

TempFit: Panoptic Symbol Spotting via Template Fitting

HONGHAO XU, Shenzhen University, China

HAO WANG, SpeedBot Robotics Co., Ltd, China

XIANG LIU, Shenzhen University, China

JUZHAN XU, Shenzhen University, China

MANFRED LAU, Lakehead University, Canada

KAI XU, Institute of AI for Industries, Chinese Academy of Sciences and Jiangsu Key Laboratory of AI for Industries, China

RUIZHEN HU*, Shenzhen University, China

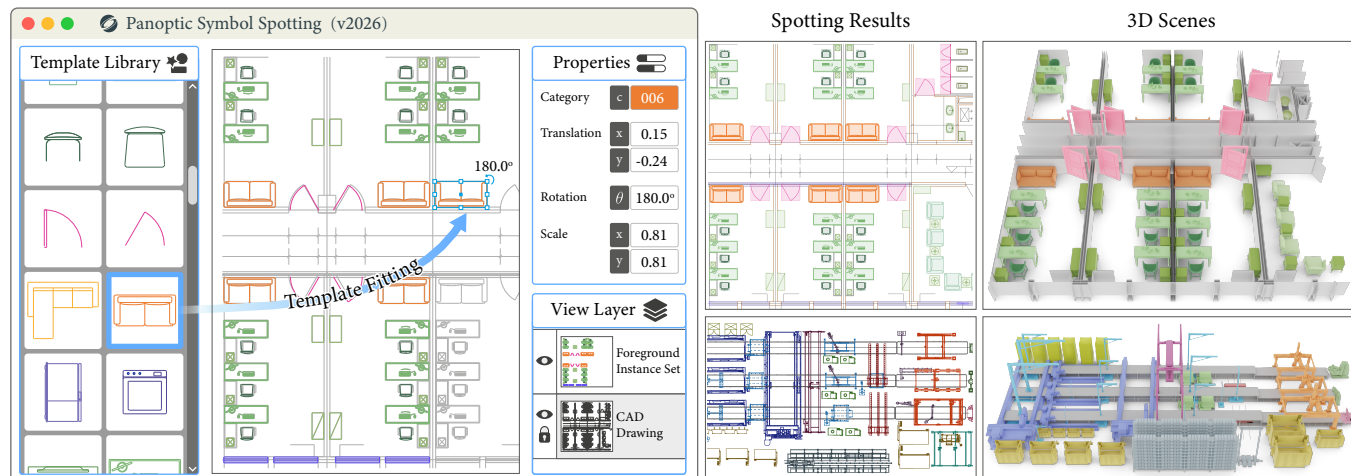


Fig. 1. Motivated by the professional “retrieve-and-transform” CAD design workflow, we reformulate Panoptic Symbol Spotting as *Template Fitting*. **Left:** given an input CAD drawing, our method retrieves the matching template from a predefined library and aligns it via pose prediction to produce the panoptic result. **Right:** since each template can be canonically associated with a 3D model, the detected symbols directly support downstream 3D applications such as digital twin reconstruction, illustrated here on floorplan (top) and industrial (bottom) scenes.

In this paper, we introduce *TempFit*, a novel approach for the problem of Panoptic Symbol Spotting on CAD drawings. Existing methods treat symbol spotting as a per-primitive classification or mask-prediction task, with no explicit constraint enforcing that the primitives assigned to one instance form a spatially connected group. As a result, predictions may erroneously merge primitives from distant symbols into a single instance or fragment a coherent symbol across multiple predictions. Motivated by the professional “retrieve-and-transform” CAD design workflow, we reformulate this task as a *Template Fitting* problem, where each foreground instance is detected by

retrieving and aligning a canonical template from a library, enforcing structural completeness by construction. Building on this formulation, we present an autoregressive transformer that unifies template retrieval and pose prediction within a single end-to-end pipeline. The framework treats the template library as an explicit geometric memory, parsing background regions in parallel via learnable background tokens and synthesizing foreground instances autoregressively while attending to template-specific geometry at each step. Experiments on FloorplanCAD and our newly collected IndustryCAD dataset, a more challenging benchmark with complex industrial symbols, heavy occlusion, and wide scale variation, demonstrate state-of-the-art performance with particularly large gains on IndustryCAD. Project page: <https://tempfit-project.github.io/>.

*Corresponding author: Ruizhen Hu (ruizhen.hu@gmail.com)

Authors’ Contact Information: Honghao Xu, Shenzhen University, Shenzhen, China, littledaisy20001227@gmail.com; Hao Wang, SpeedBot Robotics Co., Ltd, Changsha, China, 21020100007@pop.zjgsu.edu.cn; Xiang Liu, Shenzhen University, Shenzhen, China, lyr02302561@gmail.com; Juzhan Xu, Shenzhen University, Shenzhen, China, juzhan.xu@gmail.com; Manfred Lau, Lakehead University, Orillia, Canada, manfred.lau@gmail.com; Kai Xu, Institute of AI for Industries, Chinese Academy of Sciences and Jiangsu Key Laboratory of AI for Industries, Nanjing, China, kevin.kai.xu@gmail.com; Ruizhen Hu, Shenzhen University, Shenzhen, China, ruizhen.hu@gmail.com.

CCS Concepts: • **Computing methodologies** → *Shape analysis*.

Additional Key Words and Phrases: Panoptic Symbol Spotting, Primitive Fitting

ACM Reference Format:

Honghao Xu, Hao Wang, Xiang Liu, Juzhan Xu, Manfred Lau, Kai Xu, and Ruizhen Hu. 2026. TempFit: Panoptic Symbol Spotting via Template Fitting. *ACM Trans. Graph.* 45, 6, Article 213 (December 2026), 14 pages. <https://doi.org/10.1145/3842584>



This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2026 Copyright held by the owner/author(s).
ACM 1557-7368/2026/12-ART213
<https://doi.org/10.1145/3842584>

1 Introduction

Panoptic symbol spotting [Fan et al. 2021] aims to simultaneously detect and segment countable foreground object instances (e.g., furniture in a floorplan) while parsing uncountable background regions (e.g., walls) from 2D CAD drawings. In architecture, engineering and construction (AEC) industries, this task underpins critical downstream applications such as building information modeling (BIM) and 3D digital twin reconstruction, where accurate instance-level parsing is essential for faithfully recovering the structural semantics from the given CAD drawings. However, this problem is highly non-trivial, as real-world drawings often suffer from severe clutter, symbol occlusions, and significant variations in appearance and structure within the same category [Fan et al. 2022].

To tackle this challenging task, existing methods can be broadly categorized into two paradigms: primitive-wise and instance-wise methods. Primitive-wise methods [Fan et al. 2022, 2021; Zheng et al. 2022] treat each graphical primitive as an independent unit, employing GNNs or Transformers to predict a semantic label for each primitive, followed by heuristic grouping techniques to assemble them into object instances. Instance-wise methods [Liu et al. 2024a,b; Luo et al. 2025; Wei et al. 2025; Yang et al. 2024] adopt a query-based segmentation prediction paradigm, where each query outputs a class label and a binary mask indicating which primitives belong to that instance. Despite their differences, both paradigms share a fundamental limitation: they treat symbol spotting as a per-primitive classification or mask-prediction problem, without explicitly modelling the local spatial topology within each symbol—such as the connectivity of lines and arcs that compose a chair or a door. Consequently, no constraint enforces that the primitives assigned to one instance form a spatially connected group, and predictions may erroneously merge primitives from distant symbols into a single instance or fragment a coherent symbol across multiple predictions.

To overcome this spatial incoherence, we reformulate the task as a *Template Fitting* problem, motivated by the “Retrieve-and-Transform” workflow of professional CAD design: designers maintain a library of canonical templates and compose drawings by retrieving templates and adjusting their poses, rather than drawing stroke-by-stroke. Following this paradigm, each foreground symbol is modeled as the instantiation of a template under a 2D transformation—the model retrieves the best-matching template and predicts the transformation parameters that align it with the target region. Since every prediction is, by construction, a valid template, its primitives automatically form a structurally complete and topologically coherent group, directly eliminating the merge and fragmentation errors common in prior paradigms. Each template can also be canonically associated with a 3D model, enabling detected symbols to directly serve downstream BIM and digital twin reconstruction, as shown in Figure 1. We argue that this Template Fitting formulation represents a more principled approach to Panoptic Symbol Spotting, as it is fundamentally aligned with how CAD drawings are originally authored.

While the Template Fitting formulation is conceptually appealing, executing it effectively for 2D CAD symbol spotting is non-trivial. This formulation has been explored in two adjacent settings, but

neither solution transfers directly to our problem. The first is geometric shape abstraction [Huang et al. 2023; Xu et al. 2023b; Ye et al. 2025], which approximates a 3D shape with a set of parameterized templates such as cuboids or cylinders. In this setting the template library is small and fixed, so existing methods typically memorize template geometry implicitly within the network weights, rather than treating it as an explicit input. This design prevents generalization to unseen templates and is unsuitable for our task, where the template library is large and the shapes are too complex to be reliably memorized by a network. The second is scene CAD recomposition [Avetisyan et al. 2019; Kuo et al. 2020; Leng et al. 2024; Li et al. 2023], which aligns observed objects in a 3D scan or RGB image with canonical CAD meshes from an external library. This setting is closer to ours: the template library is large and complex, and its geometry is provided as explicit input. However, these methods adopt a staged detect-then-fit pipeline: an object detector first localizes each instance based on category-level appearance alone, and the template retrieval and pose-fitting steps then operate on each detected region independently. This decoupling causes per-instance error propagation—a missed or misaligned detection constrains the downstream fitting result for the same instance.

