Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

XPLNet: an accelerated unified point-line detection network

Chenhao Jia
Shanghai Jiao Tong University
Pingfang Zhou
zhoupf@sjtu.edu.cn

Shanghai Jiao Tong University

Research Article

Keywords: Line Detection, Wireframe Parsing, Point-Line Detection, UNet

Posted Date: May 26th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9773062/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

XPLNet: an accelerated unified point-line detection network

ChenHao Jia[1] and PingFang Zhou[1*]

[1]School of Aeronautics and Astronautics, Shanghai Jiao Tong University, Shanghai, 200240, China.

*Corresponding author(s). E-mail(s): zhoupf@sjtu.edu.cn;
Contributing authors: davidjia@sjtu.edu.cn;

Abstract

Real-time and robust point-line feature extraction is important for visual perception and Simultaneous Localization and Mapping (SLAM) front-ends. Existing unified point-line networks reduce redundant computation by jointly detecting keypoints and line segments, but they still rely on relatively heavy backbones and decoding structures, resulting in considerable inference latency under resource-constrained settings. This paper proposes XPLNet, an accelerated unified point-line detection network with selective point-prior fusion. XPLNet adopts Accelerated Features (XFeat) as a lightweight point-prior extractor and introduces a Point Prompted Interaction module to inject point reliability cues only into the junction proposal branch, where point responses are geometrically consistent with line endpoints and junctions. A gated multi-source fusion module and a compact Light UNet with a Transformer-Strip bottleneck are further designed to improve structural feature decoding with reduced computational cost. Experiments on the Wireframe and YorkUrban datasets show that XPLNet achieves a favorable trade-off between accuracy and efficiency. On the Wireframe dataset, XPLNet obtains 66.3 $\mathrm{sAP}^{\mathbf{10}}$ while running at 155.36 FPS. Compared with Point-Line Net (PLNet), XPLNet improves inference speed by approximately 46.0% while maintaining competitive structural accuracy. These results demonstrate that XPLNet provides an efficient unified point-line detection front-end, showing its potential for real-time lightweight visual perception and SLAM systems.

Keywords: Line Detection, Wireframe Parsing, Point-Line Detection, UNet

1 Introduction

Real-time and robust geometric feature extraction is a fundamental component of visual perception and SLAM front-ends. In practical robotic applications, such as UAV-based SLAM systems, perception modules are often deployed under limited computational, memory, and power budgets, while facing challenging imaging conditions such as illumination changes, low-texture regions, motion blur, and partial occlusions. These constraints require feature extraction networks to achieve a favorable balance between accuracy, robustness, and runtime efficiency. Point-line SLAM [1] has emerged to address the limitations of single-feature SLAM by leveraging the complementary nature of point and line geometric primitives.

However, traditional point-line SLAM frameworks treat point detection and line extraction as two independent modules, leading to substantial computational overhead. Such heavy computation severely limits systems deployment on resource-constrained platforms. To address this issue, unified deep neural networks capable of simultaneously extracting discrete point features and continuous line features in a single forward pass have become a highly favorable solution. This integrated design eliminates redundant calculations inherent to separate model inference and further facilitates synergistic learning between heterogeneous geometric features.

Existing joint extraction frameworks, such as AirSLAM/PLNet [2], have demonstrated remarkable success by adopting shared convolutional backbones (SuperPoint [3]) followed by HAWP's [4] multi-task decoupling heads. However, these methods are built upon early-generation VGG-like backbone designs. Although such architectures can provide reasonable feature quality, they are typically parameter-heavy and computationally expensive, leading to substantial memory consumption and inference latency. These limitations make it difficult to satisfy the real-time requirements of resource-constrained platforms, where onboard computation, memory, and power budgets are highly constrained. As a result, despite their effectiveness, existing methods remain insufficiently suited for resource-limited aerial applications.

Recently, modern local feature extractors such as XFeat [5] have significantly advanced point matching by combining lightweight network design, highly discriminative dense representations, coarse-to-fine keypoint localization, and efficient descriptor learning within a real-time inference pipeline. These properties make it particularly attractive for resource-constrained platforms, where both robustness and computational efficiency are critical. Therefore, it is natural to expect that incorporating such state-of-the-art light weight point feature backbones into a unified point-line extraction framework by sharing a same backbone would further enhance overall performance. However, our preliminary experiments reveal that by following the integration method in PLNet did not produce the anticipated synergistic gain. Instead, it causes severe precision degradation. This result suggests that simply replacing an early-generation point backbone with a stronger modern alternative is insufficient, and may even introduce structural incompatibilities between point-oriented feature representations and line-aware decoding objectives.

To address these challenges, we revisit the intrinsic relationship between point extraction and line extraction.

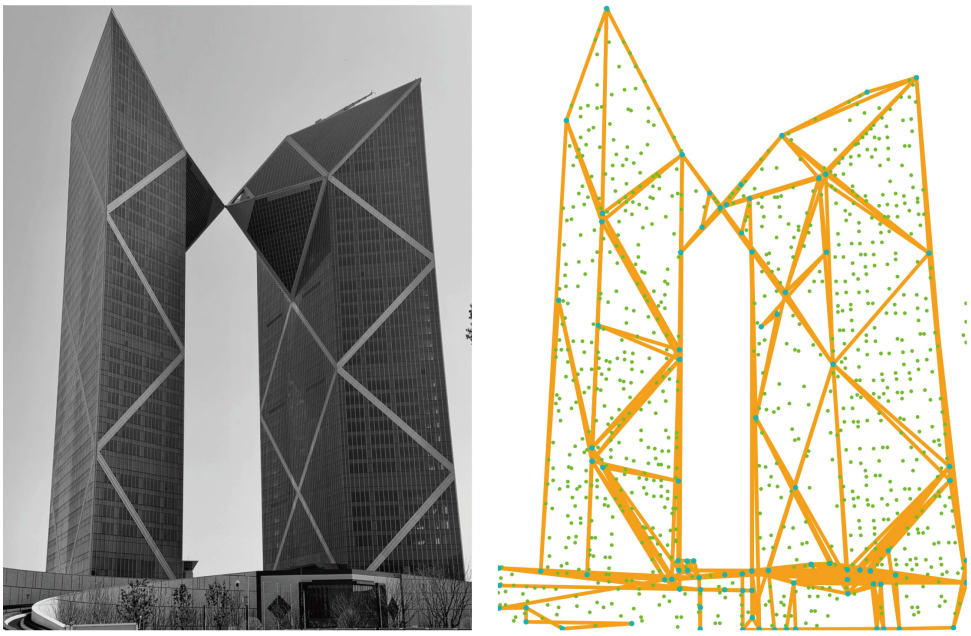

Fig. 1 Visualization of line segment and keypoint detection results produced by XPLNet

On the one hand, the two tasks are naturally complementary: keypoints often provide reliable cues for line endpoints and junction localization. On the other hand, they fundamentally differ in representation space. Point extraction focuses on sparse discrete structures, whereas line extraction requires continuous geometric reasoning over extended spatial support. Due to this discrepancy, naive feature fusion may disturb the structural continuity required for line parsing and even degrade detection quality.

