CADSpotting: Robust Panoptic Symbol Spotting on Large-Scale CAD Drawings

Fuyi Yang[1,2]
[1]ShanghaiTech University
[2]DGene Digital Technology
yangfy@shanghaitech.edu.cn

Mingqian Zhang[1,2]
[1]ShanghaiTech University
[2]DGene Digital Technology
zhangmq@shanghaitech.edu.cn

Yanshun Zhang[2]
DGene Digital Technology
zhangys@dgene.com

Jiazuo Mu[1,2]
[1]ShanghaiTech University
[2]DGene Digital Technology
mujz@chinatelecom.cn

Junxiong Zhang[1,2]
[1]ShanghaiTech University
[2]DGene Digital Technology
zhangjx@shanghaitech.edu.cn

Lan Xu[1]
ShanghaiTech University
xulan@shanghaitech.edu.cn

Yingliang Zhang[2†]
DGene Digital Technology
yingliang.zhang@outlook.com

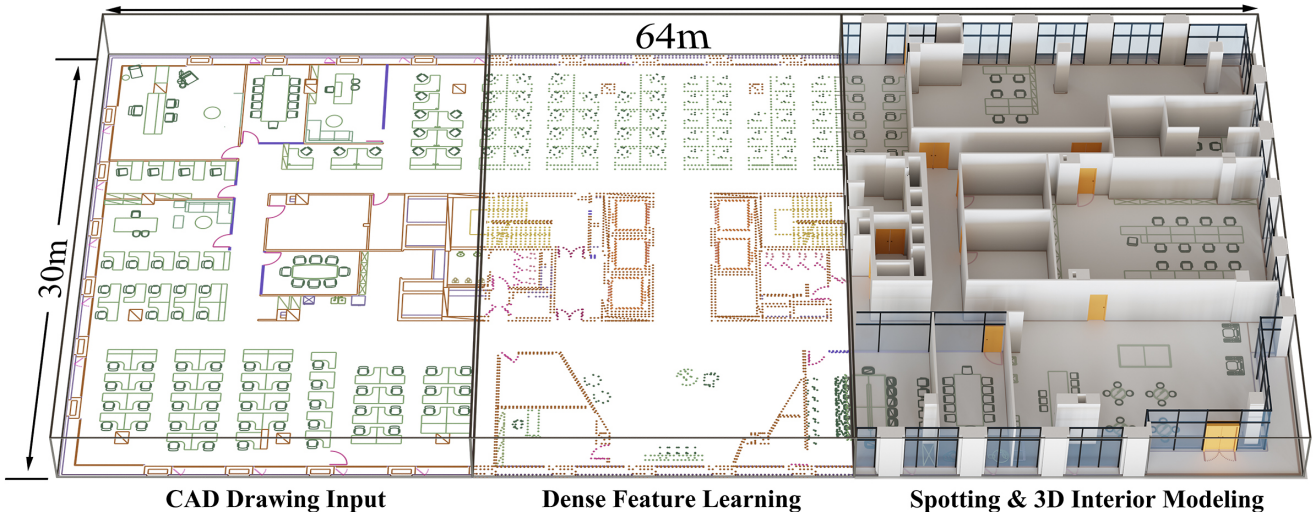


Figure 1: CADSpotting accurately identifies and segments symbols in CAD drawings, facilitating tasks like 3D interior modeling. It uses dense point sampling within a unified point cloud model to learn robust primitive features, and integrates Sliding Window Aggregation for efficient panoptic symbol spotting in large-scale drawings. The resulting semantic information enables automated parametric reconstruction of architectural 3D interiors.

Abstract

We introduce CADSpotting, an effective method for panoptic symbol spotting in large-scale architectural CAD drawings. Existing approaches often struggle with symbol diversity, scale variations, and overlapping elements in CAD designs, and typically rely on additional features (e.g., primitive types or graphical layers) to improve performance. CADSpotting overcomes these challenges by representing primitives through densely sampled points with only coordinate attributes, using a point-cloud backbone to learn robust features from 2D samples (embedded in 3D with $z = 0$). To enable accurate segmentation in large drawings, we further propose a novel Sliding Window Aggregation (SWA) technique that combines weighted voting and Non-Maximum Suppression (NMS). Moreover, we introduce LS-CAD, a new large-scale dataset comprising 45 finely annotated floorplans, each covering approximately 1,000 m^2, sig-

[†]Corresponding author

nificantly larger than prior benchmarks. LS-CAD will be publicly released to support future research. Experiments on FloorPlanCAD and LS-CAD demonstrate that CADSpotting significantly outperforms existing methods. We also showcase its practical value by enabling automated parametric 3D interior reconstruction directly from raw CAD inputs.

Keywords: Panoptic symbol spotting, large-scale CAD drawings, dense point sampling

1. Introduction

Architectural Computer-Aided Design (CAD) drawings, especially those used for interior design, serve as detailed digital representations that convey essential information about a building's structure, layout, and intricate details. These drawings include critical elements such as furniture placement, electrical layouts, and spatial organization, ensuring consistency and precision throughout the construction process. For instance, the Shanghai Mercedes-Benz Arena utilizes over 3,000 CAD drawings to guide its structural design, optimize spatial arrangements, and streamline the construction workflow. As CAD designs continue to grow in complexity, automating the detection and interpolation of symbols within these floorplans becomes increasingly important in various downstream tasks, such as code compliance checking and 3D interior modeling [19].

Despite the progress made in symbol recognition, the task of panoptic symbol spotting in CAD drawings remains under-explored. The challenges are multifaceted, arising from the vast diversity of symbol types, the need to differentiate between visually similar symbols, and the presence of overlapping elements. Additionally, variations in scale, orientation, and stylistic representation further complicate accurate identification.

Traditional query-by-example based methods [28] struggle with the diverse and complex graphical symbols in real-world CAD drawings. Learning-based approaches [24, 7] have advanced the field, with early methods using convolutional neural networks (CNNs) or graph-based models [6, 25, 39], and later works incorporating Transformers and attention mechanisms [5, 39] for improved global reasoning. However, these approaches typically operate on rasterized images derived from vector CAD data, leading to loss of geometric fidelity and recognition errors.

SymPoint [15] avoids rasterization by representing CAD drawings as point sets and applying point-cloud techniques for effective symbol spotting. SymPoint-V2 [16] further introduces layer-aware encoding and position-guided training. While effective, both methods assume a predefined set of primitive types, and SymPoint-V2 additionally requires reliable layer annotations, which are often inconsistent or unavailable in practice—hence they are brittle when unseen primitive types or mixed layer conventions appear in real CAD. Moreover, these approaches primarily target small- to medium-scale drawings and lack mechanisms to preserve spatial consistency in large layouts. As a result, panoptic symbol spotting on expansive CAD drawings remains challenging under dense clutter, scale variation, and heavy overlap.

