mathematics

MDPI

Article

Symmetry-Aware Orientation Estimation via Outer-Product Rotation Targets in YOLO26

Rok Pahič * and Aleš Hace

Faculty of Electrical Engineering and Computer Science, University of Maribor, Koroška cesta 46, SI-2000 Maribor, Slovenia; ales.hace@um.si
* Correspondence: rok.pahic2@um.si

Abstract

Regressing the 3D orientation of a symmetric object with a signed rotation representation is ill posed: the rotations that render an indistinguishable image form an equivalence class, so physically identical poses receive inconsistent labels. This is a common case in robotic manipulation, bin picking, and part feeding, where machined parts are routinely symmetric about an axis or a mirror plane. We extend the YOLO26 segmentation architecture with a rotation-regression branch whose target, building on Saxena et al., gives each sign-ambiguous axis its own outer product rather than a signed vector, and each directed axis a signed vector, so physically equivalent poses map to the same target of 6, 9, or 12 values instead of 9 to 39. The training data comes from a MuJoCo-based synthetic rendering pipeline with an on-the-fly composition loader that combines randomly posed, rotated part instances into each training image. We evaluate the outer-product representation against direct, signed regression on four part categories spanning axial and mirror-plane symmetry, testing both on held-out synthetic images and on 21 real, annotated MVTec ITODD instances, and additionally compare this grayscale-only system with the published SARR networks, including their depth-input variant, on the same real parts using the BOP AR_B metric.

Keywords: orientation estimation; rotation regression; object symmetry; outer-product representation; YOLO26; synthetic data generation

MSC: 68T45; 68T40

1. Introduction

Estimating the orientation of a rigid object from an image amounts to learning a map from image space to the rotation group $SO(3)$. For an object without symmetry, that map is well defined: one image determines one rotation. For a symmetric object, it is not. The rotations that place the object in a visually indistinguishable configuration form an equivalence class, the orbit of the pose under the object's symmetry, and any representation that stores one signed value per pose must pick one member of that orbit arbitrarily. Fitting a regressor to such labels asks a function to reproduce a one-to-many correspondence, and penalizes a correct prediction whenever the annotation happens to select a different member of the same class. This work treats that inconsistency as a property of the regression target rather than of the network, and adopts a target that is invariant under the symmetry, so that every member of an orbit maps to one value; the cases treated here are those in which the ambiguity is a sign flip of an axis (Section 3.2).

Check for updates

Academic Editor: Tie Liu

Received: 26 September 2026
Revised: 27 October 2026
Accepted: 29 October 2026
Published: 3 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

The setting that motivates this is industrial: robotic manipulation, bin picking, assembly, and part-feeding systems, where the pose of a part decides whether it can be grasped, aligned, or processed reliably, and where a predicted pose is often used directly as the initial estimate for a subsequent geometric refinement step such as iterative closest point (ICP) registration. Recent advances in deep learning have shown that orientation and pose can be predicted directly from visual, depth, point-cloud, or simulation-derived data, and in contrast to purely geometric or rule-based approaches, neural networks can learn pose-relevant features from large synthetic or measured datasets and can therefore handle clutter, occlusion, weak texture, and variations in object geometry. Industrial parts are also where the ambiguity above is common, since many machined components are symmetric about an axis or a mirror plane.

Learning-based orientation and pose estimation has been applied to industrial and bin-picking settings in several forms: as a distributional prediction over final part orientations after physical interaction with feeder hardware [1], as point-cloud-driven instance segmentation and pose estimation in cluttered scenes [2], and as keypoint-voting networks tailored to the accuracy demands of grasping and assembly [3]. The symmetries at issue are treated throughout as equivalent proper rotations in $SO(3)$ (Section 3.2) rather than as an actual reflection, which would have determinant -1. Section 2 reviews how prior symmetry-aware and general-purpose 6D pose methods address this ambiguity and positions this work's two remaining gaps relative to them: most symmetry-aware methods resolve the ambiguity through a separate keypoint-matching or point-cloud correspondence stage rather than through the structure of the regression target itself, and the more general difficulty that common rotation representations are themselves discontinuous or ambiguous as regression targets [4,5] is compounded, rather than solved, by symmetry.

Because annotated real-world orientation datasets for arbitrary industrial parts are expensive to collect, several of the works reviewed in Section 2, as well as this one, rely on synthetic training data rendered with randomized textures, backgrounds, and illumination, a strategy known as domain randomization. Prior work has shown that this strategy can allow networks trained purely in simulation to transfer to real camera images without any real-world fine-tuning [6], which is the same transfer this paper relies on when evaluating models trained only on synthetic renders against real photographs.

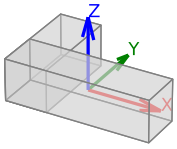
This work addresses the two gaps identified above by combining a single-image detection architecture with a symmetry-aware regression target defined directly on the rotation matrix. The underlying idea of representing a symmetric axis through its sign-invariant outer product rather than as a signed vector is not itself new and follows Saxena et al. [7]; the contribution of this work is to construct that target for each symmetry case with fewer values, giving each sign-ambiguous axis its own outer product and obtaining the third axis by a cross-product, as well as to bring it into a single-stage detection-segmentation architecture, and to evaluate it there. Concretely, we extend the YOLO26 instance-segmentation architecture [8,9] with an additional rotation-regression branch. Training data is generated by a MuJoCo-based [10] synthetic rendering pipeline with domain-randomized textures and backgrounds, combined with a custom on-the-fly composition loader that combines multiple randomly posed and rotated part instances into each training image on the fly. We evaluate five YOLO26 model sizes at several maximum object pixel sizes to select an architecture and training scale that transfers well from synthetic renders to real photographs, and then compare the outer-product representation against direct, signed regression on parts spanning axial and mirror-plane symmetry, evaluating both on held-out synthetic images and on real photographs from the MVTec ITODD (Industrial 3D Object Detection Dataset) dataset [11] using the BOP (Benchmark for 6D Object Pose Estimation) benchmark annotations [12].

In summary, this work's contributions are:

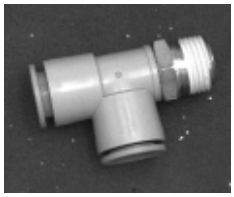
1. A construction of the regression target for each symmetry case with fewer values: A signed vector for each directed axis and the half-vectorized outer product for each sign-ambiguous axis, with the third axis obtained by a cross-product, giving 6, 9, or 12 values instead of the 9 to 39 values of the corresponding representations of Saxena et al. [7] (Table 1);
2. An added rotation-regression branch that represents axes affected by mirror-plane or axial symmetry through their sign-invariant outer product rather than as a directly regressed signed vector, so that physically equivalent orientations of a symmetric part map onto the same training target, while asymmetric parts instead use a direct, continuous two-axis regression target;
3. A unified architecture and decoding procedure for four axis-symmetry cases—asymmetric, axial, one-mirror-plane, and two-mirror-plane—within the same single-stage detection-segmentation architecture, with one model trained per part and symmetry type;
4. An experimental comparison against direct, signed-vector regression on the same parts, architecture, and training data, isolating the effect of the representation itself;
5. An evaluation under synthetic-to-real transfer, training exclusively on domain-randomized synthetic renders of industrial parts and evaluating on real photographs;
6. A system-level benchmark against Kriegler et al.'s SARR networks [13] on the same real parts using the BOP AR_B metric.

Table 1. Object symmetry cases described by equivalent axis directions, their outer-product representation, and the resulting regression-target dimensionality. For comparison, each case also gives the representation of Saxena et al. [7] for the corresponding symmetry and its number of values. Colors identify the object axes as in the equations and figures ($\mathbf{e}_x$ red, $\mathbf{e}_z$ blue).

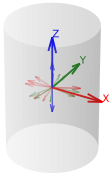
Asymmetric *(example: T-connector)*
Valid orientations: Unique orientation.
Axis equivalence: $\mathbf{e}_z$ is directed, $\mathbf{e}_x$ is directed.
Regression target:
$\mathbf{r} = [\mathbf{e}_z, \mathbf{e}_x] \in \mathbb{R}^6$
Saxena et al. [7]: $[\mathbf{u}_1; \mathbf{u}_2; \mathbf{u}_3] \in \mathbb{R}^9$
(9 values)



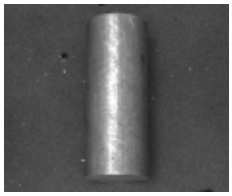
Geometric example


MVTec ITODD example

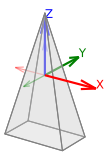
Axial symmetry *(example: Cylinder)*
Valid orientations: Continuous rotation around the symmetry axis and sign flip of the symmetry axis.
Axis equivalence: $\mathbf{e}_z \equiv -\mathbf{e}_z$
Regression target:
$\mathbf{r} = \text{vech}(\mathbf{e}_z\mathbf{e}_z{}^T) \in \mathbb{R}^6$
Saxena et al. [7]: $\mathbf{u}_1\mathbf{u}_1^T \in \mathbb{R}^{3\times3}$ (9 values)



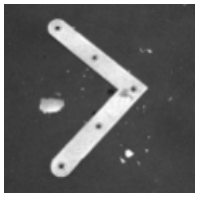
Geometric example


MVTec ITODD example

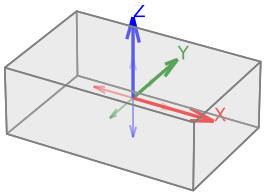
One mirror plane *(example: Planar bracket)*
Valid orientations: One directed axis and one undirected perpendicular axis.
Axis equivalence: $\mathbf{e}_z$ is directed, $\mathbf{e}_x \equiv -\mathbf{e}_x$
Regression target:
$\mathbf{r} = [\mathbf{e}_z, \text{vech}(\mathbf{e}_x\mathbf{e}_x{}^T)] \in \mathbb{R}^9$
Saxena et al. [7]: $([\mathbf{u}_2; \mathbf{u}_3], \mathbf{u}_1\mathbf{u}_1^T) \in \mathbb{R}^6 \times \mathbb{R}^{3\times3}$ (15 values)



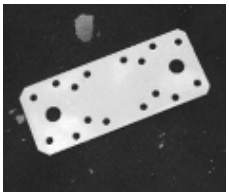
Geometric example


MVTec ITODD example

Two mirror planes *(example: Connector plate)*
Valid orientations: Both selected axes are undirected; four sign combinations are equivalent.
Axis equivalence: $\mathbf{e}_z \equiv -\mathbf{e}_z$, $\mathbf{e}_x \equiv -\mathbf{e}_x$
Regression target:
$\mathbf{r} = [\text{vech}(\mathbf{e}_z\mathbf{e}_z{}^T), \text{vech}(\mathbf{e}_x\mathbf{e}_x{}^T)] \in \mathbb{R}^{12}$
Saxena et al. [7]: $(\mathbf{u}_3, [\mathbf{u}_1; \mathbf{u}_2][\mathbf{u}_1; \mathbf{u}_2]^T) \in \mathbb{R}^3 \times \mathbb{R}^{6\times6}$
(39 values)



Geometric example


MVTec ITODD example

2. Related Work

2.1. Scope of This Review

Candidate work was identified through Google Scholar, arXiv, and the BOP (Benchmark for 6D Object Pose Estimation) challenge leaderboards and associated publications, using combinations of the search terms "symmetric object pose estimation", "rotation regression symmetry ambiguity", "sign-invariant rotation representation", "6D pose estimation industrial parts", and "continuous rotation representation neural network", restricted to 2018–2026. This is not a systematic review in the PRISMA sense; it is intended to establish the research gap addressed by this work and to justify the baseline and architecture choices made in Section 3.