To address these challenges, we introduce *TempFit*, an autoregressive transformer framework that unifies template retrieval and pose prediction into a single, end-to-end process. The core idea is to treat the template library as an explicit geometric memory: each template is encoded into a sequence of tokens and prepended to the input, enabling the model to attend to detailed shape geometry throughout the fitting process. The scene is processed through two components. Background regions are parsed in parallel via learnable background tokens, each predicting a category mask for a specific region type. Foreground instances are generated autoregressively: at each step, the model attends over the encoded template memory to jointly select and align a template with its target region in the drawing. This design directly addresses the limitations of prior Template Fitting approaches. This design avoids memorizing geometry in network weights (unlike shape abstraction) and eliminates the error accumulation of staged pipelines (unlike scene CAD recomposition).

To validate the effectiveness of our proposed method, we conducted experiments on the FloorPlanCAD [Fan et al. 2021] dataset, where our method achieves significant improvements over state-of-the-art approaches. Furthermore, we collected IndustryCAD, a new industrial CAD dataset that is more challenging than FloorPlanCAD in three aspects: greater symbol complexity, more severe occlusion, and wider scale variation. On this dataset, our method yields even larger gains over prior methods.

Our main contributions can be summarized as follows:

- We propose a novel perspective that reformulates Panoptic Symbol Spotting as a *Template Fitting* task, inherently enforcing structural completeness and topological consistency for each symbol, aligned with how CAD drawings are authored.
- We introduce *TempFit*, an autoregressive transformer that treats the template library as an explicit geometric memory and unifies template retrieval with pose prediction into a single, end-to-end process, eliminating the error accumulation of staged pipelines while generalizing to large and extensible template libraries.

- We conduct extensive experiments on FloorPlanCAD and our newly collected IndustryCAD dataset, achieving state-of-the-art results on both benchmarks with particularly large gains on the more challenging IndustryCAD.

2 Related Work

Our work intersects three lines of research: Panoptic Symbol Spotting, the target task we aim to improve; 3D Shape Abstraction, which fits parametric templates to approximate geometry; and Scene CAD Recomposition, which aligns canonical CAD meshes with visual observations. We review each and position TempFit accordingly.

2.1 Panoptic Symbol Spotting

The Panoptic Symbol Spotting task was introduced by [Fan et al. 2021] along with the FloorPlanCAD benchmark, requiring the joint parsing of countable instances (e.g., furniture) and uncountable semantic regions (e.g., walls) from CAD drawings. Existing approaches fall into two paradigms: Primitive-wise and Instance-wise methods.

Primitive-wise Methods. Early approaches typically treat each graphical primitive as an independent processing unit. The core paradigm involves using Graph Neural Networks (GNNs) or Transformers to predict a semantic label for each primitive individually, followed by heuristic grouping techniques (e.g., bounding box detection or clustering) to assemble them into instances. For example, PanCADNet [Fan et al. 2021] combines graph convolutional networks [Jiang et al. 2019] for semantic segmentation with Faster R-CNN [Ren et al. 2015] for instance grouping. GAT-CADNet [Zheng et al. 2022] employs Graph Attention Networks to capture topological relations between primitives, while CADTransformer [Fan et al. 2022] utilizes Vision Transformers [Dosovitskiy 2020] on rasterized inputs to aggregate primitive features.

Instance-wise Methods. Following the success of DETR [Carion et al. 2020], the field has shifted towards query-based set prediction. These methods formulate the task as predicting a set of instance masks, where each query outputs a semantic class and a binary mask $\in \{0, 1\}^N$ indicating which primitives belong to that instance. Based on their input representation, these methods can be categorized into three groups: (1) Point-based: SymPoint [Liu et al. 2024a] and SymPoint-V2 [Liu et al. 2024b] treat each primitive as a single point; (2) Dense-point-based: CADSpotting [Yang et al. 2024] densely samples points along primitives to learn richer features; (3) Line-based: VecFormer [Wei et al. 2025] directly processes vector segments to better preserve geometric fidelity.

Despite their differences, both primitive-wise and instance-wise paradigms share a fundamental limitation: they treat symbol spotting as a per-primitive classification or mask-prediction task, with no explicit constraint enforcing that the primitives assigned to one instance form a spatially connected group. Consequently, predictions may erroneously merge primitives from distant symbols into a single instance, or fragment a coherent symbol across multiple predictions. TempFit reformulates the task as template fitting, where each predicted instance is by construction an instantiation of a valid template whose primitives form a topologically coherent group.

2.2 3D Shape Abstraction

Shape Abstraction aims to approximate complex geometry using a set of parameterized primitives. Existing work spans a spectrum of geometric complexity. Simple forms include cuboids [Gupta et al. 2010; Li et al. 2017; Mo et al. 2019; Tulsiani et al. 2017] and cylinders [Binford 1975]. More expressive representations include superquadrics [Paschalidou et al. 2019; Pentland 1986], convex polytopes [Chen et al. 2020; Deng et al. 2020], learned implicit fields [Chen et al. 2019; Gadelha et al. 2020; Genova et al. 2019], and B-spline patches [Sharma et al. 2020].

These methods broadly fall into two paradigms: (1) Optimization-based methods directly optimize parameters to minimize reconstruction error [Liu et al. 2023; Monnier et al. 2023; Pitas and Venetsanopoulos 2002]. While mathematically precise, they often fragment semantic objects into incoherent pieces as they lack semantic awareness. (2) Learning-based methods employ neural networks to predict primitive parameters directly, trained via supervised [Jones et al. 2020; Li et al. 2017, 2019; Mo et al. 2019; Ye et al. 2025] or unsupervised [Chen et al. 2024, 2020; Deng et al. 2020; Genova et al. 2020; Sun et al. 2019] reconstruction objectives.

Applying these methods to CAD symbol spotting is fundamentally limited by their reliance on a small, fixed primitive library. Because the library is small, existing methods memorize primitive geometry implicitly within the network weights rather than treating it as an explicit input. This implicit encoding fails to generalize to our setting, where the template library is much larger and the symbols are too topologically complex to be reliably memorized by a network. A recent work [Xu et al. 2023a] takes an explicit part library as input but relies on per-shape iterative optimization that does not scale to scene-level decoding. In contrast, TempFit explicitly encodes template geometry as input, supports an extensible vocabulary via dynamic matching, and produces all instances in a single autoregressive pass without test-time optimization.

2.3 Scene CAD Recomposition

Scene CAD recomposition aligns observed objects in 3D scans or RGB images with canonical CAD meshes from external libraries such as ShapeNet [Chang et al. 2015]. This setting is closest to ours, since the template library is large and provided as explicit input. However, existing methods adopt a *detect-then-fit* pipeline.

Representative scan-based methods include Scan2CAD [Avetisyan et al. 2019], which aligns CAD via heatmap correspondences between scan keypoints and CAD voxels; SceneCAD [Avetisyan et al. 2020], which models object-object and object-layout relations with a GNN; and AutoScan2CAD [Li et al. 2023], which combines relation-guided global optimization with active scanning. Image-based methods include Mask2CAD [Kuo et al. 2020], which learns an image-CAD embedding for retrieval after detection; Patch2CAD [Kuo et al. 2021], which uses patch-wise embeddings for part-level matching when exact CAD models are unavailable; ROCA [Gümel et al. 2022], which predicts depth and normalized coordinates for differentiable pose solving; and DiffCAD [Gao et al. 2024], which learns a weakly-supervised diffusion cascade over scale, pose, and shape.

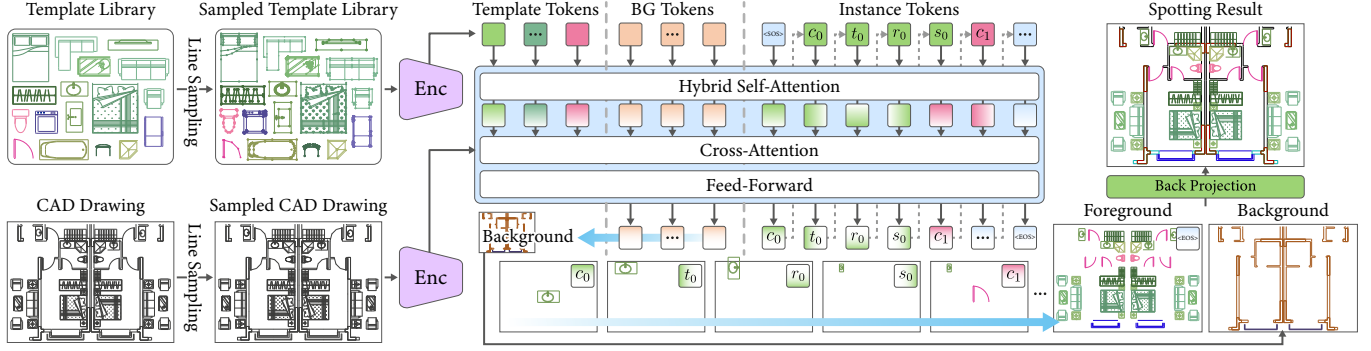


Fig. 2. Overview of TempFit. **Left:** The input drawing and canonical templates are discretized into line segments and encoded by a shared backbone, yielding drawing tokens and template tokens. **Middle:** We prepend the template tokens as an explicit geometric memory and process the scene via Background Segmentation and Foreground Template Fitting: the former predicts per-category masks in parallel via learnable background tokens, while the latter generates instances autoregressively by retrieving a template and predicting its transformation token-by-token. **Right:** The predicted background masks and foreground templates are projected back onto the input primitives to produce the final panoptic result.

The critical weakness of this decoupled design is error accumulation: detection failures—missed objects, misaligned boxes—propagate irreversibly to the fitting stage with no recovery path. The downstream retrieval and pose estimation are upper-bounded by the quality of the initial proposals, and information that would aid detection, such as template-specific geometry, is unavailable to the detector. In contrast, TempFit unifies template retrieval and transformation prediction within a single end-to-end transformer, allowing template geometry to directly inform instance detection and eliminating the intermediate detection bottleneck entirely.

