2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

PIRATR: Parametric Object Inference for Robotic Applications with Transformers in 3D Point Clouds

Michael Schwingshackl*, Fabio F. Oberweger*, Mario Niedermeyer, Huemer Johannes, Markus Murschitz
AIT Austrian Institute of Technology
Center for Vision, Automation & Control

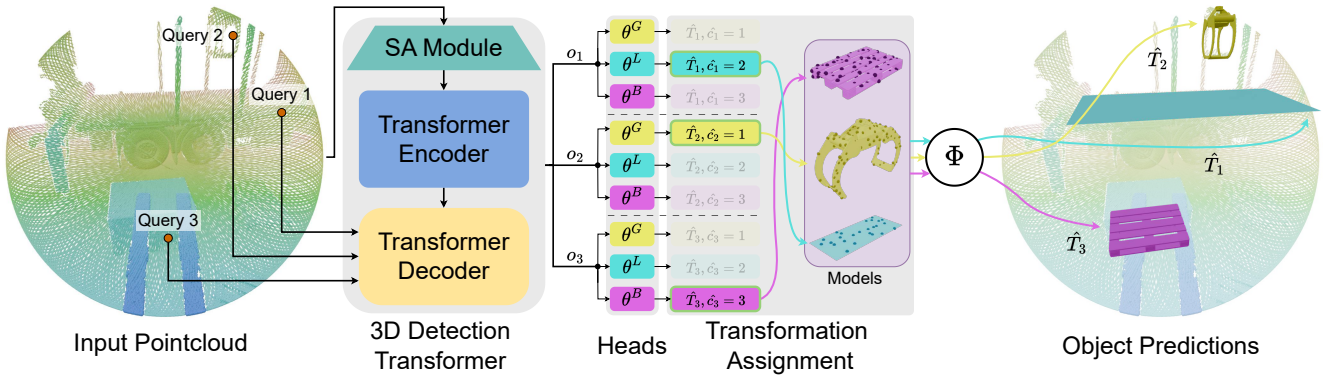


Fig. 1: PIRATR is an end-to-end trainable model that takes a point cloud as input, applies farthest point sampling to generate point queries, and encodes them into output embeddings using 3DETR [1]. Class-specific heads predict parametric objects corresponding to the gripper, loading platform, and pallet. During training, a geometry-aware matcher [2] learns to associate embeddings with the correct classes, and at inference, the model directly outputs class-specific configurations, which are applied to predefined meshes to generate the final predictions.

***Abstract*— We present PIRATR, an end-to-end 3D object detection framework for robotic use cases in point clouds. Extending PI3DETR, our method streamlines parametric 3D object detection by jointly estimating multi-class 6-DoF poses and class-specific parametric attributes directly from occlusion-affected point cloud data. This formulation enables not only geometric localization but also the estimation of task-relevant properties for parametric objects, such as a gripper's opening, where the 3D model is adjusted according to simple, predefined rules. The architecture employs modular, class-specific heads, making it straightforward to extend to novel object types without re-designing the pipeline. We validate PIRATR on an automated forklift platform, focusing on three structurally and functionally diverse categories: crane grippers, loading platforms, and pallets. Trained entirely in a synthetic environment, PIRATR generalizes effectively to real outdoor LiDAR scans, achieving a detection mAP of 0.919 without additional fine-tuning. PIRATR establishes a new paradigm of pose-aware, parameterized perception. This bridges the gap between low-level geometric reasoning and actionable world models, paving the way for scalable, simulation-trained perception systems that can be deployed in dynamic robotic environments. Code: https://github.com/swingaxe/piratr**

I. Introduction

Skilled labor shortages drive the recent automation efforts of working machines such as forklifts, cranes, and wheel loaders. Therefore, autonomous robotic machines are following the footsteps of autonomous vehicles into the outdoors. While autonomous vehicles are usually solely focusing on navigating in their environment, autonomous machines i.e., robots interact with their environment to fulfill their purpose. In practical applications such as automated forklifts, interactions are typically limited to structured objects (e.g., pallets) or other machines (e.g., loading platforms) and require reliable 6-DoF pose estimation. Some objects, like crane grippers, can change their state (e.g., opening and closing), so learning a parametric representation is crucial for understanding and interacting with the environment. This data modality, using LiDAR is exemplified in Fig. 3. LiDAR has shown to be a robust sensor modality yielding accurate 3D information even at high distances that can avoid some of the typical problems of vision-based systems, especially their dependence on lighting conditions. While there exist well-established machine learning systems for multi-class object detection and pose estimation in images and RGB-D data, point cloud based detectors are mostly composed of different computation blocks, often containing non-differentiable classical shape fitting or prepossessing methods. Also, many require an additional dense intermediate representation (such as bird's eye view or voxels).

To this end, this work's approach is inspired by the end-

*Equal contribution.

to-end transformer model for 3D edge detection presented in PI3DETR [2], which extends 3DETR [1] beyond bounding boxes to parametric 3D curves but works only on nearly complete point clouds, which are the result of a multi-view scanning system [3], [4]. In contrast, our work accounts for incomplete point clouds caused by occlusions and the characteristics of LiDAR systems and, for the first time, successfully applies this approach in the context of contact interaction scenarios of automated machines. Our method operates directly on 3D point clouds, including occlusion artifacts as commonly observed in off-the-shelf LiDAR scans. Owing to its lean end-to-end design, the framework can be readily extended beyond 6-DoF pose estimation to incorporate object-specific state parameters. We demonstrate this capability through gripper detection and localization with joint estimation of the opening angle. PIRATR is trained exclusively on synthetic data and transfers its detection capabilities to real-world point clouds. Consequently, adapting PIRATR to new applications only requires CAD models of the target objects within the synthetic data generation pipeline. Our contributions presented in this work are:

- PIRATR, a generic, fully differentiable end-to-end network for 3D multi-class, parametric multi-object detection and 6-DoF object pose estimation, without intermediate representations (Sec. III-B) that operates on LiDAR data containing occlusion artifacts.
- A synthetic data generation approach used to train PIRATR that simulates occlusion and other sensor-specific properties and artifacts (Sec. III-A).
- Integration of additional (beyond 6-DoF) object state estimation by adding a class-specific feed-forward network per state parameter, which is demonstrated by gripper opening angle estimation (Sec. III-B.1).
- Thorough analysis of the system's capabilities on synthetic and annotated real-world datasets (Sec. IV).
- PIRATR deployment on a forklift to perform fully autonomous loading tasks (see Fig. 2).