In this paper, we present CADSpotting, a unified and robust panoptic symbol spotting framework specifically tailored for handling large-scale CAD drawings. Unlike prior approaches that treat CAD parsing as a graph classification task constrained by predefined geometric primitives (e.g., lines, arcs) or unreliable layer metadata, CADSpotting reformulates the problem into a unified point cloud segmentation task. By densely sampling points along primitives and discarding explicit type constraints, we construct a compact yet expressive representation where each point is defined solely by its coordinates. This coordinate-only formulation allows our model to generalize across diverse and complex shapes (e.g., Bézier curves) that traditional primitive-based methods fail to encode effectively. We employ a powerful point cloud backbone to learn robust features from this representation, followed by a streamlined Transformer decoder for panoptic prediction. Furthermore, to address the computational bottlenecks of processing large-scale real-world drawings, we introduce a novel Sliding Window Aggregation (SWA) strategy. SWA goes beyond simple cropping by integrating weighted voting and sparse Non-Maximum Suppression (NMS), ensuring consistent and scalable inference across window boundaries without the memory overhead of global vector processing.

On the data front, we further introduce LS-CAD, a new large-scale CAD dataset containing 45 finely annotated floorplans from diverse building types, including campuses and office complexes. Each floorplan spans over 1,000 m^2 on average and follows the same fine-grained annotation standards as FloorPlanCAD. LS-CAD is the first dataset of its scale in this domain, and it will be released under a license waiver to support future research. Experimental results on both FloorPlanCAD and LS-CAD demonstrate that CADSpotting outperforms the state-of-the-art methods, demonstrating its robustness and effectiveness in real-world scenarios. Finally, we demonstrate CADSpotting's practical value by automating 3D interior reconstruction. Leveraging instance-level outputs, we extract spatial parameters such as wall contours, door/window positions, orientations, and pivot points. These parameters are then used to drive parametric modeling in Blender, enabling efficient and accurate 3D interior generation directly from raw CAD inputs. To our knowledge, this constitutes the first end-to-end pipeline that tightly integrates CAD symbol spotting with parametric 3D reconstruction (Fig. 1).

2. Related Work

Panoptic Image Segmentation. Image segmentation is a fundamental task in computer vision, traditionally divided into two main categories: semantic segmentation [17, 2, 33, 1, 35] and instance segmentation [27, 8, 9, 12]. While semantic segmentation labels each pixel by class, instance segmentation further distinguishes between object instances of the same class. Recently, SAM [12] introduces a large model trained on over one billion masks, leveraging prompt-based interactions for enhanced interactive segmentation. SAM2 [23] extends this approach to video segmentation. However, these methods often struggle to represent uncountable background elements (e.g., sky, terrain). To unify countable and uncountable regions, Kirillov et al. [11] introduce panoptic segmentation, which unifies semantic and instance segmentation, effectively handling both countable and uncountable background components. Early approaches[30, 14, 3] use CNN backbones, while recent advances like Mask2Former [4] and OneFormer [9] leverage Transformer architectures with mask attention mechanisms for improved accuracy. Large models such as SEEM [40] further refines segmentation using a prompt-driven design. Despite these advances, most methods remain pixel-centric, limiting their effectiveness in processing vector graphics like CAD drawings. As a result, they struggle to capture critical vector-based information, including precise geometry and overlapping relationships.

CAD Symbol Spotting. Early CAD symbol spotting methods [22, 21, 26, 10, 36] rely on handcrafted feature descriptors, such as shape and structure, coupled with techniques like sliding window searches or graph matching for symbol retrieval. The introduction of deep learning significantly improved performance, with models based on Faster R-CNN [24] and YOLO [7] achieving higher accuracy [25], though they primarily focus on countable object instances. To address uncountable elements, Fan et al.[6] introduce the FloorPlanCAD dataset and a CNN-GCN framework for holistic parsing. Zheng et al.[39] further model CAD drawings as graphs, using adjacency prediction for segmentation. CADTransformer [5] adopt Vision Transformers to extract features from scalar maps. Inspired by advancements in 3D point cloud segmentation [38, 34, 13, 29], SymPoint [15] reformulates CAD drawings as primary point sets and applies point-based learning for effective symbol spotting. SymPoint-V2 [16] extends this with layer-aware encoding and position-guided training. Despite their effectiveness, both methods rely on a fixed set of primitive types, and SymPoint-V2 assumes the availability of reliable layer information. These assumptions often do not hold in real-world CAD data, limiting their ability to capture the full diversity and complexity of CAD symbols.

3. Method

An overview of our CADSpotting method is illustrated in Fig. 2. By incorporating a dense point sampling strategy within a unified point cloud processing model, our method effectively extracts features for CAD primitives, yielding a robust representation that supports various primitive types. We describe each module in detail in the following sections.

3.1. Primitive Feature Learning

SymPoint [15] represents CAD primitives as sets of 2D points and constructs handcrafted features for each, encoding attributes such as type (line, arc, circle, ellipse) and length. It then applies point cloud analysis with an Attention with Connection Module to learn primitive representations, followed by a masked-attention Transformer decoder for panoptic symbol spotting. While effective, SymPoint is limited by its reliance on manually defined features and a fixed set of four primitive types, which restricts its ability to handle the full diversity of CAD symbols. Real-world drawings often include complex elements like Bézier curves, making handcrafted features insufficiently robust. To overcome these limitations, we propose a novel primitive representation based on dense point sampling and unified point cloud learning, enabling more generalizable and expressive primitive-level features.

Dense Point Sampling In CAD drawings, graphical primitives are generally classified into line types (e.g., straight segments, curves, Bézier contours) and shape types (e.g., rectangles, circles, ellipses). Inspired by SymPoint [15], we adopt a dense point sampling strategy that performs equidistant sampling along each primitive to generate dense point-level data for feature learning. Unlike methods that rely on fixed primitive types and handcrafted features, our sampling approach is more flexible and generalizable, making it suitable for a wide variety of CAD elements. Fig. 3 illustrates the sampling process. Given a CAD drawing with N primitives, we sample points at a fixed interval d to construct a 2D point cloud P, where each point $p \in P$ is represented as a 3D vector $(x, y, z) \in \mathbb{R}^3$ with $z = 0$. We then use Point Transformer V3 [34] (PTv3) as the backbone to extract robust features, leveraging its design optimized for unordered point cloud data.