2.2. Symmetry-Aware Object Pose Estimation

Recent symmetry-aware methods address the sign and rotational ambiguity of symmetric parts with point-cloud or RGB-D features combined with a symmetry-aware rotation or keypoint loss [14–16], or with symmetry-aware geometric-correspondence matching for unseen objects [17]. A broader line of general-purpose 6D pose methods handles symmetry directly within the correspondence or regression pipeline instead, using multimodal surface-fragment correspondences [18], unsupervised contrastive correspondence distributions [19], a symmetry-invariant grouped-primitive distance loss [20], or dense-correspondence-based direct regression [21]. More recently, one-to-many pixel-to-vertex correspondences have been proposed to eliminate symmetry ambiguity directly in the correspondence stage, with a direct-regression variant that bypasses the PnP step entirely [22], and a correspondence-free approach matches learned object descriptors instead of dense correspondences to remain agnostic to the specific symmetry class [23]. At the category level, a semantic-guided module built on a large vision model has been used to infer symmetry directly from RGB input without a per-instance shape prior [24].

Most of the methods above, however, operate on point clouds, depth maps, or multi-view RGB-D input, and resolve symmetry through dense keypoint voting or geometric correspondence matching rather than through the structure of the regression target itself. In parallel, single-shot 2D detectors have been extended to predict 6D pose directly from a single RGB image in one forward pass, for example, by regressing the 2D projections of 3D bounding-box corners together with the object class and detection confidence [25]. Combining this single-image detection setting with an explicit, symmetry-aware treatment of rotation ambiguity inside the regression target itself, rather than delegating it to a separate keypoint-matching or point-cloud stage, remains comparatively underexplored in the literature reviewed here.

2.3. Representation-Level Treatments of Rotation Ambiguity

A further difficulty, independent of symmetry, is that many common numerical rotation representations are themselves discontinuous or ambiguous as neural network regression targets [4,5]. Symmetric objects compound this problem: even a representation that is well behaved for a single unique pose can still force the network to choose arbitrarily between several equally valid signed targets when the underlying axis is only defined up to sign. This motivates representations that are simultaneously continuous, in the sense of Zhou et al. [4], and invariant to the sign ambiguity introduced by the object's own symmetry. Kriegler et al. [13] pursue this same goal directly at the representation level, encoding per-axis symmetry degrees into a trigonometric rotation representation that is both unique and continuous across symmetric object classes without requiring a 3D model at inference time; this is a parallel, representation-level route to the same continuity-versus-uniqueness trade-off that the outer-product target in this work resolves through a sign-invariant projec-

tor instead. Beyond the choice of embedding, the two approaches differ in how the target is used. First, SARR is a target for a separately trained network that estimates the rotation from a crop of a single detected object, whereas the outer-product target is regressed by a branch of a single-stage network that detects, segments, and estimates the orientation of every instance in the whole image in one pass. Second, SARR is trained on physically based renderings, whereas our network is trained on rendered parts that are composed on the fly onto real photographs (Section 4.2). Third, SARR applies one target form, with fold counts about the model axes, to every object, whereas our construction treats each axis according to its own ambiguity, a directed axis as a signed vector and an undirected axis through its outer product, so that a new part needs only the list of its undirected axes, and the rotation is assembled from the decoded axes by Gram–Schmidt orthogonalization. The outer-product projector itself is not new: Saxena et al. [7] originally proposed it as a symmetry-invariant regression target, regressing all three axes and, when several axes are ambiguous, one outer product of the stacked axis vectors, without embedding it in a detection architecture or evaluating it under synthetic-to-real transfer. This work builds the target per axis with fewer values (Section 3.3), embeds it in a single-stage detector, and evaluates it under synthetic-to-real transfer.

2.4. Comparison and Research Gap

Two gaps emerge from the literature reviewed above. First, symmetry handling is overwhelmingly delegated to a correspondence-matching, keypoint-voting, or point-cloud stage that sits alongside or after detection, rather than expressed in the structure of a single regression target inside a single-stage detector; only SARR shares this work's representation-level approach, and it is evaluated as a drop-in target for a separately trained CNN rather than integrated into a single-stage detection-segmentation architecture. Second, of the methods above, only SARR is evaluated on the same real, industrial ITODD parts used here, which is why it is selected as the primary baseline in Section 5.5: it is the one prior representation-level, symmetry-aware method with published predictions on the same object instances, which allows a direct comparison of the two systems on the same instances without retraining a third-party method from scratch. SARR is selected as the baseline because Kriegler et al. [13] report it outperforming a range of established BOP-leaderboard methods, including GDR-Net [21] and SC6D [23] among others, under the ARC metric on T-LESS, and specifically outperforming SC6D on ITODD; comparing against SARR is therefore comparing against a method already shown to be competitive with the wider field. YOLO26 was chosen as the base detector because it already provides a mature, real-time single-stage detection-segmentation head, so that symmetry-aware rotation estimation can be added as one additional regression branch on the same backbone and feature maps, rather than requiring a separate pose-estimation network and its own inference pass.

3. Methodology

3.1. Rotation Representation

Rotation representation is an important design choice in learning-based orientation estimation. Classical minimal representations, such as Euler angles, are low-dimensional and intuitive, but they contain singularities and discontinuities. A well-known example is gimbal lock, where two rotational degrees of freedom become aligned and the representation loses one independent degree of freedom. This makes Euler angles unsuitable as a direct regression target for neural networks, especially when the object orientation can vary over the full rotation space.

Another three-parameter representation is the axis–angle representation, where a rotation is described by a unit rotation axis $\mathbf{u} \in \mathbb{R}^3$ and a rotation angle θ. Although this

representation is geometrically meaningful, it also has ambiguities and discontinuities. The axis becomes undefined when the rotation angle is close to zero, and the pairs $(\mathbf{u}, \theta)$ and $(-\mathbf{u}, -\theta)$ represent the same rotation. In addition, rotations with angles close to π can be numerically difficult because small changes in the rotation may cause large changes in the axis direction. These issues can make the axis–angle representation unstable for supervised neural network regression.

Unit quaternions avoid gimbal lock and are widely used for representing 3D rotations. However, they are not unique because $\mathbf{q}$ and $-\mathbf{q}$ represent the same physical rotation. This double-cover property introduces an ambiguity in supervised learning: two different target vectors correspond to the same orientation. Without special handling, a network may be penalized even when it predicts an equivalent quaternion. Zhou et al. [4] showed that all rotation representations of $SO(3)$, the group of 3D rotations, embedded in Euclidean spaces of dimension four or lower are discontinuous, and proposed continuous 5D and 6D representations that are more suitable for neural network regression. Geist et al. [5] further discuss how the choice of rotation representation affects optimization, continuity, and learning behavior in 3D rotation prediction.

Because of these problems, we avoid Euler angles, axis–angle vectors, and quaternion regression as the main network output. Instead, we use a rotation-matrix-based representation. A 3D rotation matrix is defined as

$$\mathbf{R} = \begin{bmatrix} e_{xx} & e_{yx} & e_{zx} \\ e_{xy} & e_{yy} & e_{zy} \\ e_{xz} & e_{yz} & e_{zz} \end{bmatrix} \in SO(3). \tag{1}$$

$$\mathbf{R} = \begin{bmatrix} \mathbf{e}_x & \mathbf{e}_y & \mathbf{e}_z \end{bmatrix} \tag{2}$$

where $\mathbf{e}_x$, $\mathbf{e}_y$, and $\mathbf{e}_z$ are the basis vectors of the object coordinate frame expressed in the camera or world coordinate frame. A valid rotation matrix must satisfy

$$\mathbf{R}^T\mathbf{R} = \mathbf{I}, \qquad \det(\mathbf{R}) = 1. \tag{3}$$

Thus, the three columns of $\mathbf{R}$ are orthonormal and form a right-handed coordinate system.

Using the rotation matrix representation, we can predict only two of the basis vectors of the rotation matrix:

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{e}}_z \\ \hat{\mathbf{e}}_x \end{bmatrix} \in \mathbb{R}^6, \qquad \hat{\mathbf{e}}_z, \hat{\mathbf{e}}_x \in \mathbb{R}^3. \tag{4}$$

These two predicted vectors are then converted into an orthonormal basis using a Gram–Schmidt-like procedure. First, the predicted primary axis $\hat{\mathbf{e}}_z$ is normalized as follows:

$$\mathbf{e}_z = \frac{\hat{\mathbf{e}}_z}{\|\hat{\mathbf{e}}_z\|}. \tag{5}$$

The predicted secondary axis $\hat{\mathbf{e}}_x$ is then made orthogonal to $\mathbf{e}_z$ and normalized, as follows:

$$\mathbf{e}_x = \frac{\hat{\mathbf{e}}_x - (\mathbf{e}_z{}^T\hat{\mathbf{e}}_x)\mathbf{e}_z}{\|\hat{\mathbf{e}}_x - (\mathbf{e}_z{}^T\hat{\mathbf{e}}_x)\mathbf{e}_z\|}. \tag{6}$$

Finally, the remaining basis vector is computed using the cross-product:

$$\mathbf{e}_y = \mathbf{e}_z \times \mathbf{e}_x. \tag{7}$$

The final rotation matrix is therefore

$$\mathbf{R} = \begin{bmatrix} \mathbf{e}_x & \mathbf{e}_y & \mathbf{e}_z \end{bmatrix}. \tag{8}$$

This representation is low-dimensional, continuous, and directly produces a valid rotation matrix after orthogonalization. It is therefore well suited for neural network regression. It is the continuous 6D representation of Zhou et al. [4], with the columns $\mathbf{e}_z$ and $\mathbf{e}_x$ in place of the first two columns. The Gram–Schmidt step keeps the direction of $\mathbf{e}_z$ exactly, so $\mathbf{e}_z$ is chosen as the symmetry axis of an axially symmetric part and as the directed axis of a part with one mirror plane (Section 3.2).