II. Related Work

Object detection in 3D point clouds has been widely studied in autonomous driving and mobile robotics. Deep learning methods mainly follow three paradigms: voxel, point, and bird's-eye view (BEV). Voxel methods discretize the point cloud into 3D grids for convolutional feature extraction [5], but suffer from quantization artifacts and high computational cost on dense LiDAR data. Point-based methods such as PointNet [6] and its variants [7], [8] operate directly on raw points and learn permutation-invariant features, capturing fine local geometry but scaling poorly in large outdoor scenes. BEV approaches [9], [10] instead project point clouds onto a top-down grid. PointPillars [11] is a notable example encoding vertical columns into pseudo images for efficient 2D convolution, though discretization again causes information loss along the z-axis.

Most existing methods predict coarse 3D bounding boxes. Parametric and primitive-based alternatives instead aim for more detailed representations. Classical approaches use geometric fitting such as RANSAC planes [12], cylinder fitting [13], or primitive decomposition [14], [15], which yield compact and interpretable models but are sensitive to noise, clutter, and missing data. Recent learning-based methods overcome these issues by directly predicting surfaces, edges, or primitives, as in SED-Net [16], feature-preserving reconstruction [17], or transformer-based sketch-to-primitive models like Point2Primitive [18]. Other works estimate parametric objects that follow predefined rules, including extrusion cylinders [19], building wireframes [20], or CAD models [18], [21].

PI3DETR [2] combines point-based feature extraction with a 3D detection transformer [1], [22] architecture to directly predict 3D parametric edges using curve primitives such as lines, Bézier curves, arcs, and circles.

Recent pallet detection methods rely on adaptive Gaussian point feature histograms [23], although most approaches operate in 2D or use RGB-D data [24], [25]. Loading platforms or flatbeds are typically represented as planar surfaces, and plane fitting techniques such as RANSAC or robust PCA are commonly employed [25]. Other approaches leverage flatbed edge segmentation to estimate usable surface regions [26].

Voxel-, point-, and BEV-based methods provide robust 3D detection but lack the geometric detail needed for precise manipulation. Parametric and primitive-based approaches yield interpretable representations yet remain limited to simple shapes or narrow tasks. Our work extends the PI3DETR architecture to 3D object detection for robotic outdoor manipulation scenarios. Instead of predicting parametric curves, we predict the 6-DoF pose and individual parameters of pallets, loading platforms, and crane grippers using an end-to-end model, enabling direct deployment in robotic applications.

Fig. 2: Image of the autonomous forklift operating in an outdoor environment, where our parametric object detection method is tested and deployed. The main detection targets such as the gripper, loading platform and pallets are visible to provide understanding of the perception task and context.

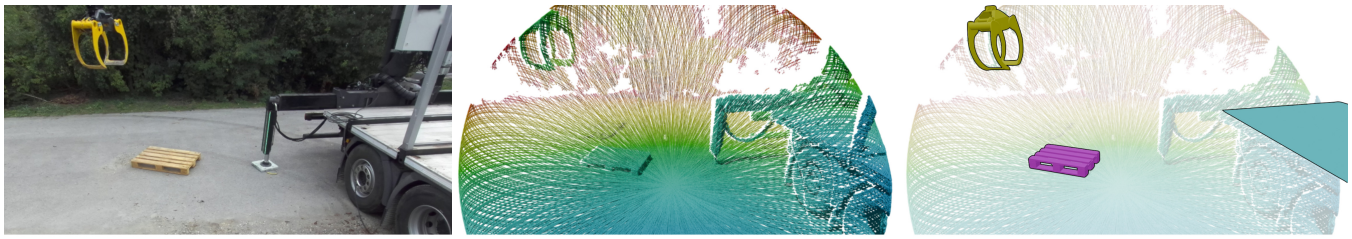


Fig. 3: **Left**: reference image from the forklift-mounted camera. **Center**: input point cloud captured with a Livox Mid70 LiDAR. **Right**: 3D annotations of the supervision targets: gripper, loading platform, and pallets.

III. Method

A. Synthetic Data Generation

To avoid the cost and effort of collecting annotated data with heavy machinery, we generate synthetic training samples in Blender. This enables rapid creation of diverse scenarios without manual intervention. In total, 5k samples were produced and split into train (90%), test (10%), and validation (10%) sets.

1) Scene generation: Closing the sim-to-real gap is challenging because simulation hardly reproduces the fine-grained, high-entropy complexity of the real world. To mitigate this, we use accurate CAD models of the forklift and truck-mounted loading crane (omitting only small details such as cables and pistons), as visible in Fig. 4a, combined with extensive domain randomization. The truck-mounted loading crane aligned at the global scene origin. Its gripper, the loading platform, and surrounding pallets form the core of our supervision setup, as their poses (and the gripper's opening state) are used as training labels following the conventions in Sec. III-A.3. To randomize scenes, we sample object placements from carefully chosen ranges that reflect realistic configurations. First, the forklift is placed at random positions along a circle of radius 5–16 m around the crane, oriented either toward one of eight predefined points of interest on the truck or away from it, which encourages the model to not assume the truck is always visible. The simulated LiDAR (Sec. III-A.2) is mounted on the forklift's mast, as visible in Figure 4a, so sensor height and tilt vary naturally with mast motion. Similar to the forklift, the crane's gripper is randomly sampled along a circle of radius 3.5–8 m centered at the backside of the truck, with randomized height (0.5–4.5 m), orientation, and opening angle α. Realistic distribution of pallets is achieved using Poisson-disk sampling modulated with fractal Perlin noise to create clusters, while preventing overlaps by keeping proximity to the truck, gripper, and forklift larger than 1.5m. Pallets may contain either boxes or other pallets, allowing stacks up to two objects high. Since this forklift is intended to work in outdoor terrain, different-sized trees and bushes are scattered with Poisson-disk sampling under a minimum spacing of 1 m. To simulate urban structure, flat wall meshes are randomly placed around the scene. Fig. 4b illustrates an example scene, including the simulated point cloud sampled from the forklift perspective.