Feature Pooling To obtain the final feature representation for CAD primitives, we use Primitive Mixed Pooling (PMP) to convert point-wise features $\mathbf{f} \in \mathbb{R}^{M \times C}$ into primitive-wise features $\mathbf{g}_i \in \mathbb{R}^{N \times C}$.

$$\mathbf{g}_i = \mathcal{PMP}_i(\mathbf{f}) \quad \forall i \in \{1, ..., N\} \tag{1}$$

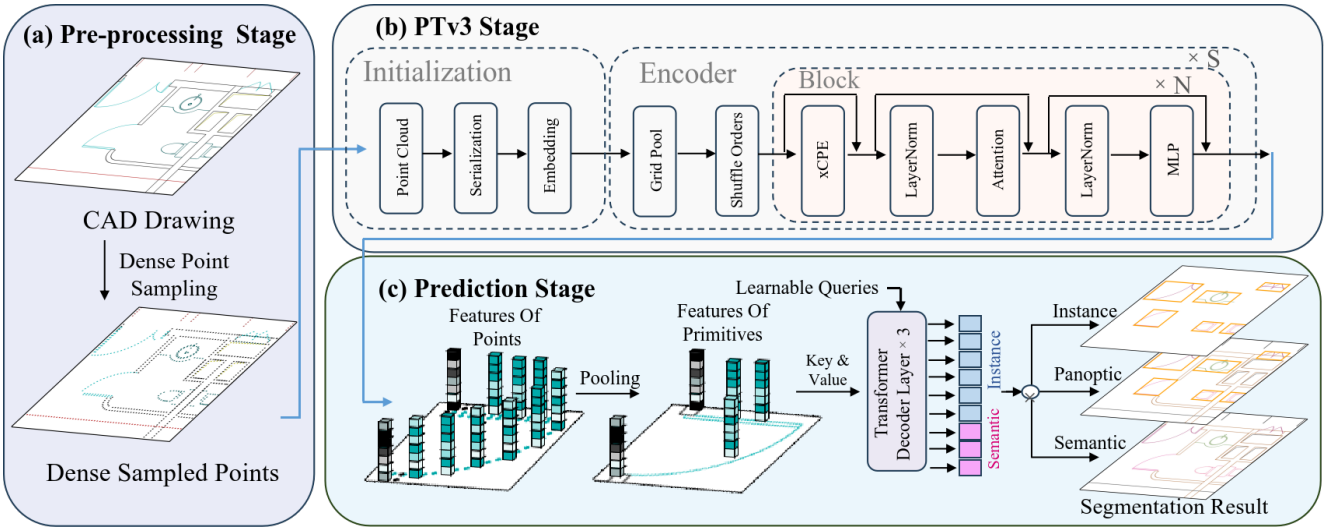


Figure 2: Overview of the CADSpotting pipeline. (a) Given a CAD drawing as input, CADSpotting first densely samples points along CAD graphic primitives to construct a point cloud representation, where each point is defined by its spatial coordinates. (b) Next, PTv3 extracts robust geometric features from the sampled point cloud.(c) Finally, after aggregating primitive-level features via mixed pooling, a streamlined Transformer decoder is employed for efficient panoptic symbol spotting.

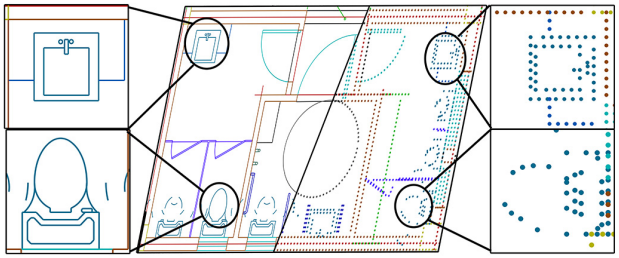


Figure 3: Dense point sampling. Left: A CAD drawing containing various primitives. Right: Dense point sampling converts each primitive symbol to a dense set of points, where d denotes the sampling interval (e.g., $d = 0.14$ units). A higher sampling density (smaller d) produces denser point sets.

where M is the total number of sampled points, and N is the number of primitives. Let $\mathbf{P}_i$ denote the dense point set corresponding to the i-th primitive. The c-th dimension of the primitive-level feature $\mathbf{g}_i$ is computed as the sum of the maximum and average values of the c-th dimension across all points in $\mathbf{P}_i$:

$$\mathbf{g}_i(c) = \mathcal{PMP}_i^c(\mathbf{f}) = \underbrace{\max_{p \in \mathbf{P}_i} \mathbf{f}_p(c)}_{\text{Max pooling}} + \underbrace{\frac{1}{|\mathbf{P}_i|} \sum_{p \in \mathbf{P}_i} \mathbf{f}_p(c)}_{\text{Average pooling}} \quad (2)$$

This pooling strategy compresses dense point features into a compact set of primitive-level representations, reducing computational and storage overhead. Meanwhile, it preserves essential geometric information, maintaining global consistency and minimizing information loss. Loss computation is also performed at the primitive level, aligning with the objective of CAD symbol spotting.

3.2. Panoptic Symbol Spotting

We utilize a lightweight Transformer decoder to extract semantic and instance-level information from the pooled primitive features. These features serve as keys and values for three consecutive decoder layers, where self-attention captures dependencies among primitives. Query embeddings are randomly initialized and optimized during training to compute accurate self-attention scores with other features.

Following OneFormer3D [13], the decoder output is divided into two parts: the first K_{ins} vectors correspond to instance proposals, and the remaining K_{sem} vectors represent semantic symbol spotting. The output has shape $N \times (\mathrm{K}_{\text{ins}} + \mathrm{K}_{\text{sem}})$, where the first K_{ins} dimensions are used to predict instance masks, and the rest indicate semantic class scores. This unified representation enables panoptic symbol spotting by combining both semantic and instance-level predictions.

Each semantic proposal includes log-likelihoods for each primitive being classified into different semantic labels and can be further processed through thresholding to generate a binary mask. Similarly, an instance proposal

also contains log-likelihood values, and these indicate the likelihood of each primitive being classified as a specific instance. During training, we apply bipartite matching using the Hungarian algorithm to align predictions with ground truth. At inference, we select the top-k instance proposals with the highest confidence and apply matrix-NMS [32] to eliminate redundant predictions.