3 Method

Given a CAD drawing represented as an unordered set of graphical primitives $\mathcal{E} = \{e_k\}_{k=1}^K$, the goal of panoptic symbol spotting is to learn a mapping $\mathcal{F} : e_k \mapsto (l_k, z_k)$. Here, $l_k \in \mathcal{L}$ denotes a semantic category, and $z_k \in \mathbb{N}$ is an instance identifier. The category set $\mathcal{L}$ consists of background classes $\mathcal{L}_{bg}$ and foreground classes $\mathcal{L}_{fg}$, which we handle through distinct mechanisms within a unified pipeline, as illustrated in Figure 2.

Unlike prior approaches that adopt a unified segmentation-based paradigm for both foreground and background objects, we handle them separately. For background regions (e.g., walls, floors), we follow the standard segmentation formulation: the model outputs a per-category binary mask over primitives, as in prior work. For foreground instances, we instead adopt a Template Fitting paradigm: each instance is detected by retrieving a canonical template from a predefined library $\mathcal{T}$ and predicting its 2D pose. Section 3.1 formalizes the approach and Section 3.2 details the network architecture.

Finally, we unify both outputs by projecting them back onto the primitive space: background masks directly assign semantic labels to their covered primitives, while each fitted foreground template is instantiated on the canvas and matched to nearby primitives via geometric alignment. This recovers the full panoptic mapping $\mathcal{F} : e_k \mapsto (l_k, z_k)$ for every primitive e_k . The back projection procedure is detailed in Section 3.3.

3.1 Template Fitting

Problem Formulation. We now formalize the template fitting approach described above. The target panoptic output decomposes into two sets $\mathcal{O} = \mathcal{O}_{bg} \cup \mathcal{O}_{fg}$. For background, each category is represented by a binary mask over primitives: $\mathcal{O}_{bg} = \{(c_j, \mathbf{m}_j) \mid c_j \in \mathcal{L}_{bg}\}$, where $\mathbf{m}_j \in \{0, 1\}^K$ indicates which primitives belong to category c_j . For foreground, each instance is parameterized as a retrieved template under a 2D transformation:

$$\mathcal{O}_{fg} = \{(c_i, \mathbf{t}_i, \theta_i, s_i) \mid i = 1, \dots, N, c_i \in \mathcal{T}\}, \quad (1)$$

where c_i indexes a template prototype in $\mathcal{T}$, and $(\mathbf{t}_i, \theta_i, s_i)$ denotes its 2D translation, rotation, and anisotropic scale. We formulate panoptic parsing as a joint conditional generation process by factorizing:

$$P(\mathcal{O} \mid \mathcal{E}, \mathcal{T}) = P(\mathcal{O}_{bg} \mid \mathcal{E}, \mathcal{T}) \cdot \prod_{i=1}^N P(o_i \mid \mathcal{E}, \mathcal{T}, o_{<i}), \quad (2)$$

where the first term predicts background regions and the second term generates foreground instances.

The template library $\mathcal{T}$ can be sourced flexibly depending on the application. In industrial settings, domain experts often maintain 3D asset libraries of the equipment present at a site; templates are then obtained by projecting these canonical models to 2D (as in IndustryCAD, cf. Section 4.1). When no such library is available, it can be automatically constructed by clustering training-set instances (as we do for FloorPlanCAD, cf. Section 4.1).

Framework Design. Inspired by shape abstraction methods that decompose a target into a set of parameterized templates [Ye et al. 2025], we adopt an autoregressive architecture for the foreground term: instances are generated sequentially, each conditioned on previously generated ones. This mirrors the retrieve-then-transform workflow of CAD designers, where each symbol is individually selected and placed.

A key limitation of prior shape abstraction methods is that they memorize template geometry implicitly within network weights, preventing generalization to large or extensible template libraries.

To address this, we encode the template library $\mathcal{T}$ as explicit input, allowing the model to access template-specific geometry through attention rather than abstract memorization. This design draws from prefix-tuning [Li and Liang 2021] in language modeling, with encoded templates replacing learnable virtual prompts.

For background regions, each category requires only a single segmentation mask and no sequential dependency across categories. Autoregressive generation is therefore unnecessary. We instead predict all background masks in parallel *before* the foreground autoregressive generation begins, so the resulting global semantic layout serves as contextual cues for subsequent instance prediction. This informs the model, for instance, where walls and open spaces lie before it places doors and furniture, helping resolve ambiguities in cluttered regions.

These three design choices—autoregressive instance generation, explicit geometric memory, and background-first prediction—are unified through a prefix attention mechanism, detailed in Section 3.2.

3.2 Network Architecture

CAD and Template Encoding. We adopt the line-based representation from VecFormer [Wei et al. 2025] and encode both the input drawing and the template library through a shared pipeline. All graphical primitives are discretized into line segments via uniform sampling, each assigned a geometric feature vector (length, direction, midpoint, and primitive centroid). These line segments, together with their geometric features, pass through a shared PointTransformerV3 [Wu et al. 2024] backbone. For the drawing, line tokens are aggregated into primitive-level tokens $\mathbf{T}_{\text{draw}} \in \mathbb{R}^{K \times d}$ via summed max- and average-pooling. For each template, all line tokens are mean-pooled into a single token, forming the explicit geometric memory $\mathbf{T}_{\text{tpl}} \in \mathbb{R}^{|\mathcal{T}| \times d}$.

Background Segmentation. To estimate the first term in Equation 2, we formulate background segmentation as a parallel detection task. $\mathbf{T}_{\text{bg}} \in \mathbb{R}^{|\mathcal{L}_{\text{bg}}| \times d}$ is a set of learnable embeddings, each fixed to a specific background category, randomly initialized and optimized end-to-end. Unlike category-agnostic DETR queries dynamically assigned to targets via bipartite matching, these tokens preserve their category-specific semantics across drawings. During the forward pass, these tokens interact with the drawing context $\mathbf{T}_{\text{draw}}$ via cross-attention. Let $\mathbf{h}_{\text{bg}}^j \in \mathbb{R}^d$ denote the output token for the j -th background category. Two parallel heads decode it: an existence classifier and a mask prediction head:

$$p_j = \sigma(\text{MLP}_{\text{cls}}(\mathbf{h}_{\text{bg}}^j)), \quad \mathbf{m}_j = \sigma(\text{MLP}_{\text{draw}}(\mathbf{T}_{\text{draw}}) \cdot \text{MLP}_{\text{mask}}(\mathbf{h}_{\text{bg}}^j)^\top), \quad (3)$$

where $\mathbf{m}_j \in [0, 1]^K$ is the per-primitive membership score for category c_j . Together, $\{p_j\}$ and $\{\mathbf{m}_j\}$ capture the scene layout via per-category existence scores and segmentation masks.

Foreground Template Fitting. We realize the second term in Equation 2 as an attribute-wise autoregressive process. As illustrated in Figure 3(a), the input sequence is $[\mathbf{T}_{\text{tpl}}, \mathbf{T}_{\text{bg}}, \mathbf{T}_{\text{inst}}]$, where $\mathbf{T}_{\text{inst}}$ denotes the autoregressively constructed sequence of instance tokens. We enable bidirectional attention within the prefix $[\mathbf{T}_{\text{tpl}}, \mathbf{T}_{\text{bg}}]$ and causal masking on $\mathbf{T}_{\text{inst}}$, while instance tokens attend to the full prefix to condition each prediction on both the geometric library

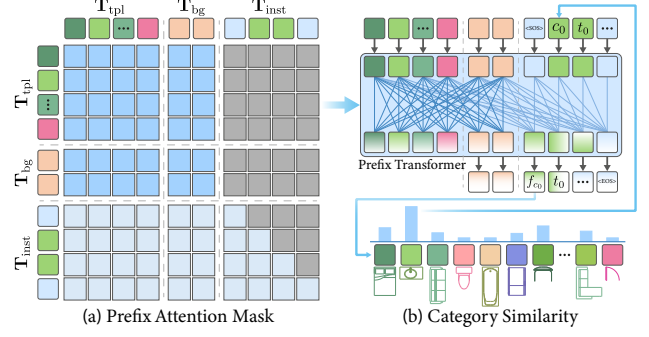


Fig. 3. **(a) Prefix attention mask:** bidirectional attention within $[\mathbf{T}_{\text{tpl}}, \mathbf{T}_{\text{bg}}]$ and causal masking on $\mathbf{T}_{\text{inst}}$ (gray cells block attention). **(b) Class prediction and feature injection:** cosine similarity matching against normalized template tokens, followed by injecting the top-scoring template token into the sequence to condition pose estimation.

and the predicted background layout. Let $\mathcal{H}_i = \{\mathbf{T}_{\text{draw}}, \mathbf{T}_{\text{tpl}}, \mathbf{T}_{\text{bg}}, 0 < i\}$ denote the context history. We decompose:

$$P(o_i | \mathcal{H}_i) = \underbrace{P(c_i | \mathcal{H}_i)}_{\text{Retrieve}} \cdot \underbrace{P(\mathbf{t}_i, \theta_i, \mathbf{s}_i | c_i, \mathcal{H}_i)}_{\text{Transform}}. \quad (4)$$

Retrieve. Instead of a fixed classifier, we perform dynamic metric matching against the template prefix. The normalized output embedding $\mathbf{f}_{\text{cls}}$ is matched via cosine similarity against all normalized template tokens:

$$P(c_i | \mathbf{f}_{\text{cls}}, \mathbf{T}_{\text{tpl}}) = \text{Softmax}(\mathbf{f}_{\text{cls}} \cdot \mathbf{T}_{\text{tpl}}^\top). \quad (5)$$

The category with maximal probability is selected. Crucially, as shown in Figure 3(b), we reuse the selected template token $\mathbf{T}_{\text{tpl}}[c_i]$ to initialize the next attribute input, explicitly conditioning pose estimation on the geometry of the retrieved prototype.