3.2. Object Symmetries

Object symmetry introduces an ambiguity in orientation learning. The training annotation usually stores one rotation matrix for each object instance, but a symmetric object can have several rotations that describe the same physical pose and produce the same visual appearance (Section 3.3 discusses the consequence of this for training).

In this work, we focus on symmetry cases that can be expressed as sign ambiguities of object-frame axes. The symmetry case of each part is specified from its geometry before training; the network does not infer it from the image. Table 1 summarizes the four cases used here. The vector $\mathbf{e}_z$ denotes the primary object axis, while $\mathbf{e}_x$ denotes a secondary axis that fixes the remaining rotation around $\mathbf{e}_z$. When an axis is undirected, the directions $\mathbf{e}$ and $-\mathbf{e}$ describe the same physical axis.

For axial symmetry, the only meaningful orientation component is the object axis itself, since rotations around it are visually indistinguishable. Objects with one mirror plane have a directed axis that distinguishes the two sides of the object, while the undirected perpendicular axis represents the mirror ambiguity. For objects with two mirror planes, both selected axes are undirected, so all four sign combinations $(\mathbf{e}_z, \mathbf{e}_x)$, $(-\mathbf{e}_z, \mathbf{e}_x)$, $(\mathbf{e}_z, -\mathbf{e}_x)$, and $(-\mathbf{e}_z, -\mathbf{e}_x)$ describe equivalent physical poses.

All four cases above reduce each axis's ambiguity to invariance under the two-element sign-flip group $\{1, -1\}$, i.e., $\mathbf{e} \equiv -\mathbf{e}$, which is exactly what the outer product $\mathbf{e}\mathbf{e}^T$ is invariant to (Section 3.3). This covers objects whose observable symmetry is a two-fold ambiguity per axis. Although not exhaustive, these four cases cover a large proportion of industrial parts whose observable orientation is resolved up to one or two axis sign flips. Two cases fall outside this scope: an object with discrete n-fold rotational symmetry about an axis for $n > 2$ (e.g., a square or hexagonal cross-section, including a cube or other regular box-like part) has a secondary axis with n equivalent positions rather than two, so the outer product's sign-flip invariance is insufficient on its own and would need to be replaced with a higher-order, n-fold-invariant embedding—conceptually a generalization of the same idea, in the spirit of the trigonometric, per-axis symmetry-degree encoding Kriegler et al. [13] use for exactly this purpose, but one that this work does not implement or evaluate; and an object with full spherical symmetry has no observable orientation at all, so pose estimation is not merely unhandled but inapplicable to it by definition.

3.3. Sign-Invariant Axis Representation via Outer Products

Section 3.1 represents object orientation directly through two directed axis vectors, the primary axis $\mathbf{e}_z$ and the secondary axis $\mathbf{e}_x$, from which the full rotation matrix $\mathbf{R} = [\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z]$ (Equation (2)) is reconstructed. This works directly for asymmetric objects, since each of $\mathbf{e}_z$ and $\mathbf{e}_x$ has exactly one physically correct direction.

As shown in Section 3.2, however, symmetric objects introduce sign ambiguities on one or both of these axes: an undirected axis $\mathbf{e}$ is physically equivalent to $-\mathbf{e}$, a mirror-symmetry ambiguity. If such an axis is regressed directly as a signed vector, the same image is labeled $\mathbf{e}$ in some training samples and $-\mathbf{e}$ in others, depending only on which of the two equally valid targets the ground-truth annotation happens to have picked. The problem is not that the network may output the wrong sign—the evaluation metric in

Section 4.4 is insensitive to the sign—but that the training targets for one image contradict each other. A loss on the signed vector is then minimized by predictions between $\mathbf{e}$ and $-\mathbf{e}$, for a balanced assignment by the zero vector, so the direction obtained after normalization is not determined by the training objective. Direct vector regression is therefore an ill-posed learning target for any axis affected by a mirror symmetry.

A useful way to construct a symmetry-invariant descriptor for such an axis is to use its outer product with itself, rather than the signed vector directly. For two general vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$, the outer product is defined as

$$\mathbf{a} \otimes \mathbf{b} = \mathbf{a}\mathbf{b}^T. \tag{9}$$

Explicitly,

$$\mathbf{a}\mathbf{b}^T = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} = \begin{bmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 \\ a_2 b_1 & a_2 b_2 & a_2 b_3 \\ a_3 b_1 & a_3 b_2 & a_3 b_3 \end{bmatrix}. \tag{10}$$

Saxena et al. [7] proposed a symmetry-invariant representation for learning 3D object orientation from images, where orientation vectors derived from the rotation matrix are mapped through tensor or outer-product forms such as $\mathbf{u}\mathbf{u}^T$. This representation assigns the same value to visually equivalent symmetric poses, making it suitable for learning orientations of objects with mirror-plane, rotational, or axial symmetries. The targets used in this work are built from the same idea but with fewer values. Saxena et al. regress all three axes and, when several axes are ambiguous, the outer product of the stacked axis vectors, for example $[\mathbf{u}_1; \mathbf{u}_2][\mathbf{u}_1; \mathbf{u}_2]^T \in \mathbb{R}^{6\times 6}$ for reflection symmetry about two planes, so that their representations of the symmetry cases considered here have between 9 and 39 values. Here, the third axis is not regressed but obtained as the cross-product of the other two, each sign-ambiguous axis receives its own outer product instead of one for the stacked axes, and only the six independent entries of each symmetric outer product are regressed. This gives targets of 6, 9, or 12 values for the four symmetry cases (Section 3.4; Table 1 lists both).

For an undirected unit axis $\mathbf{e}$ ($\|\mathbf{e}\| = 1$), i.e., one for which $\mathbf{e}$ and $-\mathbf{e}$ are physically equivalent, the rank-one matrix

$$\mathbf{M} = \mathbf{e}\mathbf{e}^T \tag{11}$$

is invariant to the sign of the axis because

$$\mathbf{e}\mathbf{e}^T = (-\mathbf{e})(-\mathbf{e})^T. \tag{12}$$

Thus, the network does not need to decide between two equivalent directions.

The original axis can be recovered up to sign from $\mathbf{M}$ by an eigendecomposition. Since $\mathbf{M} = \mathbf{e}\mathbf{e}^T$ with $\|\mathbf{e}\| = 1$ is symmetric positive semidefinite of rank one, its eigenvalues are $1, 0, 0$ and

$$\mathbf{M}\,\mathbf{u}_1 = \lambda_1 \mathbf{u}_1, \quad \lambda_1 = \max_i \lambda_i(\mathbf{M}) = 1, \quad \mathbf{e} = \pm \mathbf{u}_1, \tag{13}$$

where $\mathbf{u}_1 \in \mathbb{R}^3$ is the unit eigenvector of the largest eigenvalue λ_1. For a predicted matrix $\hat{\mathbf{M}}$, which is symmetric by construction but in general not exactly rank one, the same rule returns the unit eigenvector of its largest eigenvalue; this is the closest rank-one projector to $\hat{\mathbf{M}}$ in the Frobenius norm.

Because $\mathbf{M} = \mathbf{e}\mathbf{e}^T \in \mathbb{R}^{3\times 3}$ is a symmetric matrix, its lower triangle is redundant with its upper triangle and does not need to be stored or regressed separately: of its nine entries, only six are independent. We therefore use the half-vectorization operator $\mathrm{vech}(\cdot)$, defined on the space $\mathrm{Sym}(3)$ of real symmetric 3×3 matrices,

$$\text{vech} : \text{Sym}(3) \longrightarrow \mathbb{R}^6, \tag{14}$$

which keeps only the six entries on and above the diagonal,

$$\text{vech}\left(\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{12} & M_{22} & M_{23} \\ M_{13} & M_{23} & M_{33} \end{bmatrix}\right) = [M_{11}, M_{12}, M_{13}, M_{22}, M_{23}, M_{33}], \tag{15}$$

reducing the nine entries of the 3×3 outer product down to a six-element vector, which is what the regression branch actually outputs. The map vech is a linear bijection, so the full symmetric matrix $\hat{\mathbf{M}} = \text{vech}^{-1}(\hat{\mathbf{r}})$ can be reassembled from a predicted vector $\hat{\mathbf{r}}$ whenever it is needed, for example, before recovering the axis by eigendecomposition.

Eigendecomposition is only used to decode $\mathbf{e}$ from $\mathbf{M}$ at inference, not during training, so a predicted matrix that is not exactly rank-one/PSD, or that would decode to a near-degenerate eigengap, is not a training-time problem: the loss compares the predicted six-element vector directly against $\text{vech}(\mathbf{M})$ and never routes through the eigendecomposition. In practice, the trained network already learns to produce well-conditioned matrices, and decoding never failed on our test set. For production use, an explicit safeguard could still be added to check the predicted matrix before decoding and flag an invalid prediction.

This general construction applies to any undirected axis. For an object with one distinguishable axis, such as a cylinder whose main axis $\mathbf{e}_z$ is observable but whose rotation around this axis is ambiguous, the orientation can be represented only by the symmetry axis

$$\mathbf{M}_z = \mathbf{e}_z \mathbf{e}_z{}^T. \tag{16}$$

This representation preserves the direction of the cylinder axis up to sign, but intentionally removes the rotation around that axis. The same construction applies to any other undirected axis, such as the secondary axis $\mathbf{e}_x$ of an object with a single mirror plane, giving $\mathbf{M}_x = \mathbf{e}_x \mathbf{e}_x{}^T$.