Loss Function The overall loss function L consists of a classification loss and a primitive mask loss. The classification loss L_{cls} is defined as a multi-class cross-entropy loss. The mask loss combines binary cross-entropy L_{bce} and Dice loss [20] L_{dice}, jointly measuring the alignment between predicted instance masks and ground truth.

$$L = \lambda_{\text{cls}} L_{\text{cls}} + \lambda_{\text{bce}} L_{\text{bce}} + \lambda_{\text{dice}} L_{\text{dice}} \tag{3}$$

3.3. Sliding Window Aggregation

To address the challenge of panoptic symbol segmentation at large scales, a key issue is generalizing instance segmentation across drawings of varying sizes. During inference, larger inputs require significantly more instance proposals than those seen during training. However, Mask Transformer-based methods, including ours, are constrained by a fixed number of instance queries. Fortunately, CAD floorplans offer an advantage: instances of the same category typically maintain consistent sizes across different scenes under uniform scaling. This allows us to assume that instance sizes at inference do not exceed those in training data.

Based on this assumption, we adopt a fixed-size sliding window strategy, using windows aligned with training dimensions to process large input drawings. For each window, we collect all primitives it covers or intersects and apply the decoder to generate local semantic and instance proposals. Aggregated results across all windows are then merged using distinct strategies for semantic and instance outputs.

For semantic proposals, we apply a weighted voting scheme to assign each primitive a class label. When a primitive appears in multiple windows, we prioritize proposals from windows that observe a larger portion of it. The voting weight is proportional to the number of observed dense points relative to the total points of the primitive.

For instance proposals, we use Sparse Non-Maximum Suppression (Sparse-NMS) to merge results from overlapping windows efficiently (see supplementary material). By using a sufficiently small step size, each instance is fully captured in at least one window. Incomplete detections from other windows are eliminated during the NMS process.

Algorithm 1 details the implementation process of our SWA. In matrix-NMS [32] for sliding-window aggregation, computing the Intersection over Union (IoU) between all pairs of instance proposal masks is required to construct the IoU matrix. However, the number of instance proposals grows substantially with the size of the input drawing. If a dense matrix representation is used, both time and memory complexities become prohibitively high. Therefore, we adopt Sparse-NMS. To mitigate this issue, we observe that only a small fraction of proposal pairs exhibit nonzero overlap, as most proposals belong to different inference windows. Fig. 4 visualizes the sparsity structure of the IoU matrix during sliding window aggregation. By leveraging this sparsity, we reimplemented the IoU computation using sparse matrix routines, significantly reducing computational cost and memory usage without sacrificing accuracy.

Building	Area(m^2)	#Primitive	#Instance
Office 1	3,912	3,910	406
Office 2	1,799	28,193	1,185
Office 3	2,994	16,803	559
Campus 1	1,949	12,571	603
Hotel 1	1,193	26,709	335

Table 1: Summary statistics for three LS-CAD samples.

Algorithm 1 The pseudocode of SWA

Require: sliding steps, CAD drawing, proposals p
Ensure: `final_sem`, `final_inst`

Initialize `tot_sem` $\leftarrow [N, C]$
Initialize `tot_w` $\leftarrow [N]$
Initialize `tot_inst` $\leftarrow [\text{win\_num} \times p, N]$
for each sliding window w_i **do**
  `sem_i` $\leftarrow$ PRED_SEMANTIC(w_i)
  `inst_i` $\leftarrow$ PRED_INSTANCE(w_i)
  for each primitive Id pId **do**
    $\alpha \leftarrow$ OVERLAP_RATIO(pId, w_i)
    `prob` $\leftarrow$ SOFTMAX(`sem_i`[pId])
    `tot_sem`[pId] $\leftarrow$ `tot_sem`[pId] $+\alpha\cdot$ `prob`
    `tot_w`[pId] $\leftarrow$ `tot_w`[pId] $+\alpha$
  end for
  `tot_inst`[(i-1)*p : i*p] $\leftarrow$ `inst_i`
end for
for each primitive Id pId **do**
  `tot_sem`[pId] $\leftarrow$ `tot_sem`[pId]$/\max($`tot_w`[pId]$, \varepsilon)$
end for
`final_sem` $\leftarrow$ ARGMAX(`tot_sem`)
`final_inst` $\leftarrow$ SPARSE-NMS(`tot_inst`)

4. Experiments

4.1. The Proposed LS-CAD Dataset

We introduce LS-CAD, a new large-scale CAD dataset comprising 45 floorplans from expansive buildings, such as

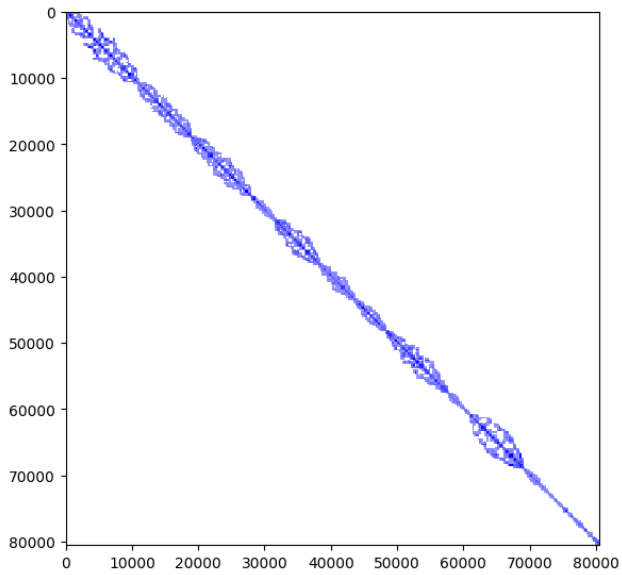


Figure 4: Visualization of the sparsity structure of the IoU matrix during sliding window aggregation for a sample drawing from the LS-CAD dataset. The Reverse Cuthill-McKee (RCM) algorithm was applied to reorder the matrix rows and columns, reducing its bandwidth and improving the visualization.

campuses and office complexes. Each drawing covers at least 1,000 square meters, with the number of primitives ranging from approximately 2,900 to over 10,000. Tab. 1 presents five representative examples, detailing area, primitive count, and instance count to highlight the large-scale nature of LS-CAD. The dataset features a rich variety of complex symbols, including doors, windows, walls, elevators, and parking spaces. All floorplans are annotated with fine-grained labels consistent with the FloorPlanCAD [6] standard. The LS-CAD dataset will be publicly released to support future research in panoptic symbol spotting for large-scale CAD drawings.