Motivated by these observations, we propose a dedicated fusion framework that achieves high computational efficiency by fully exploiting point priors while retaining superior line detection performance.

As illustrated in Fig. 1, our proposed XPLNet can extract structurally consistent line segments and keypoints from complex scenes, enabling unified point-line representation that integrates global structural priors and local discriminative details. Specifically, rather than embedding point features indiscriminately into the entire line detection pipeline, we selectively inject XFeat-derived point priors only into the junction proposal branch where point and line geometric cues are inherently compatible and highly correlated. This elaborate design fully preserves the continuous geometric modeling capability of the line parsing module, while efficiently leveraging the precise localization superiority of point features to boost overall feature extraction efficiency.

To further balance efficiency and performance, we reuse the intermediate representations generated by the keypoint detection module and redesign the stacked UNet

decoder adopted in the original PLNet. By replacing the heavy multi-stage decoder with a single lightweight UNet, the proposed method significantly improves inference speed while maintaining competitive detection accuracy. Consequently, XPLNet achieves superior runtime efficiency among existing deep-learning-based point-line extraction methods, making it more suitable for real-time applications.

The main contributions of this paper are summarized as follows:

1. We propose an asymmetric point-line interaction strategy for unified point-line detection. Instead of treating point and line features as fully symmetric representations, XPLNet selectively introduces XFeat-derived point priors into the junction proposal branch, where keypoints, endpoints, and junctions are geometrically aligned. This design exploits point localization cues while reducing their interference with continuous line geometry modeling.
2. We design a gated multi-source feature construction module that reuses intermediate representations from a lightweight XFeat branch and shallow image-domain features. The branch-wise gating mechanism adaptively recalibrates heterogeneous feature sources before fusion, enabling efficient feature sharing without relying on a heavy point-oriented backbone.
3. We introduce a compact Light UNet decoder with a Transformer-Strip bottleneck for efficient line structure decoding. Compared with the heavier decoder used in previous unified point-line frameworks, the proposed decoder reduces computational overhead while preserving the multi-scale and long-range structural modeling ability required for line segment detection.
4. We evaluate XPLNet on Wireframe and YorkUrban benchmarks. Experimental results show that XPLNet achieves a favorable accuracy-efficiency trade-off, improving inference speed over PLNet while maintaining competitive structural detection performance.

The key distinction between XPLNet and existing unified point-line networks lies in its selective and asymmetric treatment of point priors. Rather than using modern point features as a complete replacement for line-aware representations, XPLNet treats them as auxiliary priors and introduces them only at geometrically compatible stages, namely junction and endpoint reasoning. This design differs from generic feature concatenation or attention-based fusion, because the interaction location is determined by the structural relationship between points, junctions, and line segments.

2 Related work

2.1 Keypoint and Line Detection

2.1.1 Keypoint Detection

Local keypoint detection has long been a fundamental component of visual perception and SLAM systems. Early handcrafted methods, such as SIFT [6] and ORB [7], detect salient keypoints based on gradient or intensity variations and construct local descriptors for matching. Owing to their simplicity and efficiency, these methods have been widely adopted in real-time visual SLAM. However, their performance often degrades

under severe illumination changes, repetitive textures, motion blur, and weak-texture regions.

To overcome the limitations of handcrafted features, learning-based keypoint detectors and descriptors have been extensively studied. Among them, SuperPoint [3] established a highly influential paradigm by jointly learning interest point detection and local description in a fully convolutional manner. Its self-supervised training strategy and end-to-end dense prediction design make it particularly suitable for geometric vision tasks, and it has therefore been widely adopted in point-based or point-line SLAM systems. Nevertheless, SuperPoint is still built upon an early VGG-style backbone, whose representational efficiency and runtime characteristics are no longer optimal for resource-constrained platforms.

Recent studies have therefore shifted toward more lightweight and efficient learned local features. ALIKED [8], as an improved lightweight keypoint and descriptor extraction network, introduces a sparse deformable descriptor head to construct adaptive descriptors only at detected sparse keypoints rather than densely over the entire feature map. This design improves descriptor efficiency while preserving strong local geometric representation, making ALIKED suitable for real-time image matching and robotic perception tasks. More recently, XFeat [5] further advances this efficiency-oriented direction by adopting a hardware-friendly lightweight architecture together with efficient dense feature computation and coarse-to-fine keypoint localization, achieving strong matching performance with substantially reduced computational cost. Such properties make it especially attractive for deployment on embedded robotic platforms and resource-constrained platforms, where both robustness and real-time capability are critical.

2.1.2 Line Detection

Compared with point features, line segments provide richer structural and geometric information, and are particularly valuable in low-texture or man-made environments where point features are sparse or unstable [9]. Traditional line segment detectors, such as LSD [10] and EDLines [11], estimate line support regions from local gradient distributions and extract line segments through geometric verification. These methods are efficient and easy to deploy, but their performance is often sensitive to image noise, threshold selection, and illumination variations, which limits their robustness in challenging real-world scenarios.

Recent deep learning methods have substantially advanced line detection by learning higher-level structural priors directly from data. Early learning-based approaches mainly focused on line proposal generation or wireframe parsing from junctions and line hypotheses. Among these pioneering works, L-CNN [12] introduced a unified, end-to-end differentiable network for wireframe parsing. It established a classic two-stage pipeline: proposing junctions, sampling candidate lines between them, and verifying these proposals through a line classification module. Building upon this foundation, HAWP [4] is one of the most representative frameworks. It formulates wireframe parsing through a holistic attraction field representation and decomposes the problem into junction proposal generation, line proposal construction, and verification. This design elegantly eliminates the computationally expensive heuristic sampling seen in L-CNN,

significantly improving both structural accuracy and efficiency, and has become a strong baseline for line-aware geometric perception.

To bridge structural line detection with downstream applications like SLAM and image matching, recent works have shifted focus towards joint detection, descriptor extraction, and efficiency. ELSD [13] introduced a highly compact, one-stage architecture that parameterizes lines using their mid-points, angles, and lengths. By sharing feature maps between the detection and description branches, ELSD seamlessly couples geometric localization with appearance representation in real time. Pushing efficiency even further for resource-constrained environments, EM-LSD [14] provides a lightweight architecture explicitly designed for multi-scale line segment extraction. By incorporating modules like Dense Atrous Convolution (DAC) and spatial-channel attention, EM-LSD effectively captures global structural features in geometrically complex and noisy scenes, avoiding the heavy computational overhead typical of multi-stage networks.

Parallel to these supervised frameworks, SOLD^2 [15] represents a major shift toward self-supervised learning and multi-view matching. Unlike methods relying on heavily annotated wireframe datasets, SOLD^2 jointly learns line segment detection and description without human line annotations. Its robust, occlusion-aware line descriptors enable accurate line matching under severe viewpoint changes, bridging the gap between structural line detection and traditional learned feature points for downstream 3D geometric tasks.

2.2 Unified Network for Keypoint and Line Detection

Unified point-line networks aim to share low-level visual computation and simultaneously predict keypoints, junctions, and line segments in one forward pass. Representative systems, such as PLNet: the unified front-end used in AirSLAM [2], combined a shared point-oriented backbone: SuperPoint, with HAWP-style line parsing backend. Compared with running independent point and line detectors, such designs reduce redundant computation and are therefore appealing for real-time SLAM systems.