Transform. Conditioned on the injected geometric prior, the model sequentially predicts the transformation attributes $(\mathbf{t}_i, \theta_i, \mathbf{s}_i)$. Each attribute is discretized into K uniform bins over its valid range and predicted as a categorical distribution. Each attribute generation attends to the drawing context, the template prefix, and previously predicted attributes, ensuring geometrically consistent placement.

Loss Function. We train the Transformer by minimizing:

$$\mathcal{L} = \underbrace{\mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{geo}} + \mathcal{L}_{\text{huber}}}_{\text{Foreground Instance Fitting}} + \underbrace{\mathcal{L}_{\text{valid}} + \mathcal{L}_{\text{mask}}}_{\text{Background Segmentation}} + \underbrace{\mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{eos}}}_{\text{Auxiliary}}, \quad (6)$$

where $\mathcal{L}_{\text{cls}}$ is a hierarchy-aware cross-entropy loss for template classification following [Bertinetto et al. 2020], which penalizes intra-category confusions less than inter-category errors; $\mathcal{L}_{\text{geo}}$ supervises discrete transformation tokens $(\mathbf{t}_i, \theta_i, \mathbf{s}_i)$ via cross-entropy; $\mathcal{L}_{\text{huber}}$ applies Huber regression to continuous attributes obtained as expectations over discrete predictions; $\mathcal{L}_{\text{valid}}$ is a BCE loss for background category existence; $\mathcal{L}_{\text{mask}}$ combines BCE and Dice losses for background masks; $\mathcal{L}_{\text{recon}}$ is a Chamfer loss between sampled points of transformed templates and ground-truth instances; and $\mathcal{L}_{\text{eos}}$ supervises the stop-token probability via BCE. All loss terms

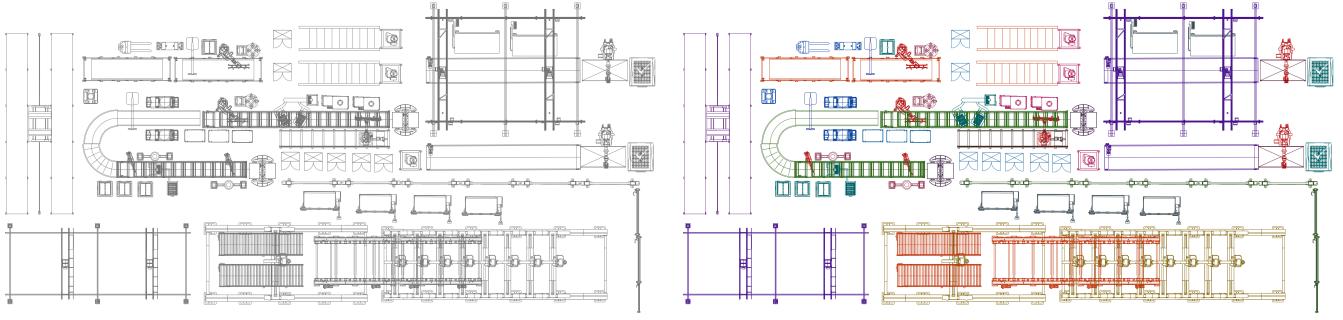


Fig. 4. An annotated example from IndustryCAD. **Left:** The raw factory drawing with text removed. **Right:** Per-primitive ground-truth annotations across 23 equipment categories, each shown in a distinct color.

are summed with equal weights. Detailed formulations are provided in the supplementary material.

3.3 Back Projection

We synthesize the final spotting results by projecting both background and foreground predictions onto the primitive set: we first assign high-confidence primitives in the predicted background masks to their respective background categories, and then solve a robust primitive-to-instance assignment for the remaining primitives.

For each predicted instance o_i , we restrict candidates to a local set $\mathcal{E}_{\text{loc}}$ consisting of unassigned primitives inside its inflated oriented bounding box. For $e_k \in \mathcal{E}_{\text{loc}}$, we define an alignment cost:

$$E(e_k, o_i) = \lambda_{\text{ang}} E_{\text{ang}} + \lambda_{\text{dist}} E_{\text{dist}} + \lambda_{\text{sub}} E_{\text{sub}}, \quad (7)$$

where E_{ang} penalizes orientation mismatch to the nearest instance primitive, E_{dist} measures the perpendicular distance, and E_{sub} penalizes projection overshoot beyond the bounds of o_i to avoid semantic leakage (e.g., assigning a long wall segment to a short window). In overlapping regions, each primitive is assigned to the instance with the minimum cost. Detailed formulations of the three cost terms and their parameter values are provided in the supplementary material.

4 Experiments

We evaluate on FloorPlanCAD and the newly introduced IndustryCAD. We first introduce the datasets (Section 4.1) and describe the experimental setup (Section 4.2), then report results on FloorPlanCAD (Section 4.3) and IndustryCAD (Section 4.4), followed by ablation studies (Section 4.5) verifying our core design choices and the effect of instance ordering, and a robustness analysis (Section 4.6) examining autoregressive error accumulation, template library size, and cross-dataset transfer.

4.1 Datasets

We evaluate on two complementary benchmarks: FloorPlanCAD [Fan et al. 2021] and our newly introduced IndustryCAD, a real-world industrial dataset characterized by high primitive complexity, severe inter-instance occlusion, and extreme scale variation.

FloorPlanCAD. We follow the standard split (6,965 / 810 / 3,827 train/val/test). Since the dataset does not ship with an explicit template library, we construct one from the training set.

We first compute a rotation- and scale-invariant shape-context descriptor [Nguyen et al. 2008] for each of the 103,959 training instances and evaluate pairwise similarity within each category. We set a similarity threshold at the top 1% of the pairwise similarity distribution, so that only the most similar 1% of instance pairs are treated as geometrically equivalent. For each instance, we count how many others exceed this threshold, yielding a repetition count that measures how frequently its geometry recurs in the training set. We then construct the template library by greedy coverage: starting with the highest-count instance, we add it to the library and simultaneously remove it and all instances whose similarity to it exceeds the threshold. We repeat this on the remaining set until at least 95% of all instances have been removed. This procedure yields 462 canonical prototypes, which form the template library used throughout training and inference.

IndustryCAD. Unlike existing CAD benchmarks that focus on residential and commercial buildings, IndustryCAD is collected from real factory scenes across 71 distinct industrial sites. We design 23 equipment categories covering common industrial machinery and adopt the per-primitive annotation protocol of FloorPlanCAD; the dataset contains 34,091 annotated instances in total. One example is shown in Figure 4. Each industrial drawing covers a large physical area (mean 3,779.9 m²), so we partition every drawing into fixed 30 m × 30 m blocks, yielding 2,745 blocks. We randomly split the 71 scenes into 63 for training and 8 for testing (2,213 / 532 blocks and 28,153 / 5,938 instances). The template library consists of 76 prototypes projected from 3D asset libraries at the data source.

IndustryCAD presents three key challenges that distinguish it from existing benchmarks. (a) *Complexity*: industrial equipment is composed of far more geometric primitives than furniture symbols, 2,233 on average per instance, versus 20.3 in FloorPlanCAD, substantially increasing the difficulty of instance grouping and semantic recognition. (b) *Occlusion*: equipment frequently sits inside or around other equipment, with 7.85% of instance pairs having overlapping bounding boxes in IndustryCAD (vs. 0.46% in FloorPlanCAD), making it harder to separate adjacent instances. (c) *Scale*

Table 1. Quantitative comparison on FloorplanCAD test set. The best and second-best results are highlighted by color. “-” denotes entries that cannot be reproduced under our protocol due to missing public implementations. Speed is single-sample inference latency on a single RTX 4090. Under the Hungarian metric, our method achieves the highest PQ on both foreground and background categories.

Methods	#Params	Speed	Greedy Metric			Hungarian Metric		
			PQ	PQ _{fg}	PQ _{bg}	PQ	PQ _{fg}	PQ _{bg}
PanCADNet [Fan et al. 2021]	–	–	59.5	65.6	58.7	–	–	–
CADTransformer [Fan et al. 2022]	–	–	68.9	78.5	58.6	–	–	–
GAT-CADNet [Zheng et al. 2022]	–	–	73.7	–	–	–	–	–
SymPoint [Liu et al. 2024a]	35.1 M	84.7 ms	83.3	84.1	48.2	61.7	64.8	29.0
SymPointV2 [Liu et al. 2024b]	35.1 M	113.3 ms	90.1	90.8	80.8	77.0	78.9	58.0
CADSpotting [Yang et al. 2024]	49.5 M	42.5 ms	88.9	89.7	80.6	–	–	–
VecFormer [Wei et al. 2025]	54.2 M	32.9 ms	91.1	91.8	90.4	48.9	59.4	18.7
PrimitiveAnything [Ye et al. 2025]	104.6 M	201.3 ms	78.3	81.3	75.2	71.4	73.9	61.7
DEIM+ICP [Huang et al. 2025]	62.0 M	65.9 ms	82.0	82.0	–	74.4	74.4	–
Ours	108.3 M	279.0 ms	86.2	86.8	79.6	80.1	81.7	65.4

variation: equipment size varies widely, with the ratio between the 99th- and 10th-percentile instance bounding-box areas reaching 510× (vs. 150× in FloorPlanCAD), so small instances are easily overlooked while large ones contain complex internal structures.