2) LiDAR simulation: In deep learning with point cloud data, the spatial distribution of points plays a crucial role for both learning stability and downstream performance [27]. To ensure realistic input, we replicate the LiDAR scan pattern as closely as possible. Unlike conventional spinning LiDARs, Livox devices employ a non-repetitive scanning mechanism that produces trajectories similar to Lissajous curves, which gradually yield uniform coverage over time. To simulate this behavior, we rely on the official Livox-SDK simulation data[1], which provides sequential text files containing timestamp, azimuth, and elevation values for over 400k rays. The direction vectors for each ray are obtained via a spherical-to-Cartesian coordinate transformation and are subsequently used as inputs to the Raycast node in Blender Geometry Nodes. For each timestamp, the node performs a hit test on the scene geometry from the sensor origin along every ray vector, and the intersection points are collected to form a simulated point cloud P. The maximum hit distance is set to 25m. P is subsequently reduced to 32k points using farthest point sampling. Targets with LiDAR hit counts below a class-specific threshold are excluded from the ground-truth annotations due to occlusions.

3) Parametrization and ground truth definition: Following [2], we define ground-truth annotations as follows. Let P denote an input point cloud containing M parametric objects. Each object instance is indexed by $i \in \{1, \dots, M\}$ and assigned a class label $c_i \in C = \{1, 2, 3\}$, where 1, 2, and 3 correspond to *grippers*, *loading platforms*, and *pallets*, respectively. The ground-truth targets are parameterized as

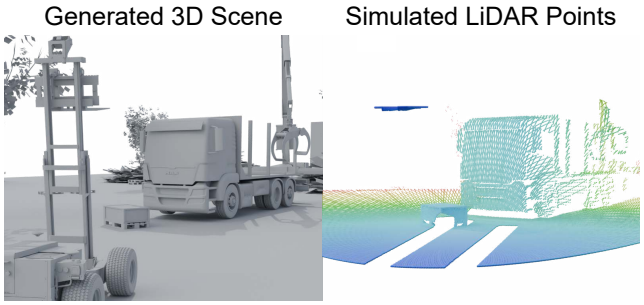
$$T_i = \begin{cases} [\mathbf{p}_i, \mathbf{q}_i, \alpha_i], & c_i = 1, \\ [\mathbf{p}_i, \mathbf{q}_i], & c_i \in \{2, 3\}, \end{cases} \tag{1}$$

where $\mathbf{p}_i \in \mathbb{R}^3$ denotes the object position, $\mathbf{q}_i \in \mathbb{R}^4$ a unit quaternion encoding its orientation, and $\alpha_i \in \mathbb{R}^+$ the opening angle of the gripper.

The interpretation of $\mathbf{p}_i$ is class-dependent. For grippers ($c_i = 1$), we use the center of mass, as they can translate and rotate freely. For loading platforms ($c_i = 2$), we select the front-right corner (facing the driver cabin), since the front/back distinction is semantically relevant and this corner is geometrically well-defined in the point cloud. For pallets

[1] https://github.com/Livox-SDK

(a) CAD models of the truck-mounted loading crane, forklift, and pallets. Supervision targets: gripper, loading platform, and pallets. Target reference coordinate frames are visualized as well as gripper opening value α. The simulated LiDAR sensor is mounted on the forklift mast, matching the real-world configuration.



(b) **Left**: Synthetic training scene generated in Blender with a forklift, truck-mounted crane, pallets, and surrounding clutter/vegetation. **Right**: Simulated Livox Mid-70 point cloud of the same scene.

Fig. 4: (a) CAD models and supervision targets. (b) Synthetic training scene and corresponding simulated point cloud.

($c_i = 3$), we use the bottom center point, reflecting the fact that pallet rotations are constrained by the supporting surface. Moreover, both grippers and pallets are treated as invariant under a 180° rotation around the vertical z-axis.

As meshes are available for all object classes, explicit dimension regression is not required. Each class mesh $\mathcal{M}^c$, with $c \in C$, can be instantiated in the scene using parameters T_i, which specify the object's pose and, in the case of grippers, its state parameter (opening angle). To obtain a compact point-based representation, we associate each mesh with a predefined set of 64 surface points. We formalize this process with a class-conditional mapping

$$\Phi : (\mathcal{M}^{c_i}, T_i) \mapsto \mathbb{R}^{64\times 3}, \tag{2}$$

which applies the configuration T_i to the mesh $\mathcal{M}^{c_i}$ of instance i and outputs its corresponding 64-point representation. For brevity, we also use Φ to denote the mapping that applies the configuration to the mesh alone, without extracting the point-based representation.

Finally, all point clouds, meshes, and position parameters are normalized with respect to the longest axis of the input cloud, ensuring that all points lie within $[-1, 1]$. The gripper opening angle α is likewise scaled to the range $[-1, 1]$ for consistency with the other parameters. Orientations are uniformly represented using unit quaternions.

B. PIRATR Overview

Our method builds upon PI3DETR [2], which in turn extends the end-to-end 3D object detection framework of [1]. Our method enhances these foundations by enabling the detection of differently parameterized objects, even under occlusion, for robotic applications in point clouds. Given the strong empirical performance of PI3DETR in its original domain, we directly adopt its model core.