However, existing unified frameworks still have two important limitations. First, many of them are built on early-generation VGG-like backbones, which introduce non-negligible memory and latency overhead on embedded platforms. Second, they often rely on direct feature sharing between point and line branches, without explicitly considering the representational difference between discrete keypoint localization and continuous line parsing. In particular, keypoints are sparse and localized primitives, whereas line extraction requires preserving elongated geometric continuity. As a result, simply replacing the point backbone with a stronger modern local feature extractor does not necessarily lead to better line parsing performance, and may even disturb the geometric representation required by the line decoder.

While advancements in modern local feature extractors have been rapid, the specific challenge of integrating these highly-optimized point features with continuous line parsing has received surprisingly little attention. Our work is based on PLNet's line of research, but instead of naively coupling point features with the entire line parsing pipeline, we exploit the observation that point and line cues are most naturally aligned

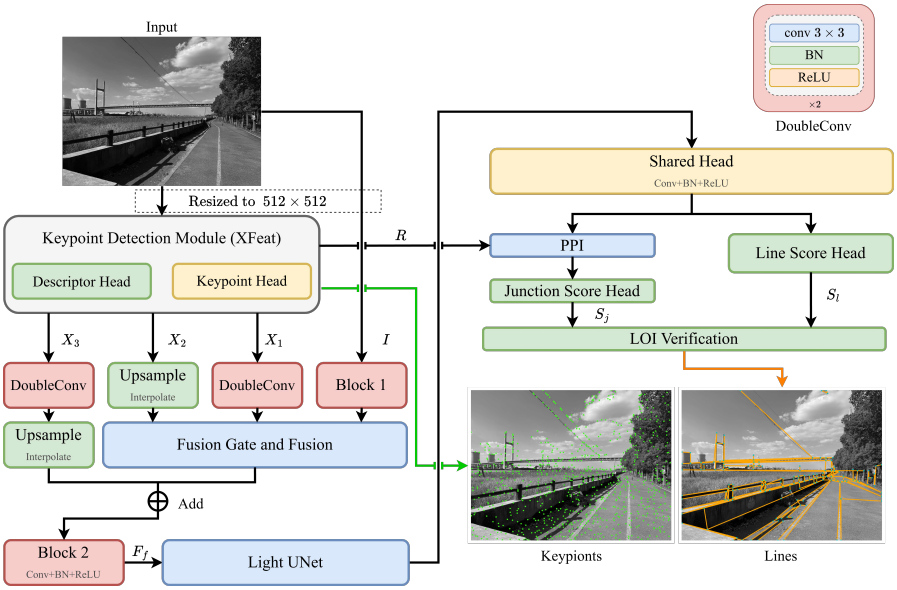

Fig. 2 Architecture of the proposed XPLNet

at junctions and endpoints. Accordingly, XFeat-derived point priors are injected only into the junction proposal branch, where they can enhance endpoint localization while avoiding unnecessary interference with continuous line representation.

3 Method

3.1 Overview

In this section we present details of the proposed XPLNet, a unified point-line detection framework that integrates a XFeat-based point extraction module with a HAWP-style line parsing backend in a shared architecture.

It is worth noting that the proposed design differs from conventional multi-branch fusion in two aspects. First, the interaction between point and line features is asymmetric. The point branch provides priors for line parsing, but the line branch is not forced to fully inherit point-oriented representations. This avoids making the dense line decoder overly dependent on sparse local keypoint responses. Second, the interaction is position-aware within the parsing pipeline. Point priors are mainly introduced at the junction proposal stage, where keypoint responses are geometrically consistent with endpoints and corners, rather than being injected into the entire line score prediction branch. In this sense, XPLNet explicitly separates point-guided junction reasoning from continuous line geometry decoding.

A straightforward solution is to follow PLNet and directly replace the SuperPoint-like backbone with XFeat. However, we observe that such a naive replacement is not effective. The reason is that XFeat is primarily designed for efficient local feature extraction and matching, and its backbone is much more lightweight than the

SuperPoint-style backbone used in PLNet. Although this design brings clear advantages in runtime efficiency, its intermediate features have limited capacity for dense wireframe parsing, especially for maintaining continuous line geometry and long-range structural consistency. Therefore, directly using XFeat features as the main representation for the entire HAWP-style line parsing pipeline leads to unsatisfactory performance.

Based on this observation, XPLNet does not treat XFeat as a full replacement for the line parsing backbone. Instead, we introduce XFeat as a point-prior provider and design a selective point-line interaction mechanism. The extracted XFeat features are used to enhance shallow structural representation through gated fusion, while the XFeat reliability response is further injected into the junction proposal branch, where point cues are most relevant to endpoint and corner localization. Meanwhile, the continuous line representation is decoded by a lightweight topology-aware UNet, which preserves the geometric continuity required for line segment extraction. This design allows XPLNet to benefit from the efficiency of XFeat without suffering from the accuracy degradation caused by direct backbone replacement.

In particular, the core design principle of XPLNet is to respect the complementary yet distinct nature of point and line representations. Point features are sparse, locally discriminative, and highly correlated with corners, endpoints, and junctions, whereas line features require dense spatial continuity, directional consistency, and long-range geometric reasoning. Therefore, instead of directly injecting point features into the entire line parsing pipeline, which may disturb the continuous structure modeling required by line extraction, we introduce point priors only at stages where the two modalities are naturally aligned: the shared structural feature enhancement stage and the junction proposal stage.

As shown in Fig. 2, given an input grayscale image resized to 512×512, denoted as $I \in \mathbb{R}^{B\times 1\times H\times W}$, XPLNet first extracts point-domain structural priors using the XFeat module.The extracted XFeat features are then projected and fused with shallow image features through a gated point-line fusion module. The fused representation is enhanced by an XFeat reliability prompt and then decoded by a lightweight topology-aware line UNet. Finally, the network predicts HAWP-style line geometric components and junction components through two decoupled prediction heads. The final prediction is obtained by concatenating the line score map S_l and junction score map S_j:

$$S = \mathrm{Concat}(S_l, S_j) \in \mathbb{R}^{B\times 9\times H/4\times W/4}. \tag{1}$$

where

$$S_l \in \mathbb{R}^{B\times 5\times H/4\times W/4}, \quad S_j \in \mathbb{R}^{B\times 4\times H/4\times W/4}. \tag{2}$$

The 9 output channels follow the HAWP-style line parsing backend [4], including line direction, line distance, residual prediction, junction localization, and junction offset. These outputs are subsequently fed into the LOI verification module for line-junction matching and decoding, producing the final line detection results.

In this way, XPLNet performs point-line extraction in a single forward pass and maximizes the utilization of point information produced by the keypoint detector, achieving high efficiency while preserving competitive structural detection capability.

3.2 XFeat-based point prior extraction

We adopt XFeat as the point prior generator because of its favorable efficiency-accuracy trade-off for real-time applications.