4.2 Experimental Setup

Implementation Details. Our autoregressive transformer uses 6 decoder layers with hidden size $d_{\text{model}} = 768$. The continuous transformation attributes (translation, rotation, and scale) are each uniformly discretized into $K = 128$ equal-width bins over their valid range, and the model predicts a categorical distribution over bins per attribute. We train the model with AdamW (weight decay 0.05) for 500 epochs on 8 RTX-4090 GPUs, using a base learning rate 5×10^{-4} with 5% linear warm-up and cosine decay schedule. We set the per-GPU batch size to 4 with gradient accumulation steps of 4. Ground-truth foreground instances are ordered left-to-right and then top-to-bottom based on their 2D translations.

Baselines. We compare against three families of approaches:

- *Symbol spotting.* Primitive-wise methods (PanCADNet [Fan et al. 2021], CADTransformer [Fan et al. 2022], GAT-CADNet [Zheng et al. 2022]) and instance-wise methods (SymPoint [Liu et al. 2024a], SymPointV2 [Liu et al. 2024b], CADSpotting [Yang et al. 2024], VecFormer [Wei et al. 2025]), all of which formulate spotting as segmentation over input primitives.
- *Shape abstraction.* We reimplement the shape abstraction paradigm by removing the template input branch from our architecture, so that the model decomposes each target into parameterized primitives with implicitly memorized geometry, mirroring the design of PrimitiveAnything [Ye et al. 2025].
- *Scene CAD recomposition.* We reimplement the detect-then-fit paradigm [Avetisyan et al. 2019; Gümeli et al. 2022; Kuo et al. 2020; Li et al. 2023] for our setting using DEIM [Huang et al. 2025] as the representative detector: DEIM detects instances with bounding boxes and predicts semantics, then templates are retrieved and aligned via ICP [Besl and McKay 1992]. Specifically, since DEIM

does not predict the template identity or rotation, we apply ICP to candidate templates within the predicted semantic category using the primitives inside the detected bounding box. The best-aligned candidate is selected as the fitted template, and its rotation is recovered from the ICP alignment.

For both template-fitting baselines, the predicted templates are processed through the same back-projection module as ours.

Evaluation Metrics. Following [Fan et al. 2021], we report Panoptic Quality (PQ) as our primary metric:

$$\text{PQ} = \frac{\sum_{(s_p, s_g) \in \text{TP}} \text{IoU}(s_p, s_g)}{|\text{TP}| + \frac{1}{2}|\text{FP}| + \frac{1}{2}|\text{FN}|},$$

where IoU is length-weighted given the vector nature of primitives.

However, prior works [Liu et al. 2024a; Wei et al. 2025] often adopt a GT-centric greedy matching protocol without a global one-to-one constraint, which allows multiple redundant predictions to be repeatedly matched to the same ground-truth instance and incorrectly counted as TPs, although they should be treated as FPs. This leads to an overestimation of PQ by under-penalizing duplicate predictions. To ensure a fair and standard evaluation, we enforce strict one-to-one matching by employing the Hungarian algorithm to compute the optimal global assignment between the full sets of predictions and ground truths. We therefore additionally report PQ under this strict Hungarian protocol; detailed evaluation procedures (including the TP/FP/FN counting rules and pseudo-code) are provided in the Supplementary Materials.

4.3 Results on FloorPlanCAD

Quantitative Evaluation. Table 1 reports results on FloorPlanCAD with both the Greedy and strict Hungarian protocols, where PQ_{fg} and PQ_{bg} evaluate foreground instances and background regions, respectively. All methods perform worse under Hungarian matching, with a particularly large gap for VecFormer (91.1 → 48.9), confirming that duplicate predictions are strongly penalized under strict one-to-one matching.

Table 2. Quantitative comparison on the IndustryCAD test set. PQ_g uses the GT-centric greedy matching protocol from prior work, while PQ_h enforces a strict global one-to-one assignment via the Hungarian algorithm. The best and second-best results are highlighted by color.

Methods	PQ_g	PQ_h
SymPoint [Liu et al. 2024a]	26.2	20.3
SymPointV2 [Liu et al. 2024b]	34.7	28.5
CADSpotting [Yang et al. 2024]	46.6	38.8
VecFormer [Wei et al. 2025]	66.9	44.4
PrimitiveAnything [Ye et al. 2025]	64.0	53.0
DEIM+ICP [Huang et al. 2025]	67.8	56.6
Ours	73.9	67.0

Under the Hungarian protocol, our method achieves the best overall PQ (80.1), outperforming the previous best SymPointV2 by 3.1 points. The gain is primarily driven by PQ_{fg} (81.7), where our template fitting formulation ensures that every predicted instance is structurally complete by construction, directly eliminating the merge and fragmentation errors that limit segmentation-based methods. For background, our category-specific tokens yield the best PQ_{bg} (65.4), as each token is statically bound to a fixed semantic category, avoiding the assignment ambiguity of DETR-style dynamic queries. Compared with the other template-fitting paradigms, our method outperforms PrimitiveAnything by 8.7 PQ points and DEIM+ICP by 5.7 points. These margins confirm that explicit geometric memory is more effective than implicit template memorization, and that joint retrieve-and-fit avoids the error accumulation of staged detect-then-fit pipelines.

In terms of efficiency, our autoregressive architecture and the inclusion of the template library increase inference latency (279.0 ms vs. 32.9 ms for the fastest method); this overhead remains practical for typical scenes in our target domain.

Qualitative Evaluation. Figure 5 compares our results against representative baselines from two paradigms: Symbol Spotting (SymPointV2, VecFormer) and Template Fitting (Shape Abstraction with PrimitiveAnything, Scene CAD Recomposition with DEIM+ICP).

Symbol Spotting baselines fail at instance grouping. SymPointV2 over-merges spatially distant primitives into a single instance, producing overly large bounding boxes that span across the room, as seen on the left of the first row in Figure 5. The highlighted elongated box around a chair in the same row further illustrates how it incorrectly includes surrounding wall lines in the foreground prediction, while its background predictions drift visibly against the ground truth. In the second row, VecFormer stacks multiple overlapping boxes for the same object and similarly merges distinct objects into a single instance.

The Template Fitting baselines avoid this over-merging by construction: every predicted instance is an instantiation of a valid template, so its constituent primitives automatically form a structurally coherent group. However, both baselines exhibit their own failure modes. PrimitiveAnything memorizes template geometry implicitly within network weights, so its predicted transformation

Table 3. PQ_h on IndustryCAD across inter-instance overlap bins. Overlap degree is the percentage of each instance covered by others; B_1 – B_5 denote $[0, 0.05)$, $[0.05, 0.25)$, $[0.25, 0.50)$, $[0.50, 0.75)$, and $[0.75, 1.00]$.

Methods	Overlap degree				
	B_1	B_2	B_3	B_4	B_5
DEIM+ICP [Huang et al. 2025]	76.4	67.4	55.3	37.8	26.9
Ours	83.4	71.0	59.0	44.7	54.4

parameters often drift, yielding instances that do not tightly cover the underlying primitives, as seen in the highlighted red door in the third row. Scene CAD Recomposition (DEIM+ICP) inherits the limitation of its detection stage: when the detected bounding box is misaligned in position or scale, as in the over-extended box around the green sofa in the fourth row, the subsequent ICP fitting cannot recover, producing misplaced or wrongly-scaled templates.

In contrast, our method produces cleaner and more precise results. By grounding predictions in explicit geometric memory from the template library and unifying retrieval with pose prediction end-to-end, it avoids both the parameter drift of implicit memorization and the cascading errors of staged detection, while maintaining accurate background segmentation that closely matches the ground truth.

4.4 Results on IndustryCAD

Quantitative Evaluation. Table 2 reports the comparison on IndustryCAD. All methods degrade from FloorPlanCAD to IndustryCAD, but the drop differs sharply across paradigms. Symbol Spotting collapses, with SymPointV2 falling from 77.0 to 28.5 PQ under the Hungarian protocol; in contrast, the three Template Fitting baselines (PrimitiveAnything, DEIM+ICP, and Ours) all stay in the 53.0 to 67.0 range, well above any Symbol Spotting method, and our method drops the least ($80.1 \rightarrow 67.0$). This indicates that the Template Fitting formulation is more robust under heavy complexity, occlusion, and scale variation.

Within the Template Fitting paradigm, DEIM+ICP outperforms PrimitiveAnything by 3.0 / 3.6 PQ on the two benchmarks, indicating that explicit template input consistently beats implicit memorization. Our method further outperforms DEIM+ICP by 5.7 / 10.4 PQ, and the gap widens on the harder benchmark, validating our joint retrieve-and-fit over the staged detect-then-fit pipeline.