In brief, an input point cloud P is first processed by a set-aggregation module (SAModule) [7], which applies farthest point sampling to down-sample the point cloud and extract d-dimensional local features. These features are passed to a transformer that operates on K non-parametric query embeddings. The queries are initialized from farthest point sampled locations $\{q_j\}_{j=1}^K$ and encoded via sinusoidal positional embeddings [28], yielding a set of output embeddings $\{o_j\}_{j=1}^K$.

Our contribution begins at this stage. Rather than revisiting the core architecture, we refer readers to [2] for a comprehensive description. Here, we focus on the novel extensions introduced by PIRATR to adapt PI3DETR for robotic manipulation scenarios.

1) Prediction feed-forward networks (FFN): Following [2], we define separate prediction heads for each class of parameterized object. Each head is implemented as a feed-forward network (FFN) with parameters $\theta^{(\cdot)}$ and takes as input an output embedding o_j from the model core. For grippers, θ^{G} maps o_j to $\hat{T}_j = [\hat{\mathbf{p}}_j, \hat{\mathbf{q}}_j, \hat{\alpha}_j]$, where $\hat{\mathbf{p}}_j \in \mathbb{R}^3$ is the predicted position, $\hat{\mathbf{q}}_j \in \mathbb{R}^4$ a unit quaternion, and $\hat{\alpha}_j \in \mathbb{R}$ the gripper opening angle. Unlike [2], we do not employ a separate scalar head. Instead, the FFN jointly predicts all parameters. The same structure is used for loading platforms (θ^L) and pallets (θ^B), with outputs $\hat{T}_j = [\hat{\mathbf{p}}_j, \hat{\mathbf{q}}_j]$. Here, the semantic meaning of $\hat{\mathbf{p}}_j$ differs by class, as defined in Sec. III-A.3. This class-dependent, multi-head setup follows [2] and is motivated by the differing positional semantics and geometric structures across object categories. For clarity of presentation, $\hat{\mathbf{p}}_j$ is described as a directly predicted position. In practice, it is predicted as an offset from the corresponding query point $\mathbf{q}_j$, such that $\hat{\mathbf{p}}_j = \hat{\Delta}_j + \mathbf{q}_j$. Finally, a classification head θ^{cls} maps o_j to $\hat{\pi}_j \in [0, 1]^4$, representing class probabilities over {*no-object*, *gripper*, *loading platform*, *pallet*}, which determine the object type at inference. As in [2], PIRATR predicts parameters for all classes per query, while the geometry-aware matcher ensures that gradients are assigned only to the ground-truth head during training.

2) Geometry-aware matching: We introduce a geometry-aware matching strategy that accounts for object symmetries when aligning predictions to ground truth. Let $\mathcal{L}^{\mathcal{S}}_{\mathrm{quat}}$ denote the unit quaternion symmetry loss with respect to a symmetry

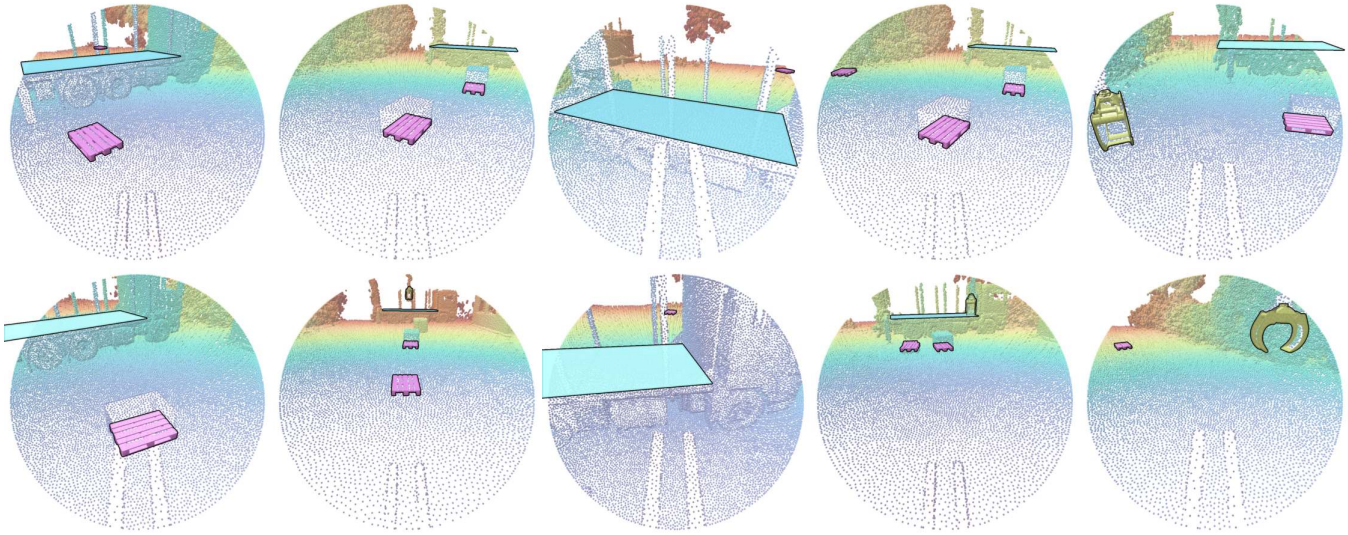


Fig. 5: Qualitative synthetic-to-real prediction of PIRATR, which is trained solely on synthetic data and evaluated on real scans. Predicted classes: **gripper** (yellow), **loading platforms** (cyan), and **pallets** (magenta).