4.2. Experimental Settings

Implementation Details The FloorPlanCAD dataset comprises approximately 15,000 drawings, annotated across 35 symbol categories, including 30 thing and 5 stuff classes. To ensure fair comparison, we utilize the same training subset from the FloorPlanCAD dataset as used by SymPoint. In addition to FloorPlanCAD, we incorporate the training split of our LS-CAD dataset. Given the immense scale of LS-CAD drawings, we employ a sliding window cropping strategy to generate training samples. By filtering out regions with sparse primitives or empty space, we obtain approximately 3,000 additional high-quality window samples for training. We train our model on a machine with 8 NVIDIA A100 GPUs, using a batch size of 2 per GPU for 512 epochs. For dense point sampling, we set a fixed interval $d = 0.14$. Given that our input window size of 140×140 corresponds to a physical scale of $10\text{m} \times 10\text{m}$, this interval translates to a physical resolution of approximately 1cm. We use the AdamW [18] optimizer with a learning rate of 10^{-4} and a weight decay of 0.05. Data augmentation techniques include random horizontal flipping with a probability of 0.5, global rotation and scaling transformations, and translation along the x and y axes. We set loss weights as $\lambda_{\text{cls}} : \lambda_{\text{bce}} : \lambda_{\text{dice}} = 0.5 : 1 : 1$. Additionally, we set $K_{\text{sem}} = 36$ as the number of semantic labels, and both the top-k value and K_{ins} are set to 220. We use $\Delta = 70$ as the step size of SWA on the LS-CAD dataset.

Method	F1	wF1
PanCADNet [6]	80.6	79.8
CADTransformer [5]	82.2	90.1
GAT-CADNet [39]	85.0	82.3
SymPoint [15]	86.8	85.5
CADSpotting (ours)	**92.8**	**93.2**

Table 2: Quantitative comparison of semantic symbol spotting on FloorPlanCAD dataset.

Method	AP50	AP75	mAP
DINO [37]	64.0	54.9	47.5
SymPoint [15]	66.3	55.7	52.8
CADSpotting (ours)	**70.6**	**66.6**	**66.3**

Table 3: Quantitative comparison of instance symbol spotting on FloorPlanCAD dataset.

Metrics Following the approach of [5], we assess the performance of our model using multiple metrics. For semantic symbol spotting, we use F1 score and weighted F1 score (wF1). For instance symbol spotting, we employ AP50, AP75, and mean Average Precision (mAP). For panoptic symbol spotting, we utilize Panoptic Quality (PQ), Segmentation Quality (SQ), and Recognition Quality (RQ). PQ is a metric combining SQ and RQ to evaluate model performance in semantic and instance segmentation tasks. Further details on metric formulation are available in [5].

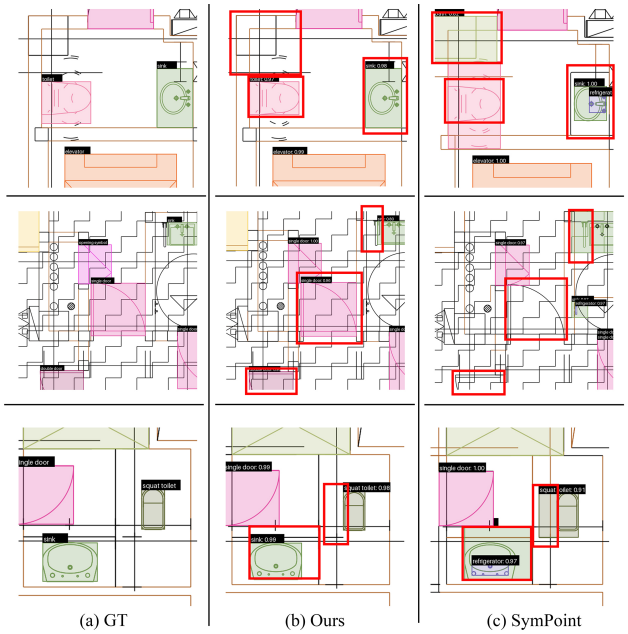
4.3. Quantitative Evaluation

To demonstrate the effectiveness of our approach, we compare it against SOTA methods on the FloorPlanCAD [6] and LS-CAD datasets. A detailed comparison of semantic, instance, and panoptic symbol spotting tasks is provided in the following paragraphs.

Semantic Symbol Spotting As shown in Tab. 2, we compare our method with PanCADNet [6], CADTransformer [5], GAT-CADNet [39], and SymPoint [15]. Our

Method	Total			Thing			Stuff		
	PQ	SQ	RQ	PQ	SQ	RQ	PQ	SQ	RQ
PanCADNet [6]	59.5	82.6	66.9	65.6	86.1	76.1	58.7	81.3	72.2
CADTransformer [5]	68.9	88.3	73.3	78.5	94.0	83.5	58.6	81.9	71.5
GAT-CADNet [39]	73.7	91.4	80.7	-	-	-	-	-	-
SymPoint [15]	83.3	91.4	91.1	84.1	**94.7**	88.8	48.2	69.5	69.4
CADSpotting (Ours)	**87.4**	**93.5**	**93.4**	**88.3**	94.2	**93.7**	**71.5**	**82.9**	**86.3**

Table 4: Quantitative comparison of panoptic symbol spotting on FloorPlanCAD dataset. Our method achieves the best performance among the comparison algorithms.



(a) GT (b) Ours (c) SymPoint

Figure 5: Qualitative comparison of panoptic symbol spotting.

CADSpotting achieves the highest performance in both the F1 and wF1 metrics.

Instance Symbol Spotting We also evaluate our method against DINO [37] and SymPoint for instance symbol spotting, using the bounding box maximization of predicted mask to compute the AP metric, consistent with SymPoint. As shown in Tab. 3, our method achieves superior results, with improvements of 4.3 p.p. AP50, 10.9 p.p. AP75 and 13.5 p.p. mAP over SymPoint.

Panoptic Symbol Spotting We compare our method with the same methods used in the semantic symbol spotting comparison. The results, presented in Tab. 4, show that our approach surpasses SymPoint and other prior methods across all metrics, including Total/Thing/Stuff PQ, SQ and RQ. Additionally, our approach offers superior generalization across diverse CAD primitives, thanks to its dense point sampling based feature representation.

Method	Panoptic			Semantic	Instance
	PQ	SQ	RQ	F1	mAP
SPv2	**90.1**	**96.3**	**93.6**	89.5	60.1
Ours	87.4	93.5	93.4	**92.8**	**66.3**

Table 5: Quantitative comparison with SPv2 on the FloorPlanCAD dataset.