Specifically, for an input image I, the point detector produces: a keypoint heatmap K, a reliability map R, and intermediate multi-scale features X_1, X_2, X_3, where

$$K \in \mathbb{R}^{B\times 1\times H\times W}, \tag{3}$$

$$R \in \mathbb{R}^{B\times 1\times H/8\times W/8}, \tag{4}$$

$$X_1 \in \mathbb{R}^{B\times 24\times H/4\times W/4}, \tag{5}$$

$$X_2, X_3 \in \mathbb{R}^{B\times 64\times H/8\times W/8} \tag{6}$$

Within our framework, K is leveraged as the source for keypoint detection. In contrast, the reliability map R, which captures dense point-supporting structures, is specifically used to guide junction localization. This allows the junction branch to exploit stable geometric priors from the point detector without being directly coupled with the sparse keypoint heatmap K. Meanwhile, the intermediate features X_1, X_2, X_3 serve as shallow multi-scale features that can be reused in the line detection early stages, therefore reducing overall network complexity while preserving useful low-level and mid-level information.

3.3 Multi-source shared feature construction

To construct a shared representation suitable for both point-aware and line-aware reasoning, we aggregate information from multiple feature sources, including shallow image-domain features and intermediate XFeat features at different scales.

The resized grayscale input image I is first processed by a shallow convolutional stem denoted as Block 1 in Fig. 2 to obtain image-domain low-level features: $P_0 = \phi_0(I)$, where $\phi_0(\cdot)$ denotes a lightweight convolutional encoder that downsamples the image to $1/4$ resolution.

Meanwhile, two intermediate feature maps from the XFeat branch are adapted to the shared feature space: $P_1 = \phi_1(X_1)$, $P_2 = \phi_2(X_2)$, where X_1 and X_2 denote XFeat intermediate features at different scales. In the code, X_1 corresponds to a $1/4$ resolution feature map and X_2 to an $1/8$ resolution map. The latter is bilinearly upsampled to match the spatial resolution of P_0.

Since these features are generated from the early convolutional layers of XFeat module, they mainly encode shallow structural and appearance cues rather than high-level semantic information. Therefore, they can be effectively reused for line feature extraction without introducing an additional heavy backbone, thereby reducing the overall computational complexity of the network. In addition, the fused intermediate representation from the XFeat branch, denoted as X_3, is also resized and projected to the same channel dimension: $P_3 = \phi_d(X_3)$.

Together with the shallow image-domain feature P_0, the adapted XFeat features P_1, P_2, and P_3 form a unified set of feature candidates that contain complementary information from raw image appearance, early structural cues, and point-domain representations.

However, directly concatenating these heterogeneous features may lead to unbalanced feature mixing, where one feature source dominates the fused representation and weakens the contribution of the others. To alleviate this problem, XPLNet introduces a FusionGate module to adaptively recalibrate each branch before fusion. Given the shallow image feature P_0, the shallow XFeat feature P_1, and P_2, the module first computes branch-specific gating weights:

$$\begin{aligned} G_0 &= \sigma(\mathrm{Conv}_{1\times1}(P_0)), \\ G_1 &= \sigma(\mathrm{Conv}_{1\times1}(P_1)), \\ G_2 &= \sigma(\mathrm{Conv}_{1\times1}(P_2)), \end{aligned} \tag{7}$$

where $\sigma(\cdot)$ denotes the sigmoid activation function. Each feature branch is then recalibrated by its corresponding gate:

$$\tilde{P}_0 = P_0 \odot G_0, \tilde{P}_1 = P_1 \odot G_1, \tilde{P}_2 = P_2 \odot G_2, \tag{8}$$

where $\odot$ denotes element-wise multiplication. The recalibrated features are concatenated along the channel dimension and further fused by a DoubleConv module:

$$F_g = \phi_g\left(\mathrm{Concat}\left(\tilde{P}_0, \tilde{P}_1, \tilde{P}_2\right)\right), \tag{9}$$

where $F_g \in \mathbb{R}^{B\times128\times\frac{H}{4}\times\frac{W}{4}}$ represents the gated fusion feature.

To further enrich the fused representation with deeper structural cues from XFeat, the feature P_3 is upsampled to the same spatial resolution and projected to the same number of channels. Unlike P_1 and P_2, P_3 is used as a deeper residual structural cue and is added after gated fusion. This design allows shallow heterogeneous features to be recalibrated before fusion, while preserving the complementary high-level response from the XFeat branch. The transformed feature $\hat{P}_3$ is then added to F_g, yielding the final fused feature F_f:

$$F_f = \mathrm{ReLU}(\mathrm{BN}(\mathrm{Conv}_{1\times1}(F_g + \hat{P}_3))) \tag{10}$$

3.4 Light UNet

3.4.1 Overview

Encoder–decoder networks with skip connections have been widely adopted in dense prediction tasks because they can combine global contextual cues with fine-grained spatial details. UNet [16] is a representative architecture of this paradigm, using a contracting path for context aggregation and an expanding path for spatial detail recovery. Inspired by this design, we propose a compact line decoder, termed Light UNet, to recover continuous line structures from the fused point-line representation, as shown in Fig. 3.

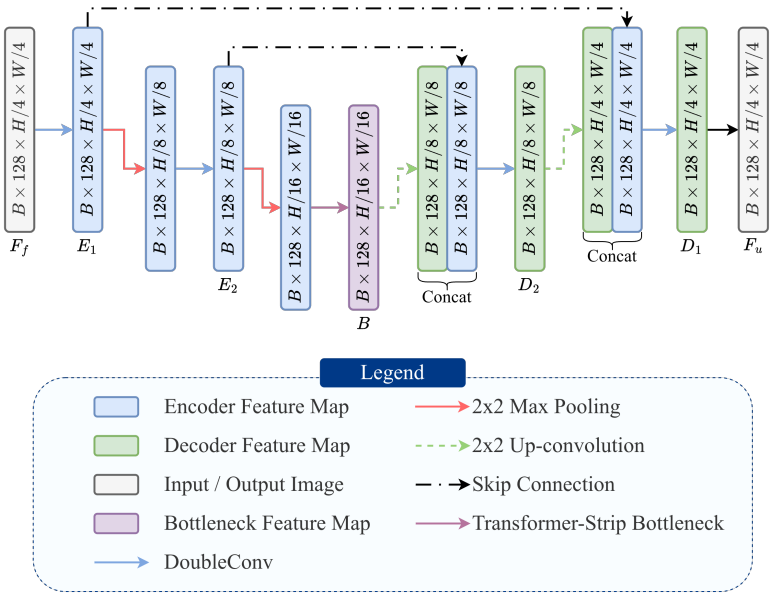

Fig. 3 Architecture of the proposed Light UNet

Compared with the heavier decoders commonly used in previous point-line parsing frameworks, Light UNet contains only two downsampling stages and two upsampling stages. The skip connections preserve local boundary and endpoint details, while the bottleneck feature provides broader structural context for line reasoning. As a result, Light UNet maintains the essential multi-scale modeling ability of U-Net while reducing computational cost, making it more suitable for real-time perception and visual SLAM front-ends.