To further analyze the larger performance gap between TempFit and the DEIM+ICP (detect-then-fit) baseline on IndustryCAD, we compare their performance under different degrees of inter-instance overlap. We define the overlap degree as the percentage of an instance covered by other instances and report PQ using Hungarian matching across five overlap intervals in Table 3. TempFit achieves higher PQ than DEIM+ICP in every interval, with the largest margin in the most heavily overlapped group (54.4 vs. 26.9). The key difference lies in how the fitted templates are obtained. DEIM+ICP aligns candidate templates to the primitives inside each detected bounding box; when instances overlap, the box contains foreign primitives that can corrupt both template selection and pose alignment. TempFit instead predicts the template identity and pose from the full

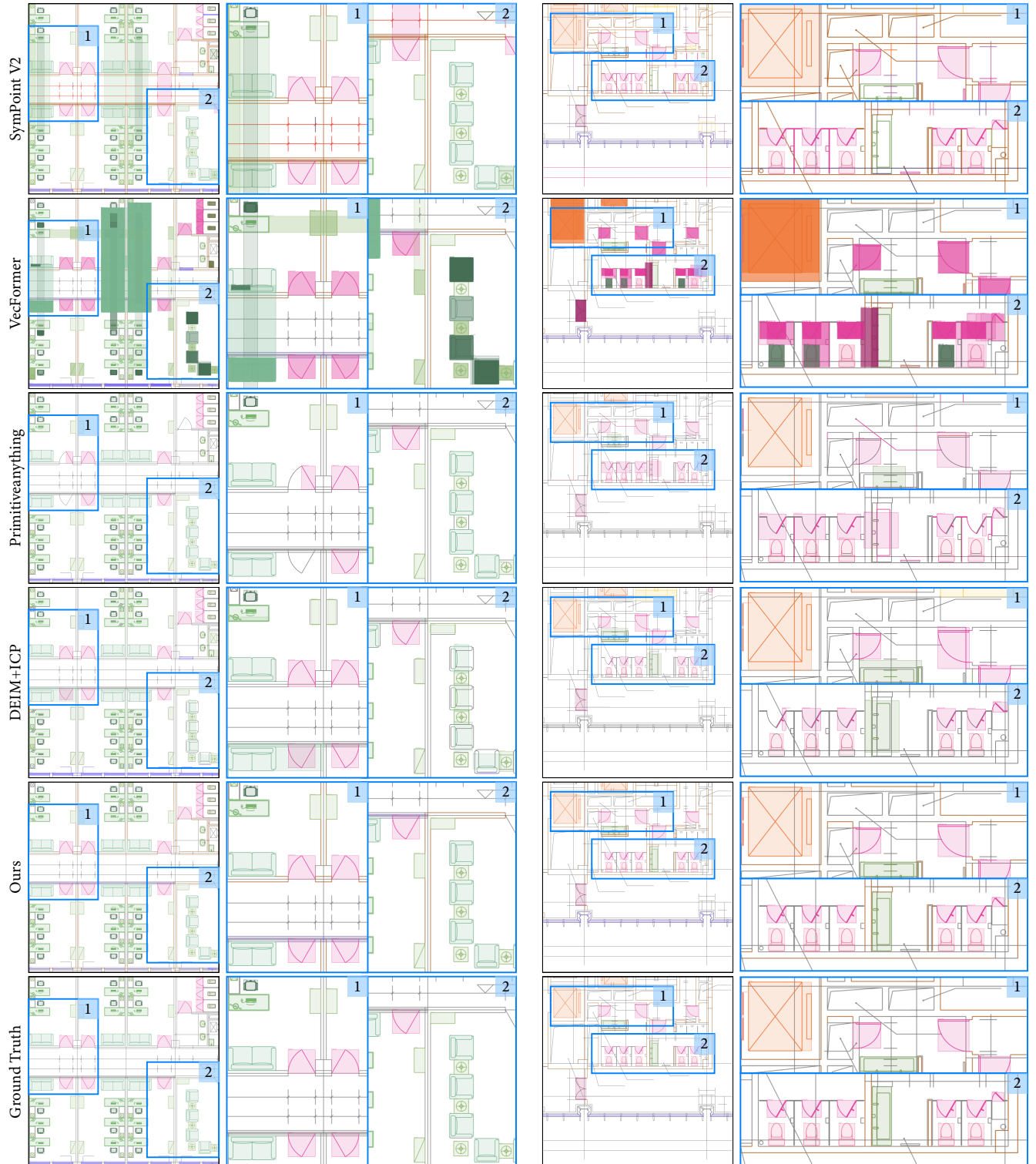


Fig. 5. Qualitative comparison on FloorplanCAD. Colors denote predicted categories, boxes mark instances, and blue insets highlight details.

drawing context without relying on a potentially contaminated local primitive set.

Qualitative Evaluation. Figure 6 compares results against representative baselines on IndustryCAD. The higher per-instance complexity and heavier inter-instance overlap of IndustryCAD amplify the failure modes of every baseline.

Symbol Spotting baselines fail in two ways. First, they struggle to recover complex instances as a whole, often breaking a single instance into several disjoint regions or missing it entirely. Second, they degrade in heavily overlapping regions, where primitives from different instances share the same drawing area. These failures are visible in the first row of Figure 6: the highlighted brown shelf on the left is broken into several disjoint regions, while highlight area on the right reveals both large gray areas where entire instances are missed and overlapping regions where primitives from different instances are not separated.

Template Fitting baselines mitigate the missing-instance problem to some extent, since every prediction is bound to a valid template. However, they still exhibit their own failure modes. PrimitiveAnything, which lacks explicit template input, struggles to fully reconstruct the geometry of complex industrial instances; both the green shelf in the left highlight and the conveyor below the right highlight are only partially recovered. DEIM+ICP, which localizes instances through detection-stage bounding boxes, cannot precisely separate distinct instances in heavily overlapping regions, instead merging multiple instances into a single prediction within the left highlight.

In contrast, our method produces cleaner and more precise results. By grounding predictions in explicit geometric memory and unifying retrieval with pose prediction end-to-end, it better preserves the geometry of each complex symbol and separates distinct instances even in heavily overlapping regions.

Since the 2D templates in IndustryCAD are projected from 3D asset library at the data source, the predicted (template, transformation) pairs can be lifted back to 3D and directly assembled into the underlying 3D industrial scene, as shown in Figure 7.

4.5 Ablation Studies

Table 4 summarizes the ablation results on the FloorPlanCAD test set, including the contributions of the core design choices and the effect of instance ordering.

Effect of Template Priors. To validate the effectiveness of the explicit template memories, we remove the template tokens and employ a standard linear head for direct class prediction. As shown in Row 1 of Table 4, this leads to a significant performance drop, particularly in PQ_{fg} . This confirms that explicitly encoding object templates provides essential geometric guidance, enabling the model to better distinguish symbols in complex drawings.

Effect of Category-Specific Background Tokens. We replace our background tokens with the same number of standard DETR-style queries. These queries are category-agnostic and dynamically assigned to the background-mask targets through bipartite matching. As shown in Row 2, PQ_{bg} drops significantly by 6.1 points, confirming that static binding eliminates the instability inherent in dynamic assignment. Notably, this degradation also negatively impacts PQ_{fg} ,

Table 4. Ablation studies on the FloorPlanCAD test set. The first three variants assess the core design choices, while the Hilbert-curve variant evaluates the effect of instance ordering.

Methods	PQ	PQ_{fg}	PQ_{bg}
w/o Template Memory	71.4	73.9	61.7
DETR-style Background Queries	74.7	76.2	59.3
w/o Robust Assignment	75.9	77.7	63.3
Hilbert-curve Ordering	79.2	80.5	65.1
Ours (Full Model)	80.1	81.7	65.4

suggesting that a stable and precise background layout serves as a critical contextual constraint for foreground object localization.

Effect of Robust Assignment. We replaced our proposed primitive assignment during back projection with a heuristic Euclidean distance strategy (assigning primitives to the spatially closest template). The results in Row 3 demonstrate that spatial proximity alone is insufficient, since nearby primitives often belong to different instances in cluttered regions.

Effect of Instance Ordering. To evaluate the model’s sensitivity to the predefined instance order, we retrain a variant with target sequences ordered by a Hilbert curve computed from the same 2D instance translations, with all other settings unchanged. As shown in Row 4 of Table 4, its PQ decreases only slightly from 80.1 to 79.2. The change mainly occurs in foreground prediction, where PQ_{fg} decreases by 1.2 points, while PQ_{bg} remains nearly unchanged (65.4 vs. 65.1). These results indicate that TempFit is robust to different spatially coherent instance orderings.

4.6 Robustness Analysis

Analysis of Autoregressive Error Accumulation. On the FloorPlanCAD test set, we analyze how an incorrect prediction affects later instances. Predictions are compared with ground truth at the same positions in the predefined spatial order. A prediction is correct if its retrieved template identity matches the corresponding ground truth and their IoU exceeds 0.5. For each sequence with an error, we locate the first incorrect prediction and measure correctness at the next four positions (+1 to +4).

The prediction immediately after the first error is the most affected, with 78.6% correctness. Correctness then increases to 84.3%, 85.0%, and 87.6% at positions +2, +3, and +4, respectively, indicating that the influence of an incorrect prediction diminishes rather than propagating persistently through the sequence. This recovery occurs because, despite attending to previous outputs through self-attention, each decoding step remains grounded in the full encoded drawing through cross-attention.