3.4. Rotation-Regression Targets per Object Symmetry

Combining a raw vector for each directed axis with a half-vectorized outer product for each undirected axis gives the target dimensionality for each of the symmetry cases considered in this work: three values for a single directed axis, six values for either two directed axes or a single undirected axis, nine values for one directed and one undirected axis, and twelve values for two undirected axes. Building on the general outer-product construction above, this section spells out the concrete regression target $\mathbf{r}$ for each of the four symmetry cases from Section 3.2, matching the per-case dimensionalities already summarized in Table 1. These four cases are not exhaustive of all possible object symmetries, but cover the sign-ambiguity patterns observed on the parts used in this work.

For objects without rotational symmetry, orientation is represented by a 6D vector, as follows:

$$\mathbf{r} = [\mathbf{e}_z, \mathbf{e}_x] \in \mathbb{R}^6, \tag{17}$$

where $\mathbf{e}_z$ and $\mathbf{e}_x$ are the two object axes, in the same primary-then-secondary order as the raw predicted vector $\hat{\mathbf{x}}$ in Section 3.1. These two axes are orthogonalized to reconstruct the full rotation matrix.

For cylindrical objects, only the symmetry axis is ambiguous (Section 3.2), so the target is its half-vectorized outer product, as defined in Section 3.3:

$$\mathbf{r} = \text{vech}\left(\mathbf{e}_z \mathbf{e}_z{}^T\right) \in \mathbb{R}^6. \tag{18}$$

Objects with one mirror plane, such as the planar bracket, represent the directed axis as a raw vector and the undirected perpendicular axis as its half-vectorized outer product:

$$\mathbf{r} = \left[\mathbf{e}_z, \mathrm{vech}\left(\mathbf{e}_x \mathbf{e}_x{}^T\right)\right] \in \mathbb{R}^9. \tag{19}$$

Objects with two mirror planes, such as rectangular box-like parts without higher-order rotational symmetry, have both selected axes undirected, so they are represented jointly by their half-vectorized outer products:

$$\mathbf{r} = \left[\mathrm{vech}\left(\mathbf{e}_z \mathbf{e}_z{}^T\right), \mathrm{vech}\left(\mathbf{e}_x \mathbf{e}_x{}^T\right)\right] \in \mathbb{R}^{12}. \tag{20}$$

These target vectors are used to reconstruct an equivalent full rotation matrix, matching the per-case entries already listed in Table 1.

Decoding into a Right-Handed Rotation Matrix

At inference, the axis of each undirected-axis block is recovered as in Section 3.3, taking either sign, and is passed with the directed axes to the Gram–Schmidt construction of Section 3.1, which yields $\det \mathbf{R} = +1$ for either sign. Flipping $\mathbf{e}_x$ gives $\mathbf{R}R_z(\pi)$ and flipping $\mathbf{e}_z$ gives $\mathbf{R}R_x(\pi)$, so the sign choice only moves $\mathbf{R}$ within its symmetry set (Section 4.4) and does not change the symmetry-aware rotation error. For the cylinder, any unit vector $\mathbf{e}_x \perp \mathbf{e}_z$ completes $\mathbf{R}$.

For every symmetry case above, the vector $\mathbf{r}$ is the regression target that the neural network's rotation-regression output layer, described in Section 4.1, is trained to predict, with the size of that output layer resized to match $\dim(\mathbf{r})$ for the part's symmetry type.

4. Experimental Evaluation

4.1. YOLO26 Segmentation Architecture Modification

The YOLO26 family was chosen as the base architecture for this work for several practical reasons. First, YOLO26 natively supports instance segmentation, so a per-object mask coefficient branch that predicts a short per-instance feature vector already exists in the architecture; this branch could be copied and repurposed to output rotation parameters instead of designing a new prediction head from scratch. Second, YOLO26 introduces native end-to-end inference and lighter detection heads [9], which matters for the intended robotic bin-picking and part-feeding use case, where pose estimates are needed for each incoming camera frame rather than in an offline batch. Third, YOLO26 is released as a family of compound-scaled variants sharing the same backbone and head topology, which makes it possible to study the trade-off between model capacity and rotation-estimation accuracy without changing anything else in the pipeline. Finally, pretrained COCO weights are available for every variant, giving the added rotation branch a well-initialized feature backbone to build on rather than requiring training from random initialization.

The original YOLO26 segmentation architecture predicts bounding boxes, class scores, and mask coefficients for each detected object. In this work, the last prediction head was extended with an additional rotation-regression branch. The standard detection and segmentation outputs were kept unchanged, while an extra branch was added for object orientation prediction.

The additional branch was constructed by copying the existing mask-coefficient branch and replacing its final convolutional layer. The number of output channels of this final convolution was changed to match the required number of rotation-regression parameters. The copied layers retained their existing initialization, while the newly required final-convolution weights were initialized randomly. Therefore, each detection produced the following output:

$$\mathbf{y} = [\mathbf{b}, \mathbf{c}, \mathbf{r}, \mathbf{m}], \tag{21}$$

where **b** denotes the bounding-box prediction, **c** the class scores, **r** the rotation-regression output, and **m** the mask coefficients.

The size and interpretation of **r** depend on the object symmetry, so a different number of regression outputs are trained into this branch for each part type; the concrete target vector for every symmetry case is defined in Section 3.4.

Thus, the same YOLO26 architecture modification and decoding procedure are used in all cases: an additional final regression branch is added to the segmentation head, and only the number and interpretation of its outputs change according to the object symmetry, following the target dimensionalities given in Section 3.4. Each part is trained as a separate single-class model whose rotation branch has $\dim(\mathbf{r})$ outputs for that part's symmetry type, so no model predicts targets of different dimensionality. A single model for parts of several symmetry types would need, for example, $\max \dim(\mathbf{r})$ outputs with the unused entries masked per class; this is not evaluated here.

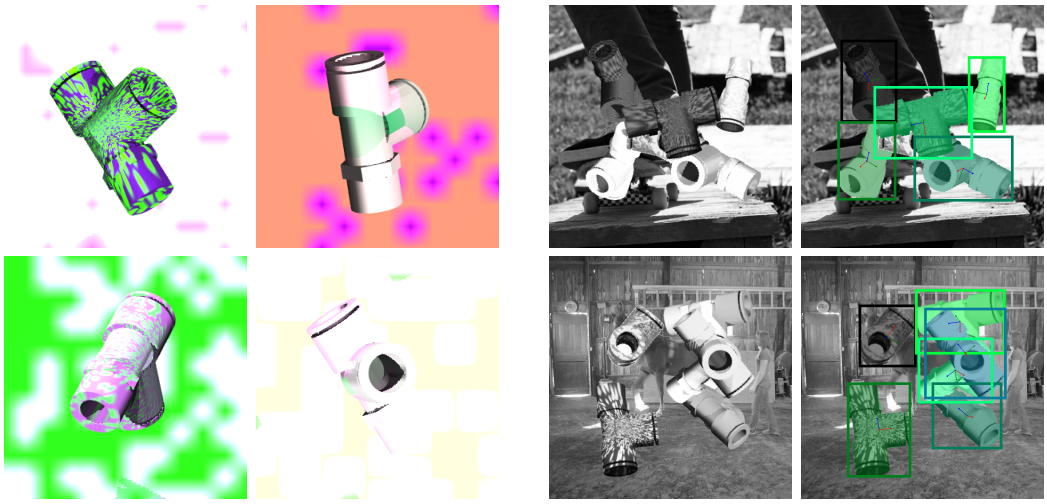
Five compound-scaled variants of the YOLO26 segmentation backbone were used in the evaluation, denoted n (nano), s (small), m (medium), l (large), and x (extra-large) [8]. The variants share the same backbone and head topology and differ only in their depth multiple, width multiple, and maximum channel count, which control the number of block repeats and the channel width of each convolutional layer, and hence the model size and computational cost of each variant prior to adding the rotation-regression branch.

4.2. Synthetic Dataset Generation

The synthetic dataset was generated using a MuJoCo-based [10] rendering pipeline, driven for each object category by a configuration file that specified the object type, physical dimensions, optional mesh, number of samples, dataset split, and desired apparent object size in pixels. The pipeline proceeded through four conceptual stages: scene construction, image rendering, automatic label extraction, and conversion to a COCO-style dataset.

Scene construction placed a single object of the given category at the scene origin, in front of a calibrated virtual camera positioned so that the object appeared at the configured pixel size, under randomized lighting and a textured background. During rendering, an object pose was sampled exactly uniformly over $SO(3)$ for every image, using Shoemake's method [26] to avoid the bias toward corner orientations that a naively normalized random vector would introduce; object texture, background, and illumination were likewise randomized between renders. The part received one of three texture types per render, either a checkerboard, a gradient, or noise, each in two random colors. These textures are not symmetric, but they are resampled for every image and are independent of the pose label, so they give no consistent cue for the sign of an undirected axis. Each rendered sample yielded an RGB image, an object-only segmentation mask, an optional depth map, and simulation metadata such as camera pose and object orientation, split into separate training and test data packs.

Automatic label extraction then derived the remaining annotations from each rendered mask: object contours gave the visible area, axis-aligned and minimum-area bounding boxes, and segmentation polygons, while the object orientation was re-expressed in the camera coordinate frame and stored as a quaternion. Finally, this labeled output was converted into a COCO-style dataset, pairing every image with its category, segmentation polygon, bounding boxes, and camera-relative orientation, ready for use in object detection and rotation estimation. Figure 1 shows four example raw rendered images produced by this pipeline.



(**a**) (**b**)

Figure 1. Synthetic training data at both pipeline stages. (**a**) Four example raw rendered images of the T-connector part from the COCO-style synthetic dataset, prior to composition by the on-the-fly composition loader. (**b**) Two example composed training images produced by the on-the-fly composition loader (left column of each pair), with multiple rotated and pasted object instances used for orientation prediction, next to their corresponding ground-truth annotation (right column of each pair), showing the per-instance bounding boxes, segmentation masks, and camera-relative orientation axes used as targets during training.

4.3. Neural Network Training

In addition to the static, pre-rendered dataset described above, training images were also generated on the fly by a custom on-the-fly composition loader built on top of the COCO-style dataset. Instead of returning a single stored sample, each call to the loader assembled a new composed training image from several independently sampled object instances, so that every training iteration produced a different combination of parts, poses, and layouts rather than reusing a fixed set of pre-composed scenes.