Given the input feature

$$F_f \in \mathbb{R}^{B\times 128\times H/4\times W/4}, \tag{11}$$

the encoder first extracts a first-level feature:

$$\begin{aligned} E_1 &= \mathrm{DoubleConv}(F_f), \\ E_1 &\in \mathbb{R}^{B\times 128\times H/4\times W/4}. \end{aligned} \tag{12}$$

After max pooling, the second encoder block produces

$$\begin{aligned} E_2 &= \mathrm{DoubleConv}\left(\mathrm{MaxPool}(E_1)\right), \\ E_2 &\in \mathbb{R}^{B\times 128\times H/8\times W/8}. \end{aligned} \tag{13}$$

The bottleneck operates on a further downsampled feature:

$$B = \mathcal{T}\left(\mathrm{MaxPool}(E_2)\right), \tag{14}$$

where $\mathcal{T}$ denotes the proposed Transformer-Strip bottleneck, and

$$B \in \mathbb{R}^{B\times 128\times H/16\times W/16}. \tag{15}$$

The decoder then upsamples the bottleneck feature and uses skip connections to recover spatial details:

$$D_2 = \text{DoubleConv}\left(\text{Concat}\left(\text{Upsample}(B), E_2\right)\right), \tag{16}$$

$$D_1 = \text{DoubleConv}\left(\text{Concat}\left(\text{Upsample}(D_2), E_1\right)\right). \tag{17}$$

The final decoder output is $F_u = D_1$,where

$$F_u \in \mathbb{R}^{B\times 128\times H/4\times W/4}. \tag{18}$$

The compact two-level encoder-decoder design balances accuracy and efficiency. It preserves sufficient multi-scale context for line parsing while avoiding the high computational cost of deeper UNet variants.

3.4.2 Transformer-Strip Bottleneck

The proposed Transformer-Strip Bottleneck is motivated by two complementary observations. On the one hand, strip pooling and asymmetric strip convolutions have shown strong capability in capturing elongated and direction-sensitive structures by aggregating long-range contextual information along specific spatial directions. On the other hand, bottleneck Transformer designs demonstrate that self-attention can be introduced into compact feature stages to enhance global context modeling with limited computational overhead. These two ideas are consistent with the requirements of line-structure parsing, where both directional continuity and global endpoint-line relationships are important.

Although strip pooling effectively enhances horizontal and vertical long-range responses, robotic visual imagery often contains oblique edges, irregular man-made structures, and complex wireframe layouts. Relying only on strip-wise aggregation may therefore be insufficient for modeling non-axis-aligned geometric dependencies. To address this limitation, XPLNet further introduces a lightweight Transformer block at the bottleneck level, where the spatial resolution is relatively low, to capture global endpoint-line interactions efficiently.

Based on these insights, the proposed module combines strip-wise directional context aggregation with lightweight self-attention in a sequential manner. Specifically, it consists of two components: a Strip Pooling [17] unit for enhancing long-range directional line cues, followed by a lightweight Transformer block for modeling global structural dependencies. This design enables XPLNet to improve line-structure representation while maintaining the efficiency required for real-time SLAM applications.

$$\mathcal{T}(X) = \text{Transformer}\left(\text{StripPooling}(X)\right). \tag{19}$$

Strip Pooling for Long-range Line Continuity: Given a bottleneck feature

$$X \in \mathbb{R}^{B\times C\times H'\times W'}, \tag{20}$$

Strip Pooling first aggregates long-range context along two orthogonal directions:

$$\begin{aligned} X_h &= \mathrm{Conv}_{3\times 1}\left(\mathrm{AvgPool}_{H'\times 1}(X)\right), \\ X_w &= \mathrm{Conv}_{1\times 3}\left(\mathrm{AvgPool}_{1\times W'}(X)\right). \end{aligned} \tag{21}$$

The two directional responses are expanded to the original spatial size and summed:

$$X_c = \mathrm{Expand}(X_h) + \mathrm{Expand}(X_w). \tag{22}$$

A 1×1 convolution followed by Batch Normalization and Sigmoid activation generates a spatial attention mask:

$$A = \sigma\left(\mathrm{BN}\left(\mathrm{Conv}_{1\times 1}(X_c)\right)\right). \tag{23}$$

The output is obtained by reweighting the input feature:$X_s = X \odot A$.

This operation provides long-range horizontal and vertical structural context with negligible computational overhead. For line parsing, it helps enhance continuous line responses while suppressing isolated background noise.

Lightweight transformer for global structural modeling: Given the strip-enhanced feature

$$X_s \in \mathbb{R}^{B\times C\times H'\times W'}, \tag{24}$$

it is flattened into a token sequence:

$$T = \mathrm{Flatten}(X_s) \in \mathbb{R}^{B\times H'W'\times C}. \tag{25}$$

Layer Normalization and multi-head self-attention are applied as:

$$T' = T + \mathrm{MHA}\left(\mathrm{LN}(T)\right). \tag{26}$$

Then a feed-forward MLP with residual connection further updates the token representation:

$$T'' = T' + \mathrm{MLP}\left(\mathrm{LN}(T')\right). \tag{27}$$

Finally, the token sequence is reshaped back into a 2D feature map:

$$X_t = \mathrm{Reshape}(T''). \tag{28}$$

Since the Transformer is applied only at 1/16 resolution, its computational overhead is controlled. Meanwhile, it enables global interaction among distant line-supporting regions, endpoints, and junctions. This improves the ability to model long, oblique, and topologically complex line structures.

3.5 Point prompted interaction

We argue that point priors should not be indiscriminately injected into all stages of line parsing. Although point responses provide useful cues for endpoint localization and junction reasoning, excessive point-driven modulation may disrupt the continuous geometric representation required for reliable line extraction.

Therefore, XPLNet introduces a Point Prompted Interaction (PPI) module and applies it selectively. In the junction prediction stage, the XFeat reliability map R is used as a prompt to guide junction localization.

Given a junction-domain feature map

$$F \in \mathbb{R}^{B\times C\times H'\times W'} \tag{29}$$

and a point prompt map

$$R \in \mathbb{R}^{B\times 1\times H_r\times W_r}, \tag{30}$$

the prompt map is first resized to the feature resolution:

$$\hat{R} = \mathrm{Resize}(R, H', W'). \tag{31}$$

Then $\hat{R}$ is projected into a 2C-channel representation:

$$[\Delta, Q] = \phi_p(\hat{R}) \tag{32}$$

where Δ is an additive prompt residual and Q is used to generate a multiplicative gate:

$$G = \sigma(Q). \tag{33}$$

The prompted feature is computed as

$$F' = F + \alpha\Delta + \beta F \odot G, \tag{34}$$

where α and β are learnable scalar parameters initialized to 0.1.

This residual-gated formulation allows the network to introduce point-prior information softly. The additive term provides prompt-based feature compensation, while the multiplicative gate adaptively enhances point-consistent regions. Since α and β are learnable, the strength of point guidance can be automatically adjusted during training.

After gated point-line fusion, XPLNet uses the XFeat reliability map R to enhance the junction feature J:

$$J_e = \mathrm{PPI}(J, R). \tag{35}$$

4 Experiments

4.1 Datasets and Implementation Details

The proposed XPLNet is trained on the Wireframe dataset and evaluated on both the Wireframe and YorkUrban test datasets. The Wireframe dataset [18] contains

5,000 training images and 462 test images, with rich annotations of line segments and junctions collected from diverse man-made environments. The YorkUrban dataset [19] consists of 102 urban-scene test images and is commonly used to evaluate the cross-domain generalization ability of line segment detectors.