set $\mathcal{S}$, defined as

$$\mathcal{L}_{\text{quat}}^{\mathcal{S}}(\hat{\mathbf{q}}, \mathbf{q}) = \min_{\mathbf{q}' \in \mathcal{E}_{\mathcal{S}}(\hat{\mathbf{q}})} \ell_1(\mathbf{q}', \mathbf{q}), \tag{3}$$

where $\mathcal{E}_{\mathcal{S}}(\mathbf{q})$ denotes the set of quaternions obtained by applying the symmetries in $\mathcal{S}$ to $\mathbf{q}$. For instance, let $r^z(\mathbf{q}) = \mathbf{q} \otimes [0, 0, 0, 1]$ denote a $180°$ rotation around the z-axis. Then, under the symmetries $\{\pm 1, \pm r^z\}$, we obtain

$$\mathcal{E}_{\{\pm 1, \pm r^z\}}(\mathbf{q}) = \{\mathbf{q},\ -\mathbf{q},\ r^z(\mathbf{q}),\ -r^z(\mathbf{q})\}. \tag{4}$$

We formulate the matching problem as a cost-minimal bipartite assignment between the M ground-truth targets and the K model predictions. Each output embedding $o_j \in \{o_j\}_{j=1}^{K}$ generates parameters for all three object types, yielding $3K$ curve hypotheses in total. For a prediction $j \in \{1, \dots, K\}$ and target $i \in \{1, \dots, M\}$, the matching cost is given by

$$\mathcal{C}_{\text{match}}(j, i) = -\hat{p}_j(c_i) + \mathcal{L}_{\text{param}}(j, i), \tag{5}$$

where $\hat{p}_j(c_i)$ denotes the predicted probability of query j belonging to class c_i [22], and $\mathcal{L}_{\text{param}}$ is a geometry-specific parameter loss defined as

$$\mathcal{L}_{\text{param}}(j, i) = \ell_1(\hat{\mathbf{p}}_j, \mathbf{p}_i) + \begin{cases} \mathcal{L}_{\text{quat}}^{\{\pm 1, \pm r^z\}}(\mathbf{q}_j, \mathbf{q}_i) + \ell_1(\alpha_j, \alpha_i), & c_i = 1, \\ \mathcal{L}_{\text{quat}}^{\{\pm 1\}}(\mathbf{q}_j, \mathbf{q}_i), & c_i = 2, \\ \mathcal{L}_{\text{quat}}^{\{\pm 1, \pm r^z\}}(\mathbf{q}_j, \mathbf{q}_i), & c_i = 3, \end{cases} \tag{6}$$

where $\ell_1(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_1$ is the element-wise ℓ_1-distance. This formulation jointly penalizes discrepancies in object position, incorporates quaternion-based symmetry invariances (e.g., $180°$ rotation around the z-axis for grippers and pallets), and integrates the gripper opening angle when relevant.

Following [22] and [2], we determine the optimal assignment $\sigma \in \mathfrak{S}_K$ via the Hungarian algorithm [29], yielding a minimal-cost permutation of predictions to ground-truth targets. The $K - M$ unmatched predictions are assigned to the no-object class ($c_{\sigma(i)} = 0$), ensuring consistency with the transformer-based detection framework.

3) Loss function: Let $i \in \{1, \dots, K\}$ denote a ground-truth target from the set including the no-object class, and let $j = \sigma(i) \in \{1, \dots, K\}$ be the prediction matched to i. The total loss for the pair (j, i) is defined as

$$\begin{aligned} \mathcal{L}_{\text{total}}(j, i) = &- w_{c_i} \log \hat{\pi}_j \\ &+ \mathbb{1}_{\{c_i \neq 0\}}\, \mathcal{L}_{\text{param}}(j, i) \\ &+ \mathbb{1}_{\{c_i \neq 0\}}\, \mathcal{L}_{\text{CD}}\Big(\Phi(\mathcal{M}^{c_i}, \hat{T}_j), \Phi(\mathcal{M}^{c_i}, T_i)\Big), \end{aligned} \tag{7}$$

where the first term is the weighted cross-entropy loss between the predicted class distribution $\hat{\pi}_j$ and the ground-truth class c_i, the second term is the geometry-aware parameter loss described in Eq. 6, and the third term is a Chamfer distance (CD) between point sets generated from the predicted and ground-truth configurations. The CD loss is given by

$$\mathcal{L}_{\text{CD}}(X, Y) = \tfrac{1}{|X|} \sum_{\mathbf{x} \in X} \min_{\mathbf{y} \in Y} \ell_2(\mathbf{x}, \mathbf{y}) + \tfrac{1}{|Y|} \sum_{\mathbf{y} \in Y} \min_{\mathbf{x} \in X} \ell_2(\mathbf{x}, \mathbf{y}), \tag{8}$$

where $\ell_2(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2$ denotes the Euclidean distance. Here, the mapping $\Phi(\mathcal{M}^{c_i}, T_i)$ applies the configuration T_i to the class-specific mesh $\mathcal{M}^{c_i}$ and yields its corresponding point-based representation. The Chamfer loss $\mathcal{L}_{\text{CD}}$ therefore enforces geometric consistency by comparing the posed mesh of the prediction against that of the ground truth through their transformed point sets, ensuring both accurate placement and correct articulation in 3D space.

If $c_i = 0$ (no-object placeholder), the loss reduces to the cross-entropy term. Class weights w_{c_i} are computed following [2].

Metric	Gripper	Loading Platform	Pallet
	Geometric Quality		
ℓ_2 [m] ↓	0.10 (± 0.07)	0.12 (± 0.09)	0.12 (± 0.08)
Geodesic [°] ↓	4.90 (± 4.42)	0.92 (± 1.06)	12.29 (± 17.08)
Yaw [°] ↓	2.37 (± 2.48)	0.74 (± 0.95)	10.48 (± 14.35)
Opening [°] ↓	6.59 (± 10.80)	–	–
	Detection (mAP = 0.982)		
Det. (AP) ↑	0.997	0.990	0.958

(a) Synthetic test set.

Metric	Gripper	Loading Platform	Pallet
	Geometric Quality		
ℓ_2 [m] ↓	0.13 (± 0.10)	0.24 (± 0.18)	0.14 (± 0.07)
Geodesic [°] ↓	6.37 (± 4.26)	1.57 (± 0.73)	15.18 (± 22.43)
Yaw [°] ↓	2.46 (± 2.20)	1.19 (± 0.85)	12.32 (± 18.02)
Opening [°] ↓	6.69 (± 5.11)	–	–
	Detection (mAP = 0.919)		
Det. (AP) ↑	0.962	0.988	0.805

(b) Real test set (Synthetic-to-real evaluation).