For each composed sample, a set of source instances was drawn from the dataset, optionally from different underlying images, and each instance was first cropped to its labeled object region. Every cropped instance was then rotated in-plane by an angle sampled independently and uniformly from 0° to 360°. This rotation was applied jointly to the image crop, the segmentation mask, and the associated pose label: the bounding box, fitted box, and mask contours were transformed by the corresponding two-dimensional rotation matrix, while the object rotation matrix or quaternion was updated by left-multiplying with the equivalent three-dimensional rotation about the camera viewing axis, so that the stored orientation label remained consistent with the visually rotated part.

The rotated instances were then pasted onto a common canvas arranged as an $n \times n$ grid of cells, with one instance per cell ($n = 2$, i.e., a fixed 2×2, four-cell base grid for every part and experiment in this work), and with 18 additional instances placed at random positions over the full canvas to increase clutter. Each instance was randomly rescaled within 70% to 100% of the size of its target cell and placed at a random offset inside the cell, with a cell overlap of 0.3 allowing neighboring parts to extend into adjacent cells and partially occlude one another. The canvas background was either a plain color estimated from the source images or an external background image, randomly drawn from the 2017 validation split of the Microsoft COCO dataset [27] and cropped and resized to the canvas size. Instances were composited using their mask as an alpha channel, and later-pasted instances occluded the visible mask area of earlier ones. To avoid sharp, unrealistic edges at instance boundaries, the alpha mask was smoothed with a Gaussian blur whose kernel size

scaled with the instance's rendered area, so that larger instances received proportionally more edge blending than smaller ones. After compositing, instance masks and bounding boxes were recomputed from the final visible mask regions, and any instance whose visible mask area fell below 30% of its original, unoccluded area was discarded together with its label, so that heavily occluded or truncated instances were not used as training targets. Figure 1 shows two example composed training images produced by this loader together with their corresponding ground-truth annotations.

Each composed input image was converted to a tensor and then to a three-channel grayscale image before being passed to the network, because MVTec ITODD provides only grayscale images [11]. Replicating the grayscale channel is equivalent to a one-channel network: with identical input channels, the first convolution equals a one-channel convolution whose kernel is the sum of the three pretrained kernels. The COCO-pretrained checkpoint can therefore be used unchanged, with its first-layer filters acting in this channel-summed form.

Training was performed with the AdamW optimizer. The learning rate was set to $\eta = 1.0 \times 10^{-4}$, and weight decay was set to $\lambda = 5.0 \times 10^{-4}$. Early stopping was used based on the validation loss: if the validation loss did not improve for 10 validation checks, training was stopped.

For every reported training condition, the raw synthetic dataset contained 2000 training images and 200 test images before on-the-fly composition. The batch size was 10, 18 additional instances were layered on top of the fixed 2×2 base grid, and each pasted instance was scaled to 70–100% of the size of its target cell.

Algorithm 1 summarizes the complete procedure, including on-the-fly composition, symmetry-specific target construction, optimization, and inference-time decoding. Lines 2–3 sample four instances for the 2×2 grid and 18 further instances for random positions from the raw rendered images of the part. Lines 4–7 crop each instance to its labeled region, draw an in-plane rotation uniformly from $[0^\circ, 360^\circ)$ and a scale uniformly from $[0.70, 1.00]$ of the cell size, and apply both jointly to the image crop, mask, and boxes; the pose label is left-multiplied by the same rotation about the camera viewing axis, so that it stays consistent with the rotated image. Line 8 recomputes the visible masks and boxes after pasting and discards instances with less than 30% of their unoccluded area visible. Line 9 converts the canvas to three-channel grayscale and computes the target $\mathbf{r}_{\text{gt}} = T_s(\mathbf{R}_{\text{gt}})$ of Section 3.4 for each remaining instance. Lines 10–11 run the network and minimize the unchanged YOLO26 loss plus the rotation term, weighted by 10, with AdamW. At inference, lines 13–16 run the network on a grayscale test image and decode each retained detection: directed-axis entries are used directly, the axis of each outer-product block is the eigenvector of its largest eigenvalue (Section 3.3), and the two axes are orthonormalized as in Section 3.1.

The dataset was split into training and validation parts using a validation fraction of 0.2. Therefore, 80% of the generated data was used for training and 20% for validation. Table 2 reports one trained network per configuration, as in the comparison of Kriegler et al. [13]. In the single-run rows of Table 3, each YOLO26-Medium configuration was trained with three random seeds, and the run with the lowest synthetic validation loss is reported; no real images were used for this selection. The YOLO26-Large rows show one of three training runs. To quantify the variation between runs, Table 4 reports YOLO26-Medium over ten seeds as mean $\pm$ standard deviation, with a paired test of the two targets, and the loss-weight ablations of Section 5.3 use three seeds per setting.

Algorithm 1 Symmetry-aware orientation training and inference

Require: Raw synthetic image set $\mathcal{D}$ with masks and poses; symmetry-specific target map T_s; pretrained YOLO26 segmentation model f_θ

for each training step **do**
  Initialize a 2×2 canvas and sample four source instances from $\mathcal{D}$
  Add 18 further sampled instances at random canvas locations
  for each sampled instance **do**
    Sample an in-plane rotation and a scale in $[0.70, 1.00]$
    Transform its image, mask, bounding box, and pose consistently, then paste it on the canvas
  end for
  Recompute visible masks and boxes; discard instances with visible-mask area below 30%
  Convert the canvas to three-channel grayscale and form each target $\mathbf{r}_{\text{gt}} = T_s(\mathbf{R}_{\text{gt}})$
  Predict boxes, classes, masks, and rotation vectors with f_θ
  Minimize $\mathcal{L}_{\text{YOLO26}} + 10\,\text{SmoothL1}(\hat{\mathbf{r}}, \mathbf{r}_{\text{gt}})$ using AdamW
end for
for each inference image **do**
  Convert it to three-channel grayscale and apply f_θ
  For every retained detection, decode $\mathbf{r}$ according to its symmetry class to obtain the orientation
end for

The native YOLO26 segmentation loss, comprising the box, classification, DFL (Distribution Focal Loss, used for the bounding-box regression targets), and segmentation-mask terms, was left unchanged. The regression branch from Section 4.1 is trained by comparing its predicted vector to the corresponding per-symmetry target $\mathbf{r}$ from Section 3.4 with the smooth L1 loss with threshold $\beta = 1$. The loss is averaged over the target elements and over the predictions assigned to ground-truth objects, so each element is weighted by $1/\dim(\mathbf{r})$ and the size of the rotation term does not grow with the target dimension (6, 9, or 12). The resulting scalar is scaled by a regression loss weight of 10 and added to the summed detection, classification, DFL, and segmentation losses, so that a single combined scalar is minimized by the optimizer. This elementwise, equally weighted smooth L1 construction serves as a naive, pragmatic baseline because individual target elements can affect the final rotation error differently; it is not intended to establish this loss as optimal. For a predicted target $\hat{\mathbf{r}}$ and its ground-truth target $\mathbf{r}_{\text{gt}}$ of dimension $d = \dim(\mathbf{r})$, the rotation term is

$$\text{SmoothL1}(\hat{\mathbf{r}}, \mathbf{r}_{\text{gt}}) = \frac{1}{d} \sum_{k=1}^{d} \ell_\beta\left(\hat{r}_k - r_{\text{gt},k}\right), \qquad \ell_\beta(x) = \begin{cases} x^2/(2\beta), & |x| < \beta, \\ |x| - \beta/2, & \text{otherwise}, \end{cases} \tag{22}$$

with $\beta = 1$, additionally averaged over the predictions assigned to ground-truth objects in a batch.

Table 2. Testing results for different YOLO26 architecture sizes and maximum part pixel sizes in the image.

YOLO26 Architecture	Max. Part Pixel Size	Validation Loss	Training Loss	Best Epoch	Duration [mm:ss]	Angular Error (°)	Detections (Matches / GT)	MVTec Angular Error (°, Mean/Median)
Nano	60 px	70.47	72.83	54	14:12	50.25	*5/6*	*23.5/21.1*
	100 px	19.81	20.49	78	25:04	26.78	6/6	19.0/17.0
	190 px	10.33	10.79	89	40:48	13.01	*1/6*	*31.7/31.7*
	280 px	8.06	8.18	80	76:06	12.21	*0/6*	–
Small	60 px	50.25	53.12	54	12:25	36.45	6/6	67.1/38.4
	100 px	14.20	14.42	97	30:52	16.96	6/6	24.3/28.2
	190 px	10.42	10.55	35	19:35	11.27	*4/6*	*17.5/10.5*
	280 px	7.31	7.45	47	48:44	10.68	*0/6*	–
Medium	60 px	18.62	19.21	88	29:28	19.23	6/6	19.7/17.8
	100 px	13.19	13.33	45	16:28	15.83	6/6	19.0/18.9
	190 px	7.69	8.02	58	36:27	9.07	6/6	20.7/21.1
	280 px	6.36	6.70	47	51:57	9.17	*0/6*	–
Large	60 px	16.99	17.84	97	31:28	16.78	6/6	18.7/18.6
	100 px	12.24	12.71	53	20:54	13.96	6/6	22.6/21.4
	190 px	7.28	7.43	66	42:22	9.04	6/6	20.7/18.0
	280 px	5.60	5.83	72	79:42	6.83	*5/6*	*60.2/23.5*
Extra-large	60 px	32.54	33.84	86	21:56	15.46	6/6	20.5/19.4
	100 px	12.04	12.01	55	24:53	12.49	6/6	24.9/23.7
	190 px	6.52	7.03	81	69:57	6.62	6/6	29.3/19.2
	280 px	5.51	5.82	66	104:40	7.42	*5/6*	*76.4/69.5*

Italic entries mark rows where at least one ground-truth instance was not detected (matches < GT); the corresponding MVTec angular error is computed only over the matched subset and understates the model's real-world error. Row shading indicates the maximum part pixel size (darker for larger sizes).

Table 3. YOLO26-Medium and YOLO26-Large orientation prediction results. For each architecture and symmetric part, the outer-product row is followed by the shaded direct signed-regression row; the T-connector is asymmetric and therefore has no outer-product row.