Method	Panoptic			Semantic	Instance
	PQ	SQ	RQ	F1	mAP
SPv2 + BP	26.7	82.0	32.6	55.2	15.1
Ours + BP	58.8	**83.7**	70.2	88.7	48.3
Ours + SWA	**75.5**	80.9	**93.3**	**93.5**	**57.5**

Table 6: Quantitative comparison of semantic and panoptic symbol spotting on LS-CAD dataset.

We then compare our method with the concurrent work SymPoint-V2 (SPv2) [16], which incorporates primitive types into feature vectors. Our first comparison is conducted on the FloorPlanCAD dataset. As shown in Tab. 5, SPv2 achieves superior SQ, likely benefiting from geometric priors inherent in vector primitives that yield smoother masks. However, our method delivers a 3.3-point gain in F1 score and a 6.2-point improvement in mAP. This demonstrates that while our point-based representation may produce slightly discretized boundaries, it achieves significantly stronger detection robustness compared to SPv2. While SPv2 performs well on FloorPlanCAD, its reliance on four fixed primitive types limits its generalization to more diverse CAD symbol sets. In contrast, our method relies solely on coordinate attributes, enabling more flexible and robust feature learning. Furthermore, SPv2 depends on CAD layer information, which is often inconsistent or unavailable in real-world scenarios. Tab. 6 further supports this analysis. On the LS-CAD dataset, where layer annotations are unavailable, our method significantly outperforms SPv2 across all metrics, demonstrating stronger robustness in practical applications.

We also evaluate the effectiveness of Sliding Window Aggregation (SWA) versus Block Partitioning (BP) in CADSpotting on the LS-CAD dataset. In the BP setting, we divide large-scale CAD drawings into non-overlapping

blocks matching the dimensions used in FloorPlanCAD. To ensure unique primitive representation, only those with starting points inside a block are retained. As shown in Tab. 6, SWA substantially outperforms BP in panoptic quality (PQ), achieving a 16.7-point improvement, with only a modest 2.8-point reduction in segmentation quality (SQ). These results demonstrate that CADSpotting, when combined with SWA, achieves accurate and scalable symbol detection in large-scale CAD drawings.

4.4. Qualitative Comparison

We perform qualitative comparisons with SymPoint on the FloorPlanCAD dataset. Fig. 5 shows that our method achieves accurate and robust panoptic symbol spotting, even in challenging scenarios involving walls, complex or overlapping symbols, and uncommon furniture symbols. This demonstrates the effectiveness of our proposed dense point sampling based primitive features. Additionally, Fig. 6 presents our semantic and panoptic symbol spotting results on a large-scale CAD drawing of an office building with 28,193 primitives and 1,185 instances from our LS-CAD dataset. These results demonstrate that our CADSpotting enhanced by SWA technique, accurately identifies semantic and instance information in large-scale CAD drawings.

To evaluate model generalizability and demonstrate the utility of LS-CAD dataset, we also train models on a combined dataset consisting of FloorPlanCAD and LS-CAD. Comprehensive experimental results and analyses are provided in the supplementary material.

4.5. Ablation Studies

We further present ablation studies on feature learning. Additional ablation studies, which explore various pooling strategies and the use of color information, and evaluate performance across all classes, are provided in the supplementary material.

Feature Learning We compare our dense point sampling based primitive feature learning method with the approach in SymPoint, which uses handcrafted features to represent each individual primitive. SymPoint employs Point Transformer(PTv1) [38] as its backbone network and incorporates hierarchical multi-resolution primitive features to leverage intermediate features, enhancing the decoder's performance. To evaluate the impact of different point cloud analysis methods, we first replace SymPoint's backbone with PTv3 [34], maintaining the same decoder architecture as ours to ensure a fair comparison of feature learning techniques in panoptic symbol spotting. As shown in Tab. 7, with the same decoder configuration, simply switching to PTv3 results in a 1.8 p.p. improvement in PQ. When we apply our proposed primitive feature learning method, performance further increases 7.2 p.p. in PQ, demonstrating the significant performance gains achieved by our dense point sampling based feature learning approach.

Method	Backbone	PQ	SQ	RQ
Handcrafted Features	PTv1	78.4	88.5	88.6
Handcrafted Features	PTv3	80.2	90.2	89.0
Ours	PTv3	**87.4**	**93.5**	**93.4**

Table 7: Ablation study on feature learning methods on FloorPlanCAD dataset. All methods utilize the same decoder architecture to learn intermediate features.

Primitive Pooling Type	PQ	SQ	RQ
w/o Pooling	81.9	89.4	91.6
Max	86.9	93.3	93.2
Average	87.0	93.1	**93.4**
Mixed (Max+Average)	**87.4**	**93.5**	**93.4**

Table 8: Ablation study on different pooling methods test on FloorPlanCAD dataset.

Method	PQ	SQ	RQ
w/o color	87.4	93.5	93.3
w/ color	88.9	95.6	93.0

Table 9: Ablation study on with and without prior color informantion as input.

Pooling. We further evaluate the necessity and impact of different pooling methods on our approach, using a baseline configuration without a pooling layer. As shown in Tab. 8, max pooling improves performance by 5.0 p.p., average pooling by 5.1 p.p., and mixed pooling by 5.5. p.p. in PQ. Based on these results, we select mixed pooling for our CADSpotting approach.

Prior Color Information. To evaluate the necessity of color prior, we incorporate RGB channels as an optional input. As shown in Table 9, adding color improves PQ by 1.5 p.p. and SQ by 2.1 p.p., while slightly decreasing RQ by 0.4 p.p. More importantly, the geometry-based representation (*w/o Color*) achieves strong baseline performance, demonstrating that our model maintains high recognition quality even without color information.

Cross-Dataset Joint Training. To demonstrate the generalization capability of our model and highlight the value of the LS-CAD dataset, we implement a cross-dataset joint training paradigm. We randomly partition complete large-scale CAD drawings into three subsets: 80% for training, 10% for validation, and 10% for testing. The division of the LS-CAD dataset strictly adheres to the standardized data splitting protocol established in FloorPlanCAD. The training and validation sets from both datasets are subsequently integrated to construct a cross-dataset collaborative training

Figure 6: Semantic and panoptic symbol spotting results of CADSpotting with the SWA technique on a large-scale office building CAD drawing from our LS-CAD dataset. The figure shows full CAD drawings with Ground Truth(GT) and predicted results in the center, with close-up views(as indicated by the red ellipses) of GT in (a, c) and predicted results in (b, d).