TABLE I: Quantitative test set evaluations for geometric errors and detection AP. Values are mean (± std.).

4) Implementation details:

a) Model: PIRATR is implemented in PyTorch [30] and follows the model configurations of PI3DETR [2]. The backbone consists of 3 Transformer encoder layers and 9 decoder layers [28], with a token feature dimension of 768. The input to the Transformer is a set of 2048 points encoded by a SAModule [7], together with 128 sine-embedded query points [1] used for prediction. Training is initialized from a checkpoint provided by the authors of PI3DETR. We first train for 330 epochs using AdamW [31] with a learning rate of 10^{-4}, followed by continued training of 100 epochs with a reduced learning rate of 10^{-5}. An effective batch size of 132 is achieved with gradient accumulation, and gradients are clipped at an ℓ_2-norm of 0.2. Losses are normalized per object. Training is performed on a single NVIDIA RTX 4090 GPU. Data augmentation includes random point cloud rotations of up to 5° around the x- and y-axes. In addition, with probability $\frac{1}{3}$, Gaussian noise with standard deviation sampled from the interval $(0.0, 0.04]$ is added to the normalized points (see Sec.III-A.3).

b) Real Data Inference: The autonomous forklift is equipped with a Livox Mid-70 sensor operating at a wavelength of 905 nm with a circular field of view of 70.4°, delivering 10 packages of 10k points per second via a ROS2 [32] interface. Points are filtered according to the Livox user manual[2], retaining only those with low noise characteristics in terms of intensity and spatial position. Since our aggregated LiDAR scans can contain between 50k and 400k points, we first discard all points beyond 25m and then subsample the remaining points to 32k using farthest point sampling. This not only aligns the point count and distance range with our synthetic training data but also ensures an even point distribution, mitigating artifacts caused by high-density regions in the raw scans. The filtered points are rotated 180° around the z-axis to align with Blender conventions used for training and then passed to the network described in Sec. III-B. The network's predicted poses for each parametric object are subsequently rotated back -180° around the z-axis to match ROS2 conventions and the original point cloud frame.

[2]https://www.livoxtech.com/mid-70/downloads (accessed on 2025-09-14)

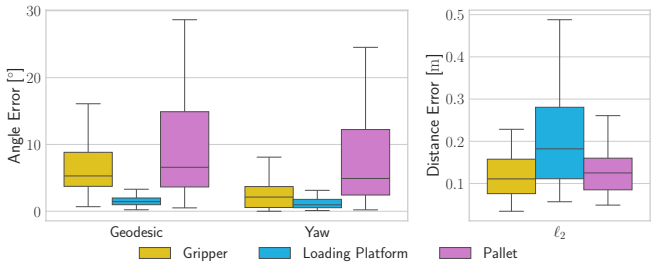


Fig. 6: Boxplots of synthetic-to-real evaluation error distributions for angles and distances.

IV. Experiments

We evaluate PIRATR on both a synthetically generated dataset and a real-world dataset collected with the Livox Mid-70, which was manually annotated. We present quantitative results on both datasets (Sec. IV-A and Sec. IV-B), and we include qualitative evaluations as well as robustness tests (Sec. IV-B.1) on the real-world dataset. The real-world dataset comprises 73 scenes containing 34 grippers, 32 loading platforms, and 70 pallets.

To evaluate our model quantitatively, we use two classes of metrics. Geometric quality: we report the ℓ_2-distance for positional offsets, the geodesic error for full 3D orientation, and the yaw error. For gripper predictions, we additionally measure the error in the predicted opening angle. All angular errors are reported in degrees (°) and the ℓ_2 is reported in meters (m). Detection accuracy: we report (mean) average precision (AP). A prediction j is considered a match with ground truth i if $\hat{c}_j = c_i$ and $\mathcal{L}_{\text{CD}}\Big(\Phi(\mathcal{M}^{c_i}, \hat{T}_j), \Phi(\mathcal{M}^{c_i}, T_i)\Big) < 0.00125$. The matches are reused for the computation of the geometric metrics.

A. Syntetic Dataset Evaluation

Tab. Ia presents the results on the synthetic test dataset. The proposed method attains a mean average precision (mAP) of 0.982 and maintains accurate predictions in terms of ℓ_2-distance, with an average offset below 12 cm across all classes. The loading platform yields the best scores due to its simpler geometry, while the gripper class is also predicted with high accuracy. The pallet class constitutes the most challenging case, yet the method still achieves an average precision of 0.958. Rotation estimation remains the most difficult component compared to the other evaluated metrics.

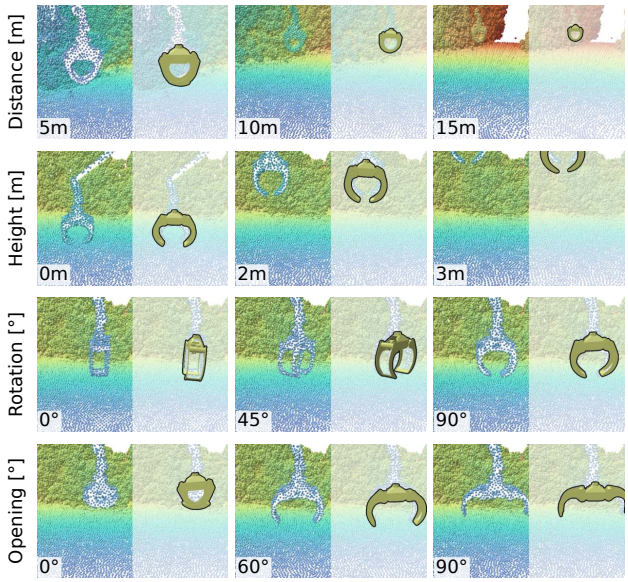


Fig. 7: Synthetic-to-real robustness evaluation on gripper predictions.