Part	Symmetry/ Representation	Validation Loss	Training Loss	Best Epoch	Angular Error (°)	Detections (Matches/GT)	MVTec Angular Error (°, Mean/Median)
YOLO26-Medium							
T-connector	Asymmetric, direct	13.19	13.33	45	15.83	6/6	19.0/18.9 (max 27.5)
Cylinder	Axial, outer product	7.90	8.30	48	3.98	4/4	5.7/5.8 (max 6.8)
	Axial, direct	48.77	49.34	88	46.81	4/4	58.1/64.8 (max 73.7)
Planar bracket	One mirror plane, outer product	39.38	42.43	93	35.97	6/6	11.9/12.2 (max 17.7)
	One mirror plane, direct	51.85	52.97	81	43.12	*4/6*	*29.8/18.1 (max 77.3)*
Connector plate	Two mirror planes, outer product	30.22	31.68	47	20.36	5/5	10.4/9.7 (max 17.3)
	Two mirror planes, direct	58.66	59.46	47	59.62	5/5	26.6/23.3 (max 50.6)
YOLO26-Large							
T-connector	Asymmetric, direct	12.24	12.71	53	13.96	6/6	22.6/21.4
Cylinder	Axial, outer product	7.81	8.19	56	3.26	4/4	10.0/10.5 (max 13.9)
	Axial, direct	25.58	25.45	33	47.72	4/4	64.0/65.7 (max 76.5)
Planar bracket	One mirror plane, outer product	20.56	22.27	59	43.74	6/6	8.2/8.6 (max 10.0)
	One mirror plane, direct	26.44	27.67	46	49.23	6/6	13.6/9.9 (max 38.6)
Connector plate	Two mirror planes, outer product	15.28	16.20	24	22.40	5/5	14.1/14.5 (max 18.3)
	Two mirror planes, direct	27.32	27.94	61	56.71	5/5	48.3/38.0 (max 69.1)

Italic entries mark the row where at least one ground-truth instance was not detected (matches < GT); the corresponding MVTec angular error is computed only over the matched subset and understates the model's real-world error. The single-run YOLO26-Medium rows show, for each configuration, the best of three runs by synthetic validation loss, and the YOLO26-Large rows one of three runs (Section 4.3); the YOLO26-Medium results over ten seeds are given in Table 4. Shaded rows: direct signed regression; the bold labels separate the YOLO26-Medium and YOLO26-Large results.

Table 4. YOLO26-Medium over ten seeds (mean ± standard deviation over seeds 0–9; for the cylinder, over seeds 1–4 and 6–11, the ten seeds in which both models detected the part, since in seeds 0 and 5, one model detected no cylinder), with the training settings of the YOLO26-Large rows of Table 3. MVTec: Mean error and, in brackets, largest instance error per seed. Paired test: Per-seed difference between the mean MVTec errors (direct minus outer product), number of seeds in which the outer product is better, and two-sided Wilcoxon signed-rank test. *Italic*: At least one instance not detected.

Part	Symmetry/ Representation	Validation Loss	Training Loss	Angular Error (°)	MVTec Angular Error (°, Mean)	Direct − Outer Product (°)	Outer Product Better	Wilcoxon p
T-connector	Asymmetric, direct	13.10 ± 0.69	13.31 ± 0.70	15.03 ± 1.77	*21.4 ± 9.1 (max 41.1 ± 40.6)*	–	–	–
Cylinder	Axial, outer product	8.04 ± 0.31	8.45 ± 0.23	3.43 ± 0.30	*5.5 ± 1.8 (max 8.6 ± 4.2)*	52.9 ± 22.0	10/10	0.002
	Axial, direct	23.93 ± 0.86	23.79 ± 1.13	35.19 ± 9.85	*58.4 ± 22.0 (max 74.5 ± 15.1)*			
Planar bracket	One mirror plane, outer product	22.12 ± 1.74	23.61 ± 1.43	38.58 ± 4.31	*11.1 ± 2.5 (max 16.2 ± 5.8)*	19.1 ± 11.1	10/10	0.002
	One mirror plane, direct	31.19 ± 1.19	32.58 ± 0.95	51.26 ± 1.45	*30.2 ± 12.3 (max 56.4 ± 22.4)*			
Connector plate	Two mirror planes, outer product	15.18 ± 0.70	15.79 ± 0.58	18.41 ± 2.02	15.4 ± 3.9 (max 26.4 ± 7.5)	22.2 ± 15.1	10/10	0.002
	Two mirror planes, direct	28.61 ± 0.83	28.91 ± 0.91	57.38 ± 3.15	37.6 ± 16.5 (max 60.6 ± 25.7)			

Shaded rows: direct signed regression.

In addition to this scalar training and validation loss, which mixes detection, mask, and rotation terms in incompatible units and is therefore not directly interpretable, the rotation error was separately computed and logged in degrees at every validation checkpoint. This was completed by decoding the predicted regression vectors into full rotation matrices as described above, matching predictions to ground-truth poses, and evaluating the angular distance defined in the following subsection. This degree-valued rotation error, rather than the raw scalar loss, is what is reported as the angular error in the result tables.

4.4. Symmetry-Aware Rotation Error

The rotation prediction is evaluated by comparing the predicted rotation matrix $\hat{\mathbf{R}} \in SO(3)$ with the ground-truth rotation matrix $\mathbf{R}_{\text{gt}} \in SO(3)$. For objects without rotational symmetry, the angular error is computed as

$$e(\hat{\mathbf{R}}, \mathbf{R}_{\text{gt}}) = \arccos\left(\frac{\text{tr}\left(\hat{\mathbf{R}}^T \mathbf{R}_{\text{gt}}\right) - 1}{2}\right). \tag{23}$$

This is the geodesic distance on $SO(3)$ under its standard bi-invariant Riemannian metric [28]: the angle of the relative rotation $\hat{\mathbf{R}}^T \mathbf{R}_{\text{gt}}$, equal to the arc length of the shortest path on $SO(3)$ connecting the two orientations. The final error is reported in degrees. As a sanity check, a θ rotation about a shared axis gives $e = \theta$ (e.g., 15° off yields 15° error); with no shared axis at all—e.g., $\hat{\mathbf{R}}$, a 90° rotation about x and $\mathbf{R}_{\text{gt}}$, a 90° rotation about y—the formula still collapses to one geodesic angle, $e = \arccos(-1/2) = 120°$.

For symmetric objects, equivalent ground-truth rotations are considered. Given a symmetry set $\mathcal{S}$, the error is

$$e_{\mathcal{S}}(\hat{\mathbf{R}}, \mathbf{R}_{\text{gt}}) = \min_{S \in \mathcal{S}} \arccos\left(\frac{\text{tr}\left(\hat{\mathbf{R}}^T \mathbf{R}_{\text{gt}} S\right) - 1}{2}\right). \tag{24}$$

For cylindrical symmetry, only the object axis is evaluated. Let $\hat{\mathbf{z}}$ and $\mathbf{z}$ be the third columns of $\hat{\mathbf{R}}$ and $\mathbf{R}_{\text{gt}}$, respectively. Since the cylinder axis is equivalent in both directions, the error is

$$e_{\text{cyl}}(\hat{\mathbf{R}}, \mathbf{R}_{\text{gt}}) = \arccos\left(\left|\hat{\mathbf{z}}^T \mathbf{z}\right|\right). \tag{25}$$

For the one-mirror-plane case from Section 3.2, also known by its point-group name of two-fold conical symmetry, the object is invariant to a rotation of π around its local z axis. The symmetry set is

$$\mathcal{S}_{\text{cone2}} = \{I, R_z(\pi)\}, \tag{26}$$

where

$$R_z(\pi) = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{27}$$

For the two-mirror-plane case from Section 3.2, also known by its point-group name of regular box symmetry, rotations of π around the local x, y, and z axes are treated as equivalent. The symmetry set is

$$\mathcal{S}_{\text{box}} = \{I, R_x(\pi), R_y(\pi), R_z(\pi)\}. \tag{28}$$

$R_z(\pi)$ is given above; the remaining two are

$$R_x(\pi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \qquad R_y(\pi) = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{29}$$

When multiple predictions and ground-truth poses are present in one image, predictions are sorted by confidence score. Each prediction is greedily matched to the unmatched ground-truth object with the highest bounding-box IoU, and each ground-truth object can be matched at most once; a pair is accepted only if its IoU is at least 0.5. This is the matching rule of the standard COCO-style detection evaluation. We verified that an optimal (Hungarian) assignment, which maximizes the total IoU over all pairs, gives identical pairs for every model evaluated in this work. This assignment is based purely on detection geometry and does not use the predicted or ground-truth orientation, so the rotation error computed afterward has no influence on which prediction is paired with which ground-truth pose. This greedy matching, like the network's own NMS step, is a discrete, non-differentiable assignment.

Although the angular error and the symmetry-aware angular error are suitable for evaluation, they are not always suitable as direct training losses for a neural network. The main reason is that the function contains the $\arccos(\cdot)$ operation, whose derivative becomes numerically unstable when its argument is close to -1 or 1. This often happens when the predicted rotation is already close to the ground truth, exactly where stable gradients are important for convergence. In addition, the symmetry-aware formulation contains a minimum over the symmetry set $\mathcal{S}$. This makes the loss piecewise-defined and non-smooth, because the selected symmetry element can change abruptly during training. As a result, small changes in the network output may cause discontinuous changes in the gradient.

4.5. Real-World Data Evaluation

In addition to the validation performed on held-out synthetic images, the best checkpoint of every trained model, selected by the lowest validation loss reached before early stopping, was evaluated on real photographs from the MVTec ITODD dataset [11], using the scene annotations provided in the BOP benchmark format [12]. Since the network was trained exclusively on synthetic, domain-randomized renders, this real-image evaluation measures how well the learned rotation representation transfers to real sensor data rather than to unseen synthetic poses alone.

Two distinct subsets of the MVTec ITODD data were used for this real-image evaluation. The first is a validation subset for which ground-truth 6D poses are publicly available in the BOP annotation format; this subset is used for the quantitative rotation-error evaluation described below. The second is the separate, more challenging BOP19 test subset of MVTec ITODD, for which ground-truth poses are not publicly released; since no ground truth is available for this subset, it is used only for qualitative inspection of the predicted orientations, without any corresponding quantitative rotation error. Figure 2 shows example images from the validation subset, and Figure 3 shows example images from the more challenging test subset.