Method	Training Dataset	Test on FloorPlanCAD			Test on LS-CAD		
		PQ	SQ	RQ	PQ	SQ	RQ
SymPoint [15]	FloorPlanCAD	83.3	91.4	91.1	33.2	83.0	39.9
SymPoint	Cross-dataset	79.0	89.2	88.6	57.0	84.0	67.8
CADSpotting	FloorPlanCAD	**87.4**	**93.5**	**93.4**	52.9	90.4	58.5
CADSpotting	Cross-dataset	84.0	91.3	91.9	58.8	83.7	70.2
CADSpotting(SWA)	FloorPlanCAD	-	-	-	60.1	**90.9**	66.1
CADSpotting(SWA)	Cross-dataset	-	-	-	**75.5**	80.9	**93.3**

Table 10: Cross-dataset joint training performance comparison between SymPoint and CADSpotting.

environment. As shown in Tab. 10, the cross-dataset joint training strategy significantly improves model performance on the LS-CAD test set, although with slight performance degradation on the original FloorPlanCAD test set. This demonstrates the necessity of the proposed LS-CAD dataset for providing more diverse CAD representations. Notably, CADSpotting+SWA method reaches the best performance of 75.5 on the PQ metric. After cross-dataset joint training, the PQ score for CADSpotting decreases by only 3.4 p.p. on FloorPlanCAD, whereas it drops by 4.3 p.p. on SymPoint. This indicates that our model exhibits stronger generalization capabilities.

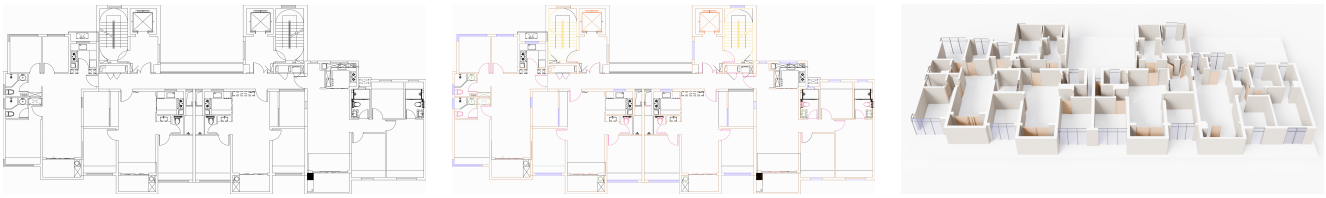
Inference Efficiency To address concerns regarding computational cost, we evaluate the inference latency of different configurations on the FloorPlanCAD test set using a single NVIDIA A100 GPU. As shown in Table 11, replacing the PTv1 backbone with PTv3 in the handcrafted feature baseline (SymPoint setting) reduces latency from 83ms to 60ms due to PTv3's optimized sparse operations. While our dense point sampling strategy inherently increases the input data volume (raising latency to 133ms without pooling), the introduction of Primitive Mixed Pooling effectively compresses the feature space. Consequently, our final model achieves an inference speed of 90ms per drawing.

Method	Backbone	Params	Pooling	Latency
Handcrafted	PTv1	35M	×	83ms
Handcrafted	PTv3	49M	×	60ms
Ours	PTv3	49M	×	133ms
Ours	PTv3	49M	✓	90ms

Table 11: Comparison of model size (Params) and average inference latency on a single FloorPlanCAD drawing using an NVIDIA A100 GPU.

5. Application in Automated 3D Interior Reconstruction

CADSpotting produces accurate panoptic segmentation results that serve as the foundation for our parametric 3D interior modeling pipeline. Using instance-level segmentation and primitive positions, we extract spatial parameters for key architectural elements, including walls, doors, and windows. For doors, we compute positions, orientations, and pivot points by analyzing geometric relationships between arcs and lines within each segmented instance. Window positions are directly derived from their corresponding



(a) CAD drawing as raw input (b) Prediction from CADSpotting (c) Automated reconstruction

Figure 7: Our CADSpotting takes a CAD drawing as input (a) and outputs predicted semantic and instance segmentation results (b). After that, we design an automated reconstruction pipeline to generate 3D interior models (c).

instances. Wall modeling involves a more specialized process: we merge endpoints of adjacent lines within wall instances to form closed polygons, which are rasterized into binary masks. Final wall contours are then extracted using the method from [31]. These spatial parameters are fed into a parametric reconstruction pipeline implemented in Blender. Predefined 3D assets for walls, doors, and windows are used as reusable templates, enabling efficient and automated generation of structurally accurate 3D interiors. This approach converts raw CAD inputs into detailed 3D models within minutes. An example result is shown in Fig. 7, with further details provided in the supplementary material.

By systematically optimizing the parameters of predefined instance components using instance segmentation data and primitive spatial coordinates, we achieve accurate 3D model reconstruction through component-level geometric adaptation

For wall polygon extraction, we employ an innovative SVG-PNG conversion strategy to differentiate floor areas from walls: 1) Merge adjacent endpoints and remove duplicate segments to simplify SVG paths. 2) Convert vectors to 8K resolution binarized PNG while maintaining coordinate alignment. 3) Identify the largest connected component as the floor through connected component analysis, with secondary components as candidate walls. 4) Re-vectorize boundaries through coordinate mapping to generate closed wall polygons.

The proposed method exhibits notable robustness against recognition inaccuracies – incomplete walls are automatically reconciled during stage 1, while small noise particles are filtered in stage 3 using area thresholds.

For doors and windows, we employ category-specific parameterization strategies. Doors are systematically categorized into four distinct subtypes (single/double/sliding/folding) based on semantic labels, with positions, orientations, and pivot points determined through linear-arc spatial relationships.

Windows are grouped via union-find algorithms, calculating average lines of co-grouped segments as representative lines, then selecting the segment closest to the representative line within each group as the window instance position.

The final reconstruction leverages Blender's geometry nodes for procedural modeling: 1) Extrude wall polygons to 3D volumes using floorplan height attributes. 2) Instance-place door/window assemblies at inferred positions. 3) Apply floor based on origin CAD drawing.

This integrated approach achieves automated 3D reconstruction with exceptional efficiency, transforming raw CAD data into structurally complete 3D interior models within minutes through streamlined procedural generation.

6. Conclusion

We have introduced CADSpotting, a novel method for panoptic symbol spotting in large-scale CAD drawings. Our approach uses dense point sampling within a unified point cloud framework to represent CAD primitives, combined with a Sliding Window Aggregation technique that integrates weighted voting and Non-Maximum Suppression for precise segmentation. We also present LS-CAD, a dataset of 45 annotated floorplans from diverse building types. Experiments on LS-CAD and FloorPlanCAD datasets have confirmed the effectiveness and scalability of our method, with successful applications in automated 3D reconstruction demonstrating its practical value.