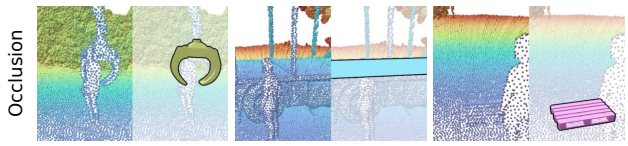


Fig. 8: Synthetic-to-real predictions under occlusion scenarios caused by a human standing in front of the objects, shown from left to right: gripper, loading platform, and pallet.

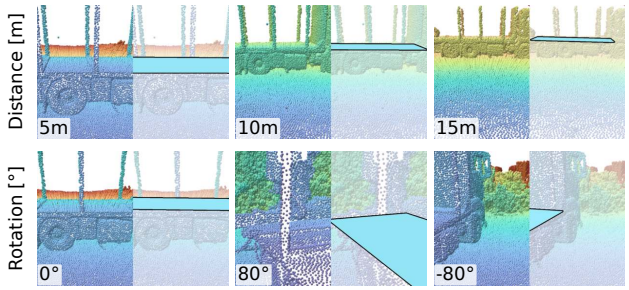


Fig. 9: Synthetic-to-real robustness evaluation on loading platform predictions.

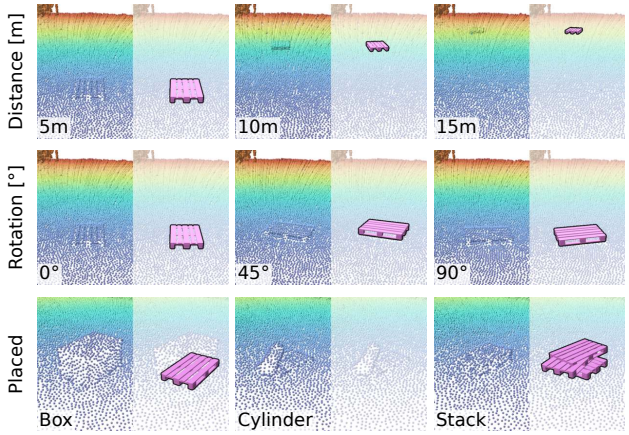


Fig. 10: Synthetic-to-real robustness evaluation on pallet predictions.

B. *Synthetic-to-Real Evaluation*

To assess synthetic-to-real generalization, we evaluate the model trained exclusively on synthetic data on the real world dataset. The results are reported in Tab. Ib. Compared to the synthetic test results in Tab. Ia, PIRATR achieves performance that remains consistent across most metrics. The largest drop occurs for the pallet class, with an average precision of 0.805 compared to 0.958 on synthetic data. This decrease is likely due to the random placement of pallets during synthetic data generation, where some instances appear in vegetation and are still considered valid ground truth if sufficient point coverage is captured. The boxplots in Fig. 6 complement Tab. Ib and provide further insight into the error distributions. For all classes, more than 50% of the angle errors are below $10°$, and more than half of the ℓ_2 errors are below 20 cm. Overall, the results demonstrate that the proposed model transfers reliably from synthetic to real data, as illustrated in Fig. 5.

1) Robustness Tests: In this section, we qualitatively evaluate synthetic-to-real robustness across different class-specific scenarios. Fig. 7 and Fig. 9 show predictions for grippers and loading platforms under variations in distance, height, rotation, and gripper opening angle. In all figures, the point cloud is shown on the left and the corresponding prediction on the right in a grayed area for clarity. PIRATR demonstrates robust performance for both classes, with the only strong inaccuracies observed in the close-up $80°$ loading platform sample. For pallets, Fig. 10 presents cases where additional objects such as boxes, cylinders, or stacked pallets are placed on top. In these scenarios, pallets are not detected correctly when a cylinder is placed on top, and for stacks the rotation and number of predictions are inaccurate. Cylinders on pallets were not included in the training data. Finally, Fig. 8 shows tests with a person occluding the target object across all classes. Despite the absence of such cases in training, the method consistently detects the target objects.

2) Point Cloud Accumulation & Runtime: Since LiDAR point cloud acquisition depends on aggregation time, we evaluate the performance of the synthetically trained model on different point aggregation counts of real point clouds. Tab. II reports metrics for 50k, 200k, and 400k points, which are fed into the model after applying the preprocessing described in Sec. III-B.4.b. The 400k setting, which is closest to the synthetic data generation setup, delivers the most stable performance across metrics. Nevertheless, all aggregation counts yield decent performance. From raw input to final predictions, the method requires 218ms (± 59ms), 371ms (± 62ms), and 598ms (± 76ms) for 50k, 200k, and 400k on a NVIDIA RTX 4090 GPU, respectively.

Metric	Gripper			Loading Platform			Pallet		
	50k	200k	400k	50k	200k	400k	50k	200k	400k
	Geometric Quality								
ℓ_2 [m] ↓	**0.13**	**0.13**	**0.13**	0.21	0.28	**0.24**	**0.14**	0.15	**0.14**
Geodesic [°] ↓	9.12	7.15	**6.37**	1.71	**1.37**	1.57	**12.52**	17.90	15.18
Yaw [°] ↓	3.32	2.47	**2.46**	1.24	**0.96**	1.19	**10.19**	14.69	12.32
Opening [°] ↓	10.49	8.90	**6.69**	–	–	–	–	–	–
	Detection								
Det. (AP) ↑	0.923	**0.992**	0.962	0.940	0.940	**0.988**	**0.805**	0.710	**0.805**

TABLE II: Synthetic-to-real geometric and detection metrics for different point cloud accumulation sizes (50k, 200k, 400k points).