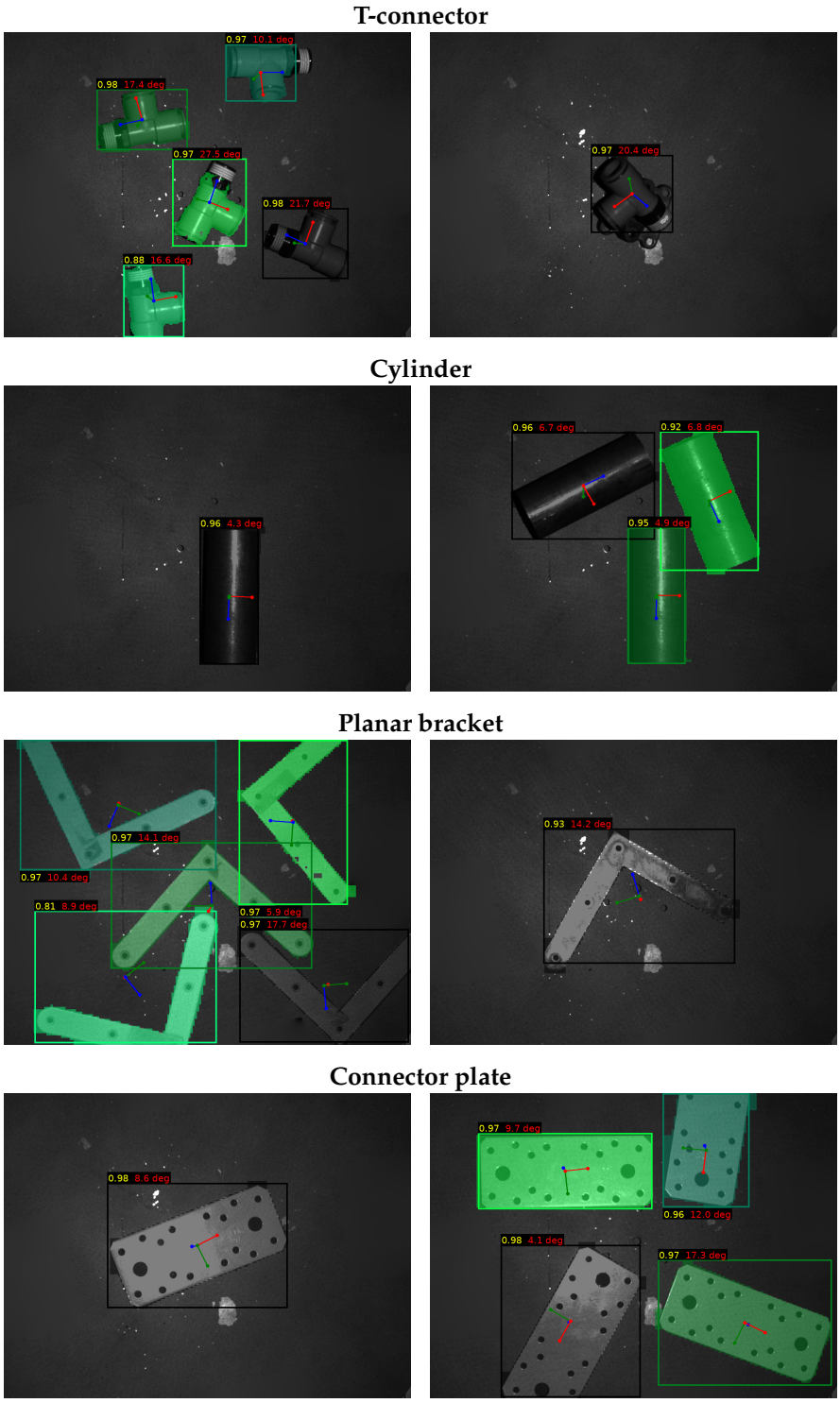


Figure 2. Prediction examples on MVTec ITODD validation images, with two examples for each part type.

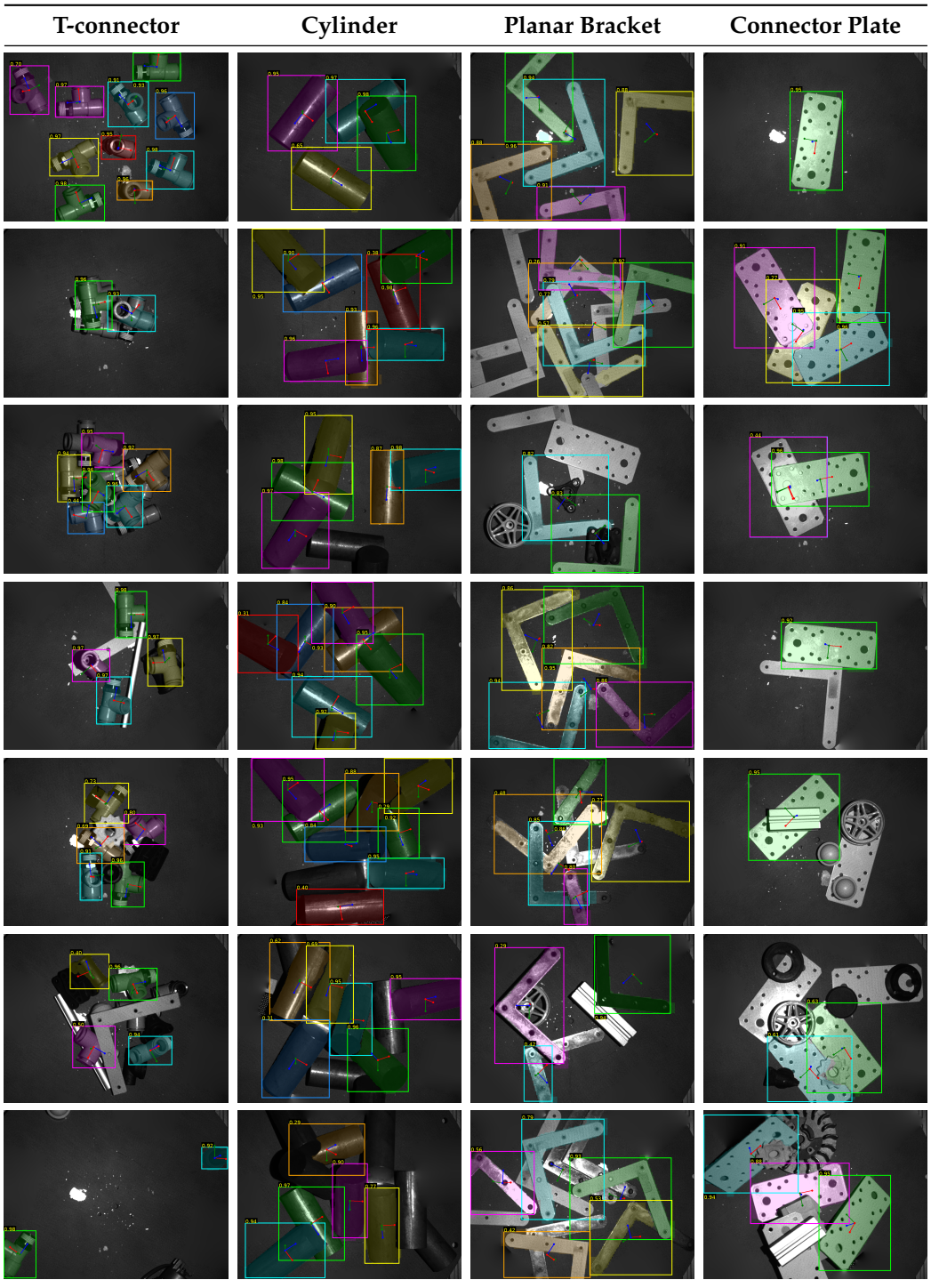


Figure 3. Prediction examples for different object parts. Each column corresponds to one object type.

Each of the four parts represents one sign-flip symmetry case of Section 3.2; the T-connector serves as an asymmetric reference. Within each case, we selected the object with the most validation instances and images; the washer has more instances than the cylinder, but all of them lie flat in a single image.

The real instances serve two different purposes, which we keep separate. The six T-connector instances are the selection set: they were used in Section 5.1 to choose the maximum part pixel size and the architecture. The 15 instances of the three symmetric parts are a test set: their annotations were not used for any choice of architecture, training scale, loss weight, or training setting, and the comparison between the outer-product and

direct representations in Section 5.2 is made on them, with the same selected architecture for both representations.

For each tested part, all annotated ITODD validation scenes containing at least one instance of that part were selected, and every annotated instance contributed one ground-truth 6D pose. Each scene image was rescaled by one fixed factor per part, so that the part appeared at approximately the pixel size used during training. This factor was chosen by running the detector on the scene images at several scales and keeping the scale with the highest detection confidence; neither the ground-truth poses nor the rotation error were used for this choice, and the same factor was used for every model of a part. The trained detection, segmentation, and rotation-regression head was then run on each rescaled scene image, and every resulting detection was decoded into a rotation matrix following the decoding procedure described in Section 3.4. Predictions were matched to the annotated ground-truth objects using the same greedy, confidence-ordered bounding-box IoU matching strategy described for the symmetry-aware rotation error, so that here too the correspondence between a prediction and a ground-truth pose is fixed before any orientation error is computed. The number of successfully matched instances together with the total number of ground-truth poses is reported as "Detections (matches/GT)" in the results tables. Scenes in which no instance of the part was detected at all are reported as having failed to detect the part, rather than being assigned an arbitrarily large rotation error.

5. Results and Discussion

5.1. Architecture Size, Object Scale, and Real-World Transfer

Before evaluating the outer-product training scheme, a suitable network architecture size and input image scale first had to be chosen. Five compound-scaled YOLO26 variants (nano, small, medium, large, and extra-large) were evaluated at four maximum part pixel sizes (60, 100, 190, and 280 px) on the T-connector part, which does not require outer-product training and can therefore be used as a controlled test case. Here, the maximum part pixel size denotes the largest bounding-box side length of the rendered part in the training image, i.e., the effective object scale used during synthetic data generation; the results for all twenty combinations are reported in Table 2.

The 100 px maximum part size was chosen first to match the apparent object scale that YOLO26's COCO-pretrained backbone is best adapted to. The COCO evaluation protocol splits objects into small (area $< 32^2$ px^2), medium ($32^2 \leq$ area $< 96^2$ px^2), and large (area $\geq 96^2$ px^2) categories [27], and in the COCO training set, only 41.43% of object instances are small while 34.4% are medium and 24.2% are large [29], so a large majority of the objects YOLO26 was pretrained on have a bounding-box side length at or above the 96 px medium/large boundary. A part rendered with a maximum pixel size of 100 px therefore lies just above this boundary, matching the scale at which most of the backbone's pretrained features were learned, rather than the small-object regime in which COCO-pretrained detectors are known to perform worst. This choice proved to give the most consistently good results across all five architectures: at 100 px, every variant detected and matched all six ground-truth poses in the real-image MVTec ITODD test set, with mean rotation errors ranging from 19.0° (medium) to 24.9° (extra-large).

