 (w/ CS)	HAWPv3
d_s	Rep-5 ↑	0.306	0.282	**0.356**
	Loc-5 ↓	2.697	**2.636**	2.912
d_{orth}	Rep-5 ↑	0.573	0.384	**0.629**
	Loc-5 ↓	**1.233**	1.367	1.890
#lines/image		310	95	170

Evaluation of Synthetically-Trained Models As the self-supervised wireframe parsers spring from the synthetically-trained models, we evaluate our HAWPv3 and SOLD^2 [32] before adopting the Homography Adaptation learning scheme on the real-world images. We use the metrics of repeatability scores and localization errors as in Sec. 7.2 on the Wireframe dataset. As reported in Tab. 7, our proposed HAWPv3 model that is trained only using the synthetic data achieves better repeatability scores for both d_s and d_{orth} distance metrics than SOLD^2 and SOLD^2 (w/CS). For the localization error, SOLD^2 obtains the best on d_{orth} metric and the candidate selection (w/ CS) scheme improves the localization error on the d_s metric. Although the localization error by our HAWPv3 is larger than SOLD^2, our final model is significantly better than SOLD^2 (see Tab. 5).

Iterative Learning and Refinement (Cascade SSL) Thanks to the fast pseudo wireframe label generation and the efficient training schedule of our proposed HAWPv3, we have the computational budget for iteratively training the HAWPv3 model with more and more accurate pseudo wireframe labels (*i.e.*, cascade SSL). In this ablation study, we run several experiments to study the possible settings of the iterative learning and refinement: (1) Random model initialization v.s. Warmp-up model initialization. (2) The training schedules in terms of the initial learning rate, the number of epochs and the learning rate decay milestone(s). In the comparisons, we denote the HAWPv3 models by $\texttt{HAWPv3}^{(k)}$ as the model trained with the pseudo labels generated by the $\texttt{HAWPv3}^{(k-1)}$ ($k \geq 1$). $\texttt{HAWPv3}^{(0)}$ is the synthetically-pretrained model and used in generating the

TABLE 8
The ablation study on the settings of iteratively training HAWPv3 models. It is done based on the repeatibility and localization performance on the Wireframe dataset. For the training schedule, we denote its initial learning rate, the total training epochs and the decay milestone by `lr/epochs/milestone`.

Model Name	Init. Type	Sched.	d_s		d_{orth}		#lines /image
			Rep-5↑	Loc-5↓	Rep-5↑	Loc-5↓	
HAWPv3$^{(1)}$-A*	Random	4e-4/30/25	0.691	1.672	0.806	0.944	104
HAWPv3$^{(1)}$-B	HAWPv3$^{(0)}$	4e-4/30/25	0.610	1.790	0.762	0.946	101
HAWPv3$^{(2)}$-A	Random	4e-4/30/25	0.700	1.504	0.833	0.844	147
HAWPv3$^{(2)}$-B	HAWPv3$^{(1)}$-A	4e-4/30/25	0.707	1.569	0.835	0.888	141
HAWPv3$^{(2)}$-C*	HAWPv3$^{(1)}$-A	4e-5/30/25	0.751	1.487	0.874	0.841	145
HAWPv3$^{(3)}$-A	HAWPv3$^{(2)}$-C	4e-5/30/-	0.741	1.509	0.883	0.847	166
HAWPv3$^{(3)}$-B	HAWPv3$^{(2)}$-C	4e-6/30/25	0.739	1.503	0.879	0.844	161

initial pseduo wireframe labels for real images. The results are summarized in Tab. 8. We have the observations as follows.

First, at the first stage of the cascade SSL, we do not need to use the synthetically pre-trained model weights to warm up the model to be trained on the real images with the pseudo-wireframe labels. With the same training schedule, HAWPv3$^{(1)}$-A outperforms HAWPv3$^{(1)}$-B by large margins, which can be intuitively understood based on the semantic gap between synthetic images and real images. Compared to SOLD^2 (see Tab. 5), HAWPv3$^{(1)}$-A obtains an absolute improvement by 9.6 points on the repeatability score.

Second, after the first stage of the cascade SSL, the real image-trained HAWPv3 model can be leveraged in warming up the model weights in the next stage, and better performance can be achieved with a more conservative initial learning rate. With the same training schedule, HAWPv3$^{(2)}$-A and HAWPv3$^{(2)}$-B have very similar performance. With a reduced initial learning rate, HAWPv3$^{(2)}$-C improves the repeatability scores by 3 points and 4.4 points for the two distance metrics respectively, while reducing the localization errors by about 14% for both d_s and d_{orth}. Compared with HAWPv3$^{(1)}$ models, the average number of SSL "annotated" line segments is significantly increased (*e.g.*., 98 vs. 134). Note that the overall training cost of HAWPv3$^{(2)}$-C is still significantly less than that used by the SOLD^2 [32].

Last but not least, the potential of the cascade SSL is quickly saturated, which is reasonably expected due to the essence of the simulation-to-reality SSL pipeline. On top of the model weights of HAWPv3$^{(2)}$-C, we do not observe further improvement with different training schedules in HAWPv3$^{(3)}$-A and HAWPv3$^{(3)}$-B. For more training rounds with lower learning rates (*e.g.*, 4e-7), we also did not observe any improvement. We did not explore the mixed training in which both the synthetic training data and the SSL "annotated real data are used.

APPENDIX D
SSL EXTENSIBILITY FOR HAWPV1 AND HAWPV2

Because the architecture of HAWPv3 is primarily based on HAWPv2, which has demonstrated outstanding performance in fitting human annotation wireframe labels, a natural question arises: "Is it necessary to upgrade HAWPv1 to HAWPv2 first in order to obtain HAWPv3 in self-supervised learning?" To quantitatively address this question, we adapt the HAWPv1 model by adding a high-resolution edge map learning branch and train a series of models to evaluate repeatability under the SSL evaluation protocol. We refer to the model adapted from HAWPv1 as HAWPv3†. As shown in Table 9, HAWPv3† achieves inferior performance at all learning stages. In the synthetic stage, the repeatability scores on the wireframe dataset decrease by 11 points and 7.2 points in terms of d_s and d_{orth}, respectively. During the first round of learning with real-world images, further performance degradation of HAWPv3† is observed compared to HAWPv3, particularly in terms of localization errors. These results indicate that achieving similar or better performance than SOLD^2 with the HAWPv1 architecture would be challenging, further justifying our design rationale for developing HAWPv2 to enable self-supervised learning.

TABLE 9
The ablation study on the model architecture designs for self-supervised learning. HAWPv3 inherits the main design of HAWPv2 while HAWPv3† comes from HAWPv1 [23].

Model Name	Learning Stage	d_s		d_{orth}		#lines /image
		Rep-5↑	Loc-5↓	Rep-5↑	Loc-5↓	
HAWPv3	Synthetic	**0.388**	**2.863**	**0.608**	**1.887**	76
HAWPv3†		0.278	3.429	0.536	2.439	89
HAWPv3	Real (Round 1)	**0.691**	**1.672**	**0.806**	**0.944**	104
HAWPv3†		0.541	3.054	0.772	1.895	178
HAWPv3	Real (Round 2)	**0.751**	**1.487**	**0.874**	**0.841**	145