XPLNet is implemented in PyTorch and trained on an NVIDIA RTX 5090 GPU. The training process is divided into two stages. In the first stage, only the keypoint detection module is trained. In the second stage, the keypoint detection module is frozen, and the remaining line parsing network is optimized. This two-stage strategy stabilizes the training process and prevents the point detector from being degraded by the line parsing objective.

For network input, all images are resized to 512×512 and converted to grayscale. During dataset preparation, we apply random horizontal flipping, random scaling with a scale factor sampled from 0.6 to 1.3, random cropping, and color jittering for data augmentation. Each training image is augmented six times to improve the robustness of the model.

The network is optimized using a two-phase solver strategy. Specifically, AdamW is adopted for the first 25 epochs with a batch size of 8 and an initial learning rate of 4×10^{-4}. In the final 5 epochs, we switch to SGD for fine-tuning, with the learning rate set to 4×10^{-3}.

4.2 Evaluation Metrics

Structural average precision (sAP). Structural Average Precision (sAP) is a commonly used metric for evaluating wireframe parsing and line segment detection. It follows the general idea of Average Precision (AP) in object detection, but replaces the conventional bounding-box Intersection-over-Union (IoU) criterion with a structural matching criterion defined on line segments. Specifically, for each ground-truth line segment $l = (\mathbf{x}_1, \mathbf{x}_2)$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ denote its two endpoints, we search for a predicted line segment $\hat{l} = (\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2)$ that satisfies the endpoint-based structural overlap condition:

$$\min_{(i,j)\in\{(1,2),(2,1)\}} \left(\|\mathbf{x}_1 - \hat{\mathbf{x}}_i\|_2^2 + \|\mathbf{x}_2 - \hat{\mathbf{x}}_j\|_2^2\right) \leq \vartheta_L, \tag{36}$$

where $(i, j) \in \{(1, 2), (2, 1)\}$ indicates the two endpoint orders are both considered, and ϑ_L denotes the structural matching threshold. For each ground-truth segment, the matched prediction with the highest confidence score is assigned as a true positive (TP), while other predictions matched to the same segment are counted as false positives (FP). Ground-truth segments without any matched prediction are false negatives (FN), and predictions that do not match any ground-truth segment are also counted as FPs. The final sAP is obtained by sorting all predictions according to their confidence scores and computing the area under the resulting precision–recall curve. We report sAP^5, sAP^{10}, and sAP^{15}, where the superscript indicates the value of ϑ_L. Smaller thresholds require more accurate endpoint localization.

Frames per second (FPS). Frames Per Second (FPS) is a commonly used metric for evaluating the runtime efficiency and real-time capability of visual perception systems. It measures the number of images that can be processed by the model per

second during inference. Specifically, let t_i denote the inference time required for processing the i-th image, including feature extraction, line parsing, and post-processing stages. Given a test set containing T images, the average FPS is computed as:

$$\text{FPS} = \frac{T}{\sum_{i=1}^{T} t_i}. \tag{37}$$

Higher FPS values indicate better runtime efficiency and stronger suitability for real-time applications. During evaluation, the reported runtime includes the complete forward inference pipeline of the proposed method.

4.3 Experimental Results and Comparisons

4.3.1 Baselines

We compare the proposed XPLNet with several representative methods for line segment detection and wireframe parsing, including L-CNN [12], ELSD-Lite [13], HAWPv2 [4], and EM-LSD [14]. These baselines cover both learning-based wireframe parsing frameworks and efficient line segment detectors. L-CNN and HAWPv2 represent deep wireframe parsing methods based on junction detection and line verification, whereas ELSD-Lite and EM-LSD are designed for fast line segment detection.

We further compare XPLNet with PLNet, which is the most closely related baseline to our work. PLNet follows a unified point-line detection paradigm and combines point feature extraction with HAWP-style line parsing. This makes it a strong reference for evaluating whether XPLNet can achieve efficient unified point-line extraction while maintaining competitive structural accuracy.

4.3.2 The Results

Table 1 shows the quantitative comparison between XPLNet and existing methods on the Wireframe and YorkUrban datasets. On the Wireframe dataset, XPLNet achieves 62.3, 66.3, and 68.1 in sAP^5, sAP^{10}, and sAP^{15}, respectively. Although its structural accuracy is slightly lower than that of PLNet and HAWPv2, XPLNet outperformed all fast line segment detection method and obtains the highest inference speed among all compared methods, reaching 155.36 FPS. Compared with PLNet, XPLNet improves the inference speed from 106.4 FPS to 155.36 FPS, corresponding to an approximate 46.0% speed improvement. This demonstrates that the proposed lightweight architecture effectively reduces computational redundancy while maintaining competitive line detection accuracy.

On the YorkUrban dataset, XPLNet achieves 27.3, 30.0, and 31.5 in sAP^5, sAP^{10}, and sAP^{15}, respectively, with an inference speed of 103.49 FPS. Compared with PLNet, XPLNet shows lower sAP values but significantly higher runtime efficiency, improving the FPS from 81.41 to 103.49. Compared with HAWPv2, XPLNet obtains similar structural accuracy, especially in sAP^5 and sAP^{15}, while also achieving faster inference. These results suggest that XPLNet maintains reasonable cross-domain generalization ability while providing a clear advantage in computational efficiency.

Overall, the proposed XPLNet achieves a favorable balance between structural detection accuracy and inference speed. Although PLNet achieves the highest sAP scores on both datasets, it introduces higher computational cost. In contrast, XPLNet substantially improves runtime efficiency by using a lightweight unified point-line architecture and reducing redundant decoding operations. Therefore, XPLNet is more suitable for real-time point-line feature extraction scenarios where computational efficiency is critical. Although the present evaluation focuses on image-level detection benchmarks, the improved efficiency suggests that XPLNet could serve as a promising front-end candidate for lightweight visual SLAM systems.

Table 1 Quantitative results and comparison with existing methods, results marked with † indicate the number is extracted from the original paper

Methods	Wireframe Dataset [18]				YorkUrban Dataset [19]			
	sAP^5	sAP^{10}	sAP^{15}	FPS	sAP^5	sAP^{10}	sAP^{15}	FPS
L-CNN [12]	59.7	63.6	65.3	47.62	25.0	27.1	28.3	34.57
ELSD-Lite [13]	57.4†	63.1†	65.5†	107.5†	24.3†	27.4†	29.3†	107.5†
HAWPv2 [4]	64.1	68.1	69.8	127.46	27.4	29.8	31.1	92.28
EM-LSD [14]	58.1†	63.2†	-	130.1†	25.1†	28.3†	-	130.1†
PLNet [2]	65.2	69.2	70.9	106.4	29.3	32.0	33.5	81.41
XPLNet(ours)	62.3	66.3	68.1	**155.36**	27.3	30.0	31.5	**103.49**

4.4 Visualization analysis

To further illustrate the practical effectiveness of the proposed method, we provide qualitative visualization results on the YorkUrban dataset and real-world indoor/outdoor scenes, as shown in Fig. 4. The first row shows the original images, while the following rows present the results produced by HAWP, PLNet, and the proposed XPLNet, respectively. Orange lines denote detected line segments, and green points denote extracted keypoints. Since HAWP is a line-only parsing method, no keypoint results are visualized for HAWP.