V. Conclusion & Outlook

We presented PIRATR, an end-to-end framework for parametric 3D object detection that jointly estimates multi-class 6-DoF poses and class-specific attributes from partially observed point clouds. PIRATR showed strong synthetic-to-real transfer on challenging outdoor LiDAR scans despite training solely on simulated data. Its practical relevance was demonstrated in an automated forklift operation outdoors. Future work includes extending supported object classes, improving synthetic data realism, reducing training times, and incorporating temporal information for greater robustness.

References

[1] I. Misra, R. Girdhar, and A. Joulin, "An end-to-end transformer model for 3d object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 2906–2917. 1, 2, 4, 6

[2] F. F. Oberweger, M. Schwingshackl, and V. Staderini, "Pi3detr: Parametric instance detection of 3d point cloud edges with a geometry-aware 3detr," *arXiv preprint arXiv:2509.03262*, 2025. 1, 2, 3, 4, 5, 6

[3] V. Staderini, T. Glück, P. Schneider, R. Mecca, and A. Kugi, "Surface sampling for optimal viewpoint generation," in *IEEE 13th International Conference on Pattern Recognition Systems*, 2023, pp. 1–7. 2

[4] V. Staderini, T. Glück, P. Schneider, and A. Kugi, "Visual quality inspection planning: A model-based framework for generating optimal and feasible inspection poses," in *IEEE International Conference on Intelligent Robots and Systems*, 2024, pp. 10 799–10 806. 2

[5] Y. Huang, S. Zhou, J. Zhang, J. Dong, and N. Zheng, "Voxel or pillar: Exploring efficient point cloud representation for 3d object detection," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 38, no. 3, 2024, pp. 2426–2435. 2

[6] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660. 2

[7] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017. 2, 4, 6

[8] G. Qian, Y. Li, H. Peng, J. Mai, H. Hammoud, M. Elhoseiny, and B. Ghanem, "Pointnext: Revisiting pointnet++ with improved training and scaling strategies," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2

[9] S. Mohapatra, S. Yogamani, H. Gotzig, S. Milz, and P. Mader, "Bevdetnet: Bird's eye view lidar point cloud based real-time 3d object detection for autonomous driving," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 2809–2815. 2

[10] A. Barrera, J. Beltrán, C. Guindel, J. A. Iglesias, and F. García, "Birdnet+: Two-stage 3d object detection in lidar through a sparsity-invariant bird's eye view," *IEEE Access*, vol. 9, pp. 160 299–160 316, 2021. 2

[11] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12 697–12 705. 2

[12] L. Li, F. Yang, H. Zhu, D. Li, Y. Li, and L. Tang, "An improved ransac for 3d point cloud plane segmentation based on normal distribution transformation cells," *Remote Sensing*, vol. 9, no. 5, 2017. 2

[13] S. Moradi, D. Laurendeau, and C. Gosselin, "Multiple cylinder extraction from organized point clouds," *Sensors*, vol. 21, no. 22, 2021. 2

[14] S. Xia, D. Chen, R. Wang, J. Li, and X. Zhang, "Geometric primitives in lidar point clouds: A review," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 685–707, 2020. 2

[15] Z. Li and J. Shan, "Ransac-based multi primitive building reconstruction from 3d point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 185, pp. 247–260, 2022. 2

[16] Y. Li, S. Liu, X. Yang, J. Guo, J. Guo, and Y. Guo, "Surface and edge detection for primitive fitting of point clouds," in *ACM SIGGRAPH 2023 Conference Proceedings*, ser. SIGGRAPH '23. New York, NY, USA: Association for Computing Machinery, 2023. 2

[17] Q. Liu, S. Xu, J. Xiao, and Y. Wang, "Sharp feature-preserving 3d mesh reconstruction from point clouds based on primitive detection," *Remote Sensing*, vol. 15, no. 12, 2023. 2

[18] C. Wang, X. Ma, B. Wang, S. Tang, Y. Meng, and P. Jiang, "Point2primitive: Cad reconstruction from point cloud by direct primitive prediction," *arXiv preprint arXiv:2505.02043*, 2025. 2

[19] M. A. Uy, Y.-Y. Chang, M. Sung, P. Goel, J. G. Lambourne, T. Birdal, and L. J. Guibas, "Point2cyl: Reverse engineering 3d objects from point clouds to extrusion cylinders," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 850–11 860. 2

[20] S. Huang, R. Wang, B. Guo, and H. Yang, "Pbwr: Parametric-building-wireframe reconstruction from aerial lidar point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 27 778–27 787. 2

[21] C. Wang, W. Sun, X. Ma, and F. Deng, "Point2skh: End-to-end parametric primitive inference from point clouds with improved denoising transformer," *Computer-Aided Design*, vol. 181, p. 103838, 2025. 2

[22] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229. 2, 5

[23] Y. Shao, Z. Fan, B. Zhu, M. Zhou, Z. Chen, and J. Lu, "A novel pallet detection method for automated guided vehicles based on point cloud data," *Sensors*, vol. 22, no. 20, 2022. 2

[24] C. Beleznai, L. Reisinger, W. Pointner, and M. Murschitz, "Pallet detection and 3d pose estimation via geometric cues learned from synthetic data," in *International Conference on Pattern Recognition and Artificial Intelligence*. Springer, 2024, pp. 281–295. 2

[25] J. Huemer, M. Murschitz, M. Schörghuber, L. Reisinger, T. Kadiofsky, C. Weidinger, M. Niedermeyer, B. Widy, M. Zeilinger, C. Beleznai *et al.*, "Adapt: An autonomous forklift for construction site operation," *arXiv preprint arXiv:2503.14331*, 2025. 2

[26] W. Zou, D. Shen, P. Cao, C. Lin, and J. Zhu, "Fast positioning method of truck compartment based on plane segmentation," *IEEE Journal of Radio Frequency Identification*, vol. 6, pp. 774–778, 2022. 2

[27] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 12, pp. 4338–4364, 2020. 3

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017. 4, 6

[29] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955. 5

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019. 6

[31] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019. 6

[32] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science robotics*, vol. 7, no. 66, p. eabm6074, 2022. 6