Robustness to Template Library Size. We analyze the robustness of our method by randomly reducing the size of the template library during inference. As shown in Table 5, the performance degrades gracefully rather than collapsing, even as the library becomes sparse. This demonstrates the model’s strong generalization capability: even

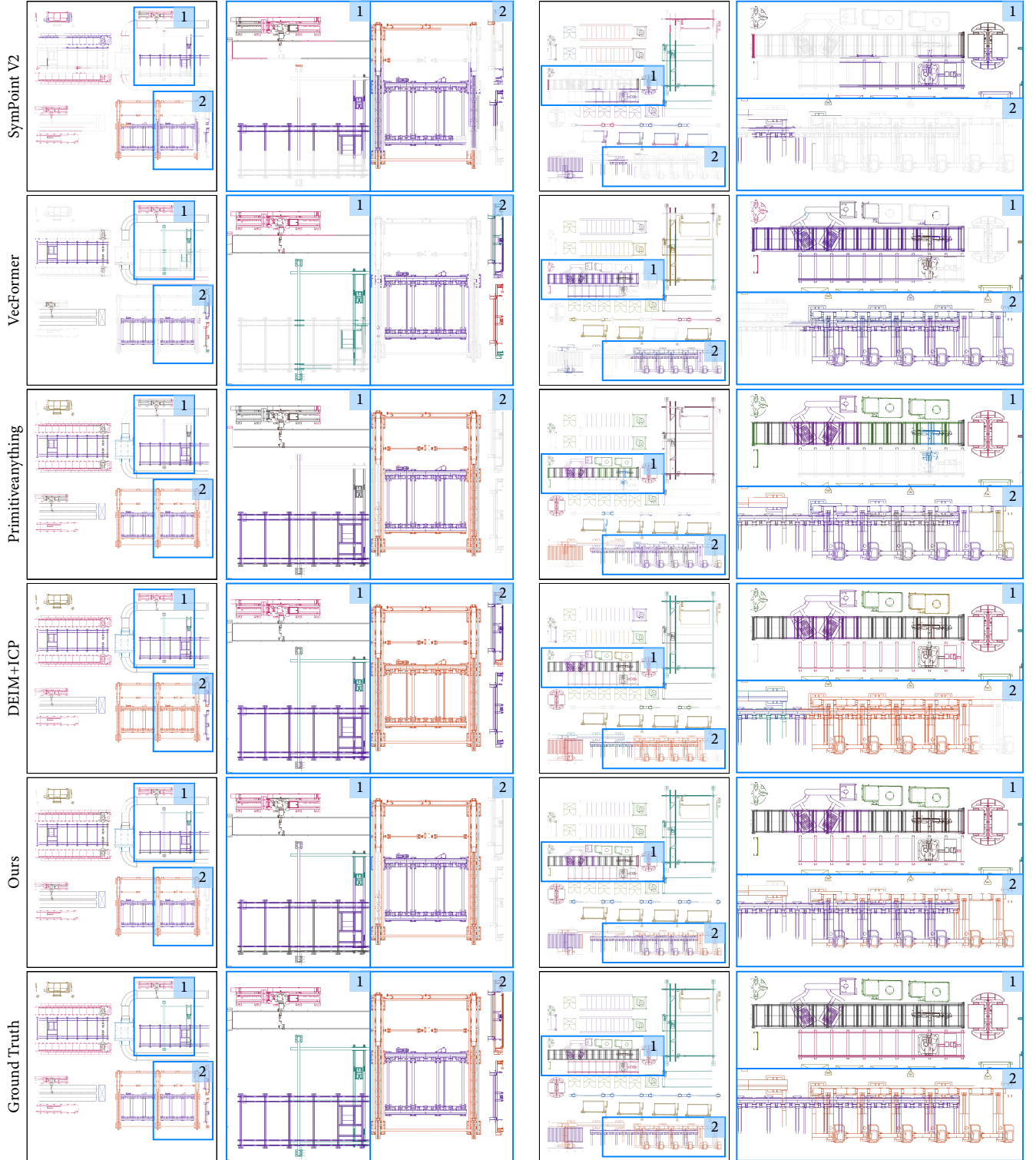


Fig. 6. Qualitative comparison on IndustryCAD. Colors denote predicted categories, and blue insets highlight details.

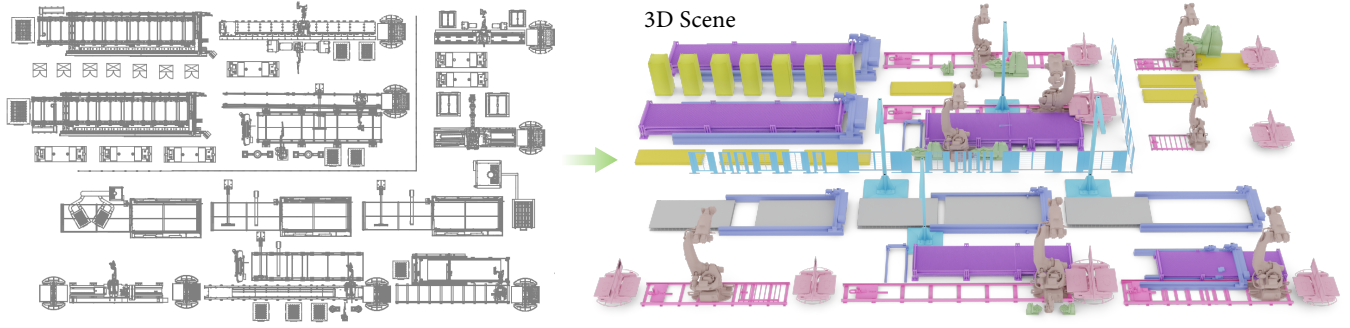


Fig. 7. Predicted (template, transformation) pairs on IndustryCAD can be lifted back to 3D and directly assembled into a 3D industrial scene.

Table 5. Robustness analysis on template library size. We evaluate the robustness of our method by randomly reducing the size of the template library during inference.

Keep Ratio	PQ	PQ _{fg}	PQ _{bg}
25%	65.2	67.2	58.9
50%	72.0	74.3	61.8
75%	76.1	78.4	63.2
100% (Full Model)	80.1	81.7	65.4

when the exact ground-truth template is missing, the model successfully retrieves a semantically similar alternative from the remaining library and transforms it to fit the target instance.

Zero-Shot Cross-Dataset Evaluation. As a stress test, we evaluate our method on SESYD [Delalandre et al. 2010] without fine-tuning. As shown in Table 6, our method achieves PQ 24.67, outperforming the strongest baseline SymPointV2 (16.23) by a large margin, as well as the template-fitting methods DEIM+ICP (11.77) and PrimitiveAnything (9.36), and VecFormer (8.43). This robustness stems from the geometric nature of our retrieve-and-fit mechanism. Rather than directly predicting instance labels from learned features, which are tightly coupled to the training distribution, our model retrieves the most geometrically similar template from the library and adapts it via an explicit 2D transformation. Even when no template exactly matches the target symbol, a sufficiently similar template can be fitted with an appropriate pose, yielding a reasonable result. This geometric grounding makes our method more tolerant to distribution shift than segmentation-based symbol spotting, whose features are distribution-specific. The other template-fitting baselines, despite also fitting templates, are less robust to domain shift for different reasons: PrimitiveAnything memorizes template geometry implicitly in network weights, which remain tied to the training distribution, while DEIM+ICP relies on a detection stage whose learned features equally suffer from domain shift, causing ICP to start from poorly localized boxes.

5 Conclusion

This work presents a new paradigm for Panoptic Symbol Spotting in CAD drawings. We reformulate the task as a *Template Fitting*

Table 6. Zero-shot cross-dataset evaluation on SESYD. Our method outperforms all baselines by a clear margin despite the substantial geometric domain gap from FloorPlanCAD.

Methods	PQ
VecFormer [Wei et al. 2025]	8.43
SymPointV2 [Liu et al. 2024b]	16.23
PrimitiveAnything [Ye et al. 2025]	9.36
DEIM+ICP [Huang et al. 2025]	11.77
Ours	24.67

problem: foreground instances are obtained by retrieving canonical shapes from a template library and aligning them to the drawing, which explicitly enforces structural completeness and geometric integrity. Based on this formulation, we introduce *TempFit*, an autoregressive transformer that treats the template library as explicit geometric memory and unifies template retrieval with pose prediction end-to-end. It parses background regions in parallel via learnable, category-specific tokens to capture the global layout, and synthesizes foreground instances through an autoregressive fitting process that retrieves templates and predicts transformation attributes. Extensive experiments on FloorplanCAD and our newly collected IndustryCAD demonstrate state-of-the-art performance, with particularly large gains on the more challenging IndustryCAD, validating our design choices over both Symbol Spotting and Template Fitting baselines.

Limitations and Future Work. Our method exhibits two failure modes, analyzed quantitatively in Figure 8. First, fitting quality degrades when an instance is geometrically far from every template in the library. As shown in Figure 8(a), the IoU of matched predictions stays roughly stable at small to moderate Chamfer distances but drops sharply once the distance exceeds ~ 0.07 , indicating that instances falling outside the geometric coverage of the library cannot be faithfully recovered. Second, our method can fail on drawings with exceptionally many instances, which are rare in the training data and introduce a substantial scale shift. The failure case in Figure 8(b) involves an unusually long sequence of 80 instances, a sequence length observed in less than 1% of the training data,

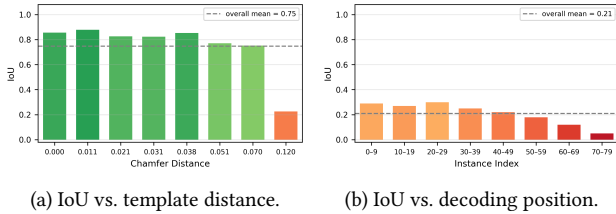


Fig. 8. Failure case analysis. (a) Mean IoU between the predicted and ground-truth instances drops once the Chamfer distance from the ground-truth instance to its nearest template exceeds ~ 0.07 . (b) For a representative drawing with $n_{\text{inst}} = 80$, the bin-averaged IoU between predicted and ground-truth instances decays along the autoregressive decoding position.

whereas more than 80% of the training blocks contain fewer than 25 instances. With so many instances in a single block, each instance appears at a substantially smaller scale than those observed during training, creating an out-of-distribution condition. Under this scale shift, the model already fails at the first decoding step, with an IoU of only ~ 0.28 , followed by progressive degradation along the unusually long autoregressive decoding chain, with IoU falling below 0.05 toward the end of the sequence. The token-by-token generation also limits inference speed on large drawings. Future work includes extending the template library or exploring open-vocabulary template adaptation to handle out-of-library cases, and predicting all attributes of an instance jointly to shorten the decoding chain and accelerate inference.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback and constructive suggestions, which helped improve this work. This work is supported by the National Key R&D Program of China (2024YFB2809102), the National Natural Science Foundation of China (62322207, 62325211, 62132021), the Research Fund of Jiangsu Key Laboratory of AI for Industries (E6420016G8), and Manfred Lau's start-up funds from Lakehead University (Faculty of Science & Environmental Studies).