From the YorkUrban example, HAWP can detect the dominant building structures, but it mainly focuses on line segments and does not provide point features for subsequent point-line SLAM. PLNet produces both line segments and keypoints; however, some line responses are fragmented. In contrast, XPLNet detects more complete structural lines along building boundaries, windows, roof edges, and facade contours. This indicates that the proposed XFeat-guided point prior can provide useful endpoint and junction cues for line structure parsing.

For the outdoor scene, the image contains weak-texture regions, long ground boundaries, vertical poles, building edges, and shadowed areas. HAWP captures several major line structures, but some long boundaries are incomplete or partially missing. PLNet generates point-line results, but its line detections contain more fragmented responses in the ground and background regions. XPLNet preserves long structural



Fig. 4 Visualization comparison of line segment and keypoint detection results produced by different methods on the YorkUrban dataset and real-world indoor/outdoor scenes. From top to bottom are the original images, the results of HAWP, PLNet, and the proposed XPLNet, respectively. Orange lines denote detected line segments, while green points indicate extracted keypoints

lines more effectively, such as pavement boundaries, building edges, poles, and object contours. Meanwhile, the extracted keypoints are distributed over both textured and structural regions, which can provide more reliable geometric constraints for visual SLAM.

In the indoor scene, the environment contains dense man-made structures, including cabinets, walls, doors, tiles, countertops, and object boundaries. HAWP detects many strong indoor line structures but also produces some redundant line responses around cluttered regions. PLNet provides additional keypoints, but some structural boundaries are still incomplete. Compared with these methods, XPLNet produces more coherent line segments along dominant indoor layouts, such as cabinet edges, wall boundaries, door frames, sink contours, and countertop structures. The simultaneously

extracted keypoints are also more abundant, especially around corners, junctions, and high-response structural details.

Overall, the visualization results show that XPLNet can produce more complete point-line representations in various scenes. Compared with HAWP, XPLNet provides additional keypoint outputs required by point-line SLAM. Compared with PLNet, the proposed method shows more stable keypoint extraction and better preservation of major structural lines in several challenging regions. These qualitative results further support the effectiveness of introducing XFeat-based point priors into line structure parsing.

4.5 Ablation studies

To better understand the contribution of several key design choices in XPLNet, we conduct ablation studies on the choice of keypoint detector, the proposed PPI module, and the lightweight UNet decoder. All experiments are evaluated on the Wireframe and YorkUrban datasets. We report structural average precision at threshold 10, denoted as sAP^{10}, model parameters, and inference speed on the Wireframe dataset.

4.5.1 Effect of different keypoint detector modules

We first investigate the influence of different keypoint detector modules. As shown in Table 2, replacing XFeat with ALIKED [8] improves the detection accuracy from 66.3 to 69.5 on Wireframe and from 30.0 to 32.8 on YorkUrban. This indicates that a stronger keypoint detector can provide more discriminative point priors for line structure parsing, especially for junction and endpoint localization.

However, ALIKED also introduces a considerably larger parameter cost. The model with ALIKED contains 11.29M parameters, while the XFeat-based model contains only 6.71M parameters. Moreover, the inference speed decreases from 155.36 FPS to 103.44 FPS when ALIKED is used. Although ALIKED brings higher accuracy, the computational burden is less favorable for real-time perception. Therefore, XFeat is selected as the default point-prior extractor in XPLNet because it provides a better balance between efficiency and accuracy.

This result also supports the motivation of our design: the proposed framework is not limited to a specific keypoint detector. Stronger point priors can improve line parsing accuracy, but lightweight point extractors are more suitable for visual front-ends on resource-constrained platforms.

Table 2 Ablation study of different keypoint detector module

KeyPoint Detector Module	Params	Wireframe sAP^{10}	YorkUrban sAP^{10}	Wireframe FPS
ALIKED [8]	11.29M	69.5	32.8	103.44
XFeat [5]	6.71M	66.3	30.0	155.36

4.5.2 Effect of PPI module

We then evaluate the effectiveness of the proposed PPI module. As shown in Table 3, removing PPI results in 64.2 sAP^{10} on Wireframe and 28.1 sAP^{10} on YorkUrban. After introducing PPI, the performance increases to 66.3 on Wireframe and 30.0 on YorkUrban.

This demonstrates that selectively injecting XFeat point priors is beneficial for line structure parsing. In particular, the reliability map guides the junction prediction branch. Compared with simple feature concatenation, the residual-gated prompt formulation allows point information to modulate line-domain features in a controlled manner, thereby improving endpoint and junction reasoning without severely disturbing line continuity.

Meanwhile, the inference speed slightly decreases from 158.23 FPS to 155.36 FPS after adding PPI. This small overhead is acceptable considering the consistent accuracy improvement on both datasets. Therefore, PPI provides an effective accuracy-efficiency trade-off and validates the necessity of selective point-prior guidance.

Table 3 Ablation study of proposed PPI module

PPI	Wireframe sAP^{10}	YorkUrban sAP^{10}	Wireframe FPS
	64.2	28.1	158.23
✓	66.3	30.0	155.36

4.5.3 Effect of the lightweight UNet decoder

Finally, we study the influence of different UNet decoder variants by replacing the UNet decoder in XPLNet. As reported in Table 4, the original UNet used in PLNet achieves 68.0 sAP^{10} on Wireframe and 30.9 sAP^{10} on YorkUrban, with 9.47M parameters and 125.97 FPS. This variant obtains the best accuracy among the compared decoders, but it also introduces the largest computational cost.

We first evaluate the lightweight decoder adopted in the final XPLNet, which uses a single LightUNet stage. This design reduces the parameter count from 9.47M to 6.71M and improves the inference speed from 125.97 FPS to 155.36 FPS. Its accuracy reaches 66.3 sAP^{10} on Wireframe and 30.0 sAP^{10} on YorkUrban. Although the accuracy is slightly lower than that of the original PLNet UNet, the single-stage LightUNet achieves a substantial improvement in efficiency. This indicates that a compact decoder can retain the main line structure modeling capability while being more suitable for real-time perception.

We further evaluate an experimental variant, XPLNet-v2, which stacks two LightUNet stages for cascaded line feature refinement. Compared with the final single-stage XPLNet, XPLNet-v2 improves the Wireframe sAP^{10} from 66.3 to 67.6. However, its sAP on YorkUrban remains unchanged, while the number of parameters increases from

6.71M to 7.86M and the inference speed decreases from 155.36 FPS to 137.40 FPS. This suggests that the second LightUNet stage can slightly enhance the refinement capability for line geometry and junction localization, but at the cost of additional computation.

Overall, the ablation study reveals a clear trade-off between accuracy and efficiency. The original PLNet UNet achieves the highest accuracy, but it introduces considerable computational cost. XPLNet-v2 improves the accuracy over the final XPLNet by stacking two LightUNet stages, but this improvement comes at the expense of inference speed. Therefore, considering the real-time requirements of lightweight visual perception and SLAM front-ends on resource-constrained platforms, we adopt the single-stage LightUNet as the default decoder in XPLNet.