Our approach currently faces limitations due to the dataset's primary emphasis on residential and commercial buildings, which restricts its applicability to industrial and cultural contexts. Additionally, it may struggle to capture significant stylistic variations among CAD symbols. Addressing these challenges through the integration of in-context learning may help enhance annotation efficiency and extend applicability. We leave this for future work.

References

[1] L.-C. Chen. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 3

[2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully con-

nected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 3

[3] L.-C. Chen, H. Wang, and S. Qiao. Scaling wide residual networks for panoptic segmentation. *arXiv preprint arXiv:2011.11675*, 2020. 3

[4] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1290–1299, 2022. 3

[5] Z. Fan, T. Chen, P. Wang, and Z. Wang. Cadtransformer: Panoptic symbol spotting transformer for cad drawings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10986–10996, 2022. 2, 3, 6, 7

[6] Z. Fan, L. Zhu, H. Li, X. Chen, S. Zhu, and P. Tan. Floorplancad: A large-scale cad drawing dataset for panoptic. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10128–10137, 2021. 2, 3, 6, 7

[7] A. Farhadi and J. Redmon. Yolov3: An incremental improvement. In *Computer vision and pattern recognition*, volume 1804, pages 1–6. Springer Berlin/Heidelberg, Germany, 2018. 2, 3

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 3

[9] J. Jain, J. Li, M. T. Chiu, A. Hassani, N. Orlov, and H. Shi. Oneformer: One transformer to rule universal image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2989–2998, 2023. 3

[10] X. Jiang, L. Liu, C. Shan, Y. Shen, X. Dong, and D. Li. Recognizing vector graphics without rasterization. *Advances in Neural Information Processing Systems*, 34:24569–24580, 2021. 3

[11] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9404–9413, 2019. 3

[12] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 3

[13] M. Kolodiazhnyi, A. Vorontsova, A. Konushin, and D. Rukhovich. Oneformer3d: One transformer for unified point cloud segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20943–20953, 2024. 3, 4

[14] Y. Li, X. Chen, Z. Zhu, L. Xie, G. Huang, D. Du, and X. Wang. Attention-guided unified network for panoptic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7026–7035, 2019. 3

[15] W. LIU, T. Yang, Y. Wang, Q. Yu, and L. Zhang. Symbol as points: Panoptic symbol spotting via point-based representation. In *The Twelfth International Conference on Learning Representations*, 2024. 2, 3, 6, 7, 9

[16] W. Liu, T. Yang, Q. Yu, and L. Zhang. Sympoint revolutionized: Boosting panoptic symbol spotting with layer feature enhancement. *arXiv preprint arXiv:2407.01928*, 2024. 2, 3, 7

[17] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 3

[18] I. Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6

[19] MarsBIM. Cad to bim conversion. https://www.marsbim.com/services/bim/cad-to-bim-conversion/, 2024. Accessed: 2024-11-10. 2

[20] F. Milletari, N. Navab, and S.-A. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. Ieee, 2016. 5

[21] T.-O. Nguyen, S. Tabbone, and A. Boucher. A symbol spotting approach based on the vector model and a visual vocabulary. In *2009 10th International Conference on Document Analysis and Recognition*, pages 708–712. IEEE, 2009. 3

[22] T.-O. Nguyen, S. Tabbone, and O. R. Terrades. Symbol descriptor based on shape context and vector model of information retrieval. In *2008 The Eighth IAPR International Workshop on Document Analysis Systems*, pages 191–197. IEEE, 2008. 3

[23] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 3

[24] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016. 2, 3

[25] A. Rezvanifar, M. Cote, and A. B. Albu. Symbol spotting on digital architectural floor plans using a deep learning-based framework. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2419–2428, 2020. 2, 3

[26] A. Rezvanifar, M. Cote, and A. Branzan Albu. Symbol spotting for architectural drawings: state-of-the-art and new industry-driven developments. *IPSJ Transactions on Computer Vision and Applications*, 11:1–22, 2019. 3

[27] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015. 3

[28] M. Rusiñol, J. Lladós, and G. Sánchez. Symbol spotting in vectorized technical drawings through a lookup table of region strings. *Pattern Analysis and Applications*, 13:321–331, 2010. 2

[29] J. Schult, F. Engelmann, A. Hermans, O. Litany, S. Tang, and B. Leibe. Mask3d: Mask transformer for 3d semantic instance segmentation. In *2023 IEEE International Confer-*

ence on Robotics and Automation (ICRA), pages 8216–8223. IEEE, 2023. 3

[30] K. Sun, Y. Zhao, B. Jiang, T. Cheng, B. Xiao, D. Liu, Y. Mu, X. Wang, W. Liu, and J. Wang. High-resolution representations for labeling pixels and regions. *arXiv preprint arXiv:1904.04514*, 2019. 3

[31] S. Suzuki and K. Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985. 10

[32] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen. Solov2: Dynamic and fast instance segmentation. *Advances in Neural information processing systems*, 33:17721–17732, 2020. 5

[33] H. Wu, J. Zhang, K. Huang, K. Liang, and Y. Yu. Fastfcn: Rethinking dilated convolution in the backbone for semantic segmentation. *arXiv preprint arXiv:1903.11816*, 2019. 3

[34] X. Wu, L. Jiang, P.-S. Wang, Z. Liu, X. Liu, Y. Qiao, W. Ouyang, T. He, and H. Zhao. Point transformer v3: Simpler faster stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4840–4851, 2024. 3, 8

[35] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34:12077–12090, 2021. 3

[36] B. Yang, H. Jiang, H. Pan, and J. Xiao. Vectorfloorseg: Two-stream graph attention network for vectorized roughcast floorplan segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1358–1367, 2023. 3

[37] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. Ni, and H.-Y. Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. In *The Eleventh International Conference on Learning Representations*, 2022. 6, 7

[38] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268, 2021. 3, 8

[39] Z. Zheng, J. Li, L. Zhu, H. Li, F. Petzold, and P. Tan. Gatcadnet: Graph attention network for panoptic symbol spotting in cad drawings. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11747–11756, 2022. 2, 3, 6, 7

[40] X. Zou, J. Yang, H. Zhang, F. Li, L. Li, J. Wang, L. Wang, J. Gao, and Y. J. Lee. Segment everything everywhere all at once. *Advances in Neural Information Processing Systems*, 36, 2024. 3