References

Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X Chang, and Matthias Nießner. 2019. Scan2CAD: Learning CAD Model Alignment in RGB-D Scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2614–2623.

Armen Avetisyan, Tatiana Khanova, Christopher Choy, Denver Dash, Angela Dai, and Matthias Nießner. 2020. SceneCAD: Predicting Object Alignments and Layouts in RGB-D Scans. In *European Conference on Computer Vision*. Springer, 596–612.

Luca Bertinetto, Romain Mueller, Konstantinos Tertikas, Sina Samangooei, and Nicholas A Lord. 2020. Making better mistakes: Leveraging class hierarchies with deep networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 12506–12515.

Paul J. Besl and Neil D. McKay. 1992. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14, 2 (1992), 239–256.

Thomas Binford. 1975. Visual perception by computer. In *Proc. IEEE Conf. on Systems and Control*, 1975.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *European conference on computer vision*. Springer, 213–229.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012* (2015).

Zhiqin Chen, Qimin Chen, Hang Zhou, and Hao Zhang. 2024. DAE-Net: Deforming Auto-Encoder for fine-grained shape co-segmentation. In *ACM SIGGRAPH 2024 Conference Papers*. 1–11.

Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. 2020. Bsp-net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 45–54.

Zhiqin Chen, Kangxue Yin, Matthew Fisher, Siddhartha Chaudhuri, and Hao Zhang. 2019. Bae-net: Branched autoencoder for shape co-segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 8490–8499.

Mathieu Delalandre, Ernest Valveny, Tony Pridmore, and Dimosthenis Karatzas. 2010. Generation of synthetic documents for performance evaluation of symbol recognition & spotting systems. *International Journal on Document Analysis and Recognition (IJDAR)* 13, 3 (2010), 187–207.

Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. 2020. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 31–44.

Alexey Dosovitskiy. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

Zhiwen Fan, Tianlong Chen, Peihao Wang, and Zhangyang Wang. 2022. Cadtransformer: Panoptic symbol spotting transformer for cad drawings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10986–10996.

Zhiwen Fan, Lingjie Zhu, Honghua Li, Xiaohao Chen, Siyu Zhu, and Ping Tan. 2021. FloorplanCAD: A large-scale cad drawing dataset for panoptic symbol spotting. In *Proceedings of the IEEE/CVF international conference on computer vision*. 10128–10137.

Matheus Gadelha, Giorgio Gori, Duygu Ceylan, Radomir Mech, Nathan Carr, Tamy Boubekeur, Rui Wang, and Subhransu Maji. 2020. Learning generative models of shape handles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 402–411.

Daoyi Gao, Dávid Rozenberszki, Stefan Leutenegger, and Angela Dai. 2024. DiffCAD: Weakly-Supervised Probabilistic CAD Model Retrieval and Alignment from an RGB Image. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–15.

Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. 2020. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4857–4866.

Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. 2019. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF international conference on computer vision*. 7154–7164.

Can Güneli, Angela Dai, and Matthias Nießner. 2022. ROCA: Robust CAD Model Retrieval and Alignment from a Single Image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4022–4031.

Abhinav Gupta, Alexei A Efros, and Martial Hebert. 2010. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *European Conference on Computer Vision*. Springer, 482–496.

Shihua Huang, Zhichao Lu, Xiaodong Cun, Yongjun Yu, Xiao Zhou, and Xi Shen. 2025. DEIM: DETR with Improved Matching for Fast Convergence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Xiaoyang Huang, Yi Zhang, Kai Chen, Teng Li, Wenjun Zhang, and Bingbing Ni. 2023. Learning shape primitives via implicit convexity regularization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3642–3651.

Bo Jiang, Ziyang Zhang, Doudou Lin, Jin Tang, and Bin Luo. 2019. Semi-supervised learning with graph learning-convolutional networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11313–11320.

R Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang, Paul Guerrero, Niloy J Mitra, and Daniel Ritchie. 2020. Shapeassembly: Learning to generate programs for 3d shape structure synthesis. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–20.

Weicheng Kuo, Anelia Angelova, Tsung-Yi Lin, and Angela Dai. 2020. Mask2CAD: 3D Shape Prediction by Learning to Segment and Retrieve. In *European Conference on Computer Vision*. Springer, 260–277.

Weicheng Kuo, Anelia Angelova, Tsung-Yi Lin, and Angela Dai. 2021. Patch2CAD: Patchwise Embedding Learning for In-the-Wild Shape Retrieval from a Single Image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12589–12599.

Baiqiang Leng, Jingwei Huang, Guanlin Shen, and Bin Wang. 2024. Shape Embedding and Retrieval in Multi-Flow Deformation. *Computational Visual Media* 10, 3 (2024), 439–451. doi:10.1007/s41095-022-0315-3

Changhao Li, Junfu Guo, Ruizhen Hu, and Ligang Liu. 2023. Online Scene CAD Reconstruction via Autonomous Scanning. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–16.

Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. 2017. Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–14.

Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J Guibas. 2019. Supervised Fitting of Geometric Primitives to 3D Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2652–2660.

- Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 4582–4597.
- Weixiao Liu, Yuwei Wu, Sipu Ruan, and Gregory S Chirikjian. 2023. Marching-primitives: Shape abstraction from signed distance function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8771–8780.
- Wenlong Liu, Tianyu Yang, Yuhao Wang, Qizhi Yu, and Lei Zhang. 2024a. Symbol as points: Panoptic symbol spotting via point-based representation. *arXiv preprint arXiv:2401.10556* (2024).
- Wenlong Liu, Tianyu Yang, Qizhi Yu, and Lei Zhang. 2024b. Sympoint revolutionized: Boosting panoptic symbol spotting with layer feature enhancement. *arXiv preprint arXiv:2407.01928* (2024).
- Ruifeng Luo, Zhengjie Liu, Tianxiao Cheng, Jie Wang, Tongjie Wang, Xingguang Wei, Haomin Wang, Yanpeng Li, Fu Chai, Fei Cheng, Shenglong Ye, Wenhao Wang, Yanting Zhang, Yu Qiao, Hongjie Zhang, and Xianzhong Zhao. 2025. ArchCAD-400K: A Large-Scale CAD Drawings Dataset and New Baseline for Panoptic Symbol Spotting. In *Advances in Neural Information Processing Systems*, Vol. 38. Curran Associates, Inc., 127715–127739. doi:10.52202/085713-4253
- Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas J Guibas. 2019. Structrenet: Hierarchical graph networks for 3d shape generation. *arXiv preprint arXiv:1908.00575* (2019).
- Tom Monnier, Jake Austin, Angjoo Kanazawa, Alexei Efros, and Mathieu Aubry. 2023. Differentiable blocks world: Qualitative 3d decomposition by rendering primitives. *Advances in Neural Information Processing Systems* 36 (2023), 5791–5807.
- Thi-Oanh Nguyen, Salvatore Tabbone, and O Ramos Terrades. 2008. Symbol descriptor based on shape context and vector model of information retrieval. In *2008 The Eighth IAPR International Workshop on Document Analysis Systems*. IEEE, 191–197.
- Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. 2019. Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10344–10353.
- Alex Pentland. 1986. Parts: Structured Descriptions of Shape.. In *AAAI*. 695–701.
- Ioannis Pitas and Anastasios N. Venetsanopoulos. 2002. Morphological shape decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12, 1 (2002), 38–45.
- Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems* 28 (2015).
- Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomir Měch. 2020. ParSeNet: A Parametric Surface Fitting Network for 3D Point Clouds. In *European Conference on Computer Vision*. Springer, 261–276.
- Chun-Yu Sun, Qian-Fang Zou, Xin Tong, and Yang Liu. 2019. Learning adaptive hierarchical cuboid abstractions of 3d shape collections. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–13.
- Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. 2017. Learning shape abstractions by assembling volumetric primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2635–2643.
- Xingguang Wei, Haomin Wang, Shenglong Ye, Ruifeng Luo, Yanting Zhang, Lixin Gu, Jifeng Dai, Yu Qiao, Wenhao Wang, and Hongjie Zhang. 2025. Point or Line? Using Line-based Representation for Panoptic Symbol Spotting in CAD Drawings. *arXiv preprint arXiv:2505.23395* (2025).
- Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. 2024. Point transformer v3: Simpler faster stronger. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4840–4851.
- Qun-Ce Xu, Tai-Jiang Mu, and Yong-Liang Yang. 2023b. A Survey of Deep Learning-Based 3D Shape Generation. *Computational Visual Media* 9, 3 (2023), 407–442. doi:10.1007/s41095-022-0321-5
- Xianghao Xu, Paul Guerrero, Matthew Fisher, Siddhartha Chaudhuri, and Daniel Ritchie. 2023a. Unsupervised 3D Shape Reconstruction by Part Retrieval and Assembly. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8559–8568.
- Fuyi Yang, Jiazuo Mu, Yanshun Zhang, Mingqian Zhang, Junxiong Zhang, Yongjian Luo, Lan Xu, Jingyi Yu, Yujiao Shi, and Yingliang Zhang. 2024. Cadspotting: Robust panoptic symbol spotting on large-scale cad drawings. *arXiv preprint arXiv:2412.07377* (2024).
- Jingwen Ye, Yuze He, Yanning Zhou, Yiqin Zhu, Kaiwen Xiao, Yong-Jin Liu, Wei Yang, and Xiao Han. 2025. PrimitiveAnything: Human-crafted 3D primitive assembly generation with autoregressive Transformer. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Papers*. 1–12.
- Zhaohua Zheng, Jianfang Li, Lingjie Zhu, Honghua Li, Frank Petzold, and Ping Tan. 2022. GAT-CADNet: Graph attention network for panoptic symbol spotting in CAD drawings. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11747–11756.