Table 4 Ablation study of proposed light UNet

UNet variant	Stages	Total Params	Wireframe sAP^{10}	YorkUrban sAP^{10}	Wireframe FPS
UNet(PLNet)	2	9.47M	68.0	30.9	125.97
LightUNet (XPLNet-v2)	2	7.86M	67.6	30.0	137.40
LightUNet (XPLNet)	1	6.71M	66.3	30.0	155.36

5 Conclusion

In this paper, we proposed XPLNet, an accelerated unified point-line detection network for real-time geometric feature extraction, with potential applications in lightweight visual perception and SLAM front-ends on resource-constrained platforms. We introduced a selective point-prior fusion strategy that injects XFeat-derived reliability responses only into the junction proposal branch through the proposed Point Prompted Interaction module. This design effectively exploits point priors for junction-aware line parsing while avoiding excessive interference with continuous line geometry.

To improve efficiency, XPLNet reuses intermediate representations from the lightweight XFeat module and constructs shared point-line features through a gated multi-source fusion mechanism. In addition, we redesigned the heavy decoder used in previous point-line parsing frameworks into a compact Light UNet. The proposed decoder preserves the essential multi-scale modeling capability required for line structure recovery while significantly reducing computational overhead. A Transformer-Strip bottleneck is further introduced to enhance long-range directional context and global structural dependency modeling at low spatial resolution.

Extensive experiments on the Wireframe and YorkUrban datasets demonstrate that XPLNet achieves a favorable trade-off between accuracy and efficiency. It should be noted that the current evaluation focuses on image-level point-line detection benchmarks. The influence of XPLNet on full SLAM pipelines, such as trajectory accuracy,

tracking robustness, and mapping quality, remains to be further investigated. Compared with PLNet, XPLNet substantially improves inference speed while maintaining competitive structural detection performance. The ablation studies confirm that the proposed PPI module consistently improves line detection accuracy, and that the lightweight UNet decoder effectively reduces parameters and increases FPS. Qualitative results further show that XPLNet can produce coherent line segments and stable keypoint distributions in urban, outdoor, and indoor scenes.

Although XPLNet improves the runtime efficiency of unified point-line detection, several limitations remain. First, its structural accuracy is still slightly lower than that of heavier models such as PLNet on some benchmarks, indicating that further improvements in lightweight feature representation are needed. Second, the current framework mainly evaluates detection performance on image-level datasets, while its direct impact on downstream SLAM accuracy has not yet been fully investigated. In future work, we will integrate XPLNet into a complete point-line visual SLAM system and evaluate its influence on camera localization accuracy, trajectory robustness, and mapping quality under challenging UAV scenarios. We will also explore adaptive point-line uncertainty estimation and hardware-aware model compression to further improve deployment efficiency on embedded aerial platforms.

References

[1] Gomez-Ojeda, R., Moreno, F.A., Scaramuzza, D., Jiménez, J.G.: PL-SLAM: a stereo SLAM system through the combination of points and line segments. CoRR **abs/1705.09479** (2017) 1705.09479

[2] Xu, K., Hao, Y., Yuan, S., Wang, C., Xie, L.: AirSLAM: An efficient and illumination-robust point-line visual slam system. IEEE Transactions on Robotics (TRO) (2024)

[3] DeTone, D., Malisiewicz, T., Rabinovich, A.: Superpoint: Self-supervised interest point detection and description. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 337–33712 (2018). https://doi.org/10.1109/CVPRW.2018.00060

[4] Xue, N., Wu, T., Bai, S., Wang, F.-D., Xia, G.-S., Zhang, L., Torr, P.H.S.: Holistically-attracted wireframe parsing: From supervised to self-supervised learning **45**(12), 14727–14744 https://doi.org/10.1109/TPAMI.2023.3312749 . Accessed 2025-12-25

[5] Potje, G., Cadar, F., Araujo, A., Martins, R., Nascimento, E.R.: XFeat: Accelerated features for lightweight image matching. arXiv. https://doi.org/10.48550/arXiv.2404.19174 . http://arxiv.org/abs/2404.19174 Accessed 2026-01-05

[6] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2), 91–110 (2004) https://doi.org/10.1023/B:VISI.0000029664.99615.94

[7] Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: An efficient alternative to sift or surf. In: 2011 International Conference on Computer Vision, pp. 2564–2571 (2011). https://doi.org/10.1109/ICCV.2011.6126544

[8] Zhao, X., Wu, X., Chen, W., Chen, P.C.Y., Xu, Q., Li, Z.: ALIKED: A lighter keypoint and descriptor extraction network via deformable transformation. arXiv. https://doi.org/10.48550/arXiv.2304.03608 . http://arxiv.org/abs/2304.03608 Accessed 2026-01-05

[9] Lin, X., Zhou, Y., Liu, Y., Zhu, C.: A comprehensive review of image line segment detection and description: Taxonomies, comparisons, and challenges. IEEE Transactions on Pattern Analysis and Machine Intelligence **46**(12), 8074–8093 (2024) https://doi.org/10.1109/TPAMI.2024.3400881

[10] Gioi, R., Jakubowicz, J., Morel, J.-M., Randall, G.: Lsd: A fast line segment detector with a false detection control. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(4), 722–732 (2010) https://doi.org/10.1109/TPAMI.2008.300

[11] Akinlar, C., Topal, C.: Edlines: A real-time line segment detector with a false detection control. Pattern Recognition Letters **32**(13), 1633–1642 (2011) https://doi.org/10.1016/j.patrec.2011.06.001

[12] Zhou, Y., Qi, H., Ma, Y.: End-to-end wireframe parsing. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 962–971 (2019). https://doi.org/10.1109/ICCV.2019.00105

[13] Zhang, H., Luo, Y., Qin, F., He, Y., Liu, X.: Elsd: Efficient line segment detector and descriptor. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 2949–2958 (2021). https://doi.org/10.1109/ICCV48922.2021.00296

[14] Hu, S., Zhao, L., Wang, Q.: Em-lsd: A lightweight and efficient model for multi-scale line segment detection. Robotics and Autonomous Systems **195**, 105192 (2026) https://doi.org/10.1016/j.robot.2025.105192

[15] Pautrat, R., Lin, J.-T., Larsson, V., Oswald, M.R., Pollefeys, M.: Sold2: Self-supervised occlusion-aware line description and detection. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11363–11373 (2021). https://doi.org/10.1109/CVPR46437.2021.01121

[16] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. ArXiv **abs/1505.04597** (2015)

[17] Hou, Q., Zhang, L., Cheng, M.-M., Feng, J.: Strip pooling: Rethinking spatial pooling for scene parsing. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4002–4011 (2020). https://doi.org/10.1109/CVPR42600.2020.00406

[18] Huang, K., Wang, Y., Zhou, Z., Ding, T., Gao, S., Ma, Y.: Learning to parse wireframes in images of man-made environments. In: CVPR (2018)

[19] Denis, P., Elder, J.H., Estrada, F.J.: Efficient edge-based methods for estimating manhattan frames in urban imagery. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) Computer Vision – ECCV 2008, pp. 197–210. Springer, Berlin, Heidelberg (2008)