To try to improve on this result, 280 px was tested next, which corresponds approximately to the apparent part size in the unscaled MVTec test images. This setting achieved the lowest synthetic validation and training losses of all four scales for every architecture (down to 5.51 for extra-large), but real-world generalization broke down: the nano, small, and medium variants failed to detect the part at all in the MVTec scenes, and even the large and extra-large variants, which did produce detections, matched only five of six ground-truth poses with much larger mean rotation errors (60.2° and 76.4°, respectively).

Training on such large, close-up synthetic crops therefore does not transfer to real camera images, despite the lower synthetic loss—evidence that the network overfits to a training-time object scale that does not match the real-world scale rather than genuinely improving pose estimation.

One plausible explanation is that at 100 px, the rendered part occupies few enough pixels that the network can only rely on coarse shape and silhouette cues, which transfer well because they do not depend on exact rendering fidelity; at 280 px, each instance carries far more per-pixel synthetic detail (the domain-randomization renderer's specific texture and shading, edge aliasing), and the network has both the capacity and the incentive to fit that fine detail to reduce loss, even though this detail is specific to the renderer and does not recur in the same form in a real photograph.

This is a small sample (Section 5.6), so this explanation should be read as a plausible mechanistic account consistent with the observed pattern, not as independently verified causes.

An intermediate scale of 190 px was tried next, half-way between 100 and 280 px, to check whether a smaller step away from 100 px could retain the improved synthetic loss while avoiding the detection failures seen at 280 px. Validation loss again improved over 100 px for every architecture, but real-image transfer remained unreliable: the nano variant matched only one of six MVTec poses (31.7° mean/median error) and the small variant matched only four of six (17.5°/10.5°), while medium, large, and extra-large detected all six instances but with mean errors (20.7°, 20.7°, and 29.3°) that were no better, and in most cases worse, than at 100 px.

Finally, 60 px was evaluated as a lower bound, placing the apparent object scale below rather than above the 96 px COCO medium/large boundary. This setting produced the worst synthetic validation and training losses of all four scales for every architecture (up to 70.47 for nano) and correspondingly large angular errors, confirming that shrinking the apparent object size below this boundary hurts accuracy, as expected from the pretraining-distribution argument above; it also gave the weakest, or least reliable, real-image results (e.g., only five of six MVTec poses matched for nano).

Together, these four scales show a consistent trade-off: synthetic validation loss decreases monotonically with pixel size for every architecture, while real-world generalization follows the opposite trend—it is best at 100 px, already degraded at 190 px, and largely absent at 280 px. This indicates that 100 px is the scale that best balances the network's ability to fit the training distribution against the domain gap between the synthetic renders and the real MVTec ITODD images, and it was therefore adopted as the fixed maximum part pixel size for the remaining experiments.

At this fixed 100 px scale, architecture size still has a measurable effect on the synthetic metrics. Training and validation loss decrease monotonically from nano (20.49/19.81) to extra-large (12.01/12.04), and the symmetry-aware angular error follows the same trend, from 26.78° for nano down to 12.49° for extra-large. Real-world performance on the MVTec ITODD test set does not follow this trend, however: the medium variant gives the lowest mean MVTec angular error (19.0°) of all five architectures, effectively tied with nano (19.0°, although nano's median error of 17.0° is the lowest overall), while large, small, and extra-large trail behind with mean errors of 22.6°, 24.3°, and 24.9°, respectively. Medium therefore combines a synthetic validation loss (13.19) close to that of the larger architectures with the best real-world generalization. Nano, despite matching all six real-world poses, has by far the weakest synthetic fit, reflecting its limited capacity, and extra-large—despite having the best synthetic metrics—gives the worst mean real-world error of all five architectures. Medium was therefore selected as the architecture used for the remaining, outer-product-based experiments in this work. This selection, like the choice of the 100 px scale, used only the six real T-connector instances; the 15 real instances of the three symmetric parts

evaluated in Section 5.2 were not used to select the architecture or the training scale (Section 4.5).

5.2. Evaluating the Outer-Product Representation

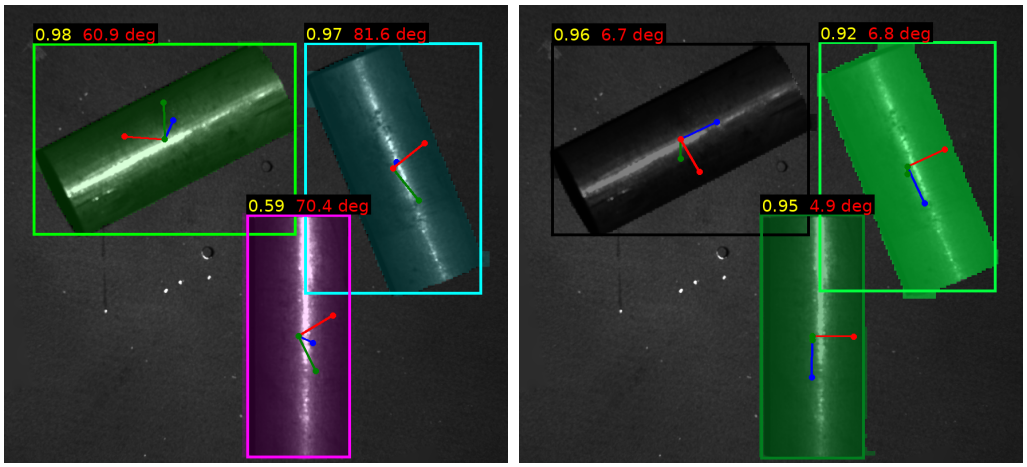
Table 3 compares, for each symmetric part, a model trained with the outer-product representation from Section 3.4 against a model trained with the direct, signed regression target; the asymmetric T-connector has no outer-product counterpart and serves only as a reference point. YOLO26-Medium is the primary architecture because it achieves the best overall balance of real-world transfer and synthetic fit in Table 2. YOLO26-Large is included as a higher-capacity confirmation because its 100 px architecture-scale run detected all six real T-connector instances. For the symmetric-part comparison rows, the table reports the maximum instance error in parentheses alongside the mean and median MVTec angular error: the largest single-instance angular error among the matched real-world instances for that part. This indicator captures the worst-case failure on real images, which the mean and median can mask.

For YOLO26-Medium, the ten-seed results in Table 4 show that the outer-product target gives lower errors than the direct target for all three symmetric parts, both on the synthetic validation images and on the real MVTec instances. The synthetic angular error is $3.43 \pm 0.30^\circ$ versus $35.19 \pm 9.85^\circ$ for the cylinder, $38.58 \pm 4.31^\circ$ versus $51.26 \pm 1.45^\circ$ for the planar bracket, and $18.41 \pm 2.02^\circ$ versus $57.38 \pm 3.15^\circ$ for the connector plate. The mean MVTec errors are $5.5 \pm 1.8^\circ$ versus $58.4 \pm 22.0^\circ$, $11.1 \pm 2.5^\circ$ versus $30.2 \pm 12.3^\circ$, and $15.4 \pm 3.9^\circ$ versus $37.6 \pm 16.5^\circ$, and in a paired comparison over the seeds, the outer-product model has the lower error in every compared seed (Wilcoxon signed-rank test, $p = 0.002$ for each of the three parts). The single-run rows of Table 3 show the same ordering for all three parts, with mean MVTec errors of 5.7° versus 58.1°, 11.9° versus 29.8°, and 10.4° versus 26.6°.

The YOLO26-Large rows provide the intended second-capacity check. They reproduce the outer-product advantage on all three symmetric parts: mean MVTec error is 10.0° versus 64.0° for the cylinder, 8.2° versus 13.6° for the planar bracket, and 14.1° versus 48.3° for the connector plate. All 36/36 model-evaluation detections are matched across the seven YOLO26-Large rows, covering the 21 unique real MVTec poses, so this comparison does not depend on differing detection counts. YOLO26-Large is not claimed to be uniformly better than YOLO26-Medium: compared with the ten-seed Medium means, its outer-product result is better for the planar bracket and the connector plate but worse for the cylinder. These results therefore confirm the representation-level effect at a second model capacity.

The large errors of direct regression cannot be explained by the network outputting the wrong sign of an axis, because the symmetry-aware metric treats $\mathbf{e}$ and $-\mathbf{e}$ as the same axis. They are instead consistent with the contradictory-target mechanism of Section 3.3: when the same image is labeled $\mathbf{e}$ in some samples and $-\mathbf{e}$ in others, the loss pulls the raw signed prediction toward the zero vector, and the direction left after normalization is only weakly determined by the image. The higher errors of the direct models and the far larger maximum per-instance MVTec errors overall (74.5°, 56.4°, and 60.6° on average over the ten seeds for direct regression on the cylinder, planar bracket, and connector plate, respectively, versus only 8.6°, 16.2°, and 26.4° for the outer-product models) match this picture: direct regression occasionally produces very large orientation errors on real images that the outer-product representation largely avoids. Figure 4 shows this on a real image with three cylinders, all lying with their axis in the image plane. Both models detect all three. The predicted z axis (blue) should be aligned with the cylinder axis; the outer-product model recovers it to within 4.9° to 6.8°, whereas the direct model places it nearly perpendicular to the cylinder axis, pointing out of the image plane, with errors of

60.9° to 81.6°. The direct prediction thus balances in the middle between **e** and −**e**, as the contradictory-target mechanism suggests. Both are equally plausible labels for the same appearance, so the component of the signed prediction along the cylinder axis is pulled toward zero, and only directions perpendicular to it remain after normalization.



(**a**) (**b**)

Figure 4. Real MVTec ITODD validation image with three cylinders (scene 1, image 233), predicted by the YOLO26-Medium single-run models of Table 3 trained with (**a**) the direct signed target and (**b**) the outer-product target; (**b**) is the cylinder example of Figure 2. Boxes and masks show the detections with a confidence of at least 0.5, with their confidence and their rotation error; the lines show the predicted axes; the predicted z axis (blue) should be aligned with the cylinder axis. All three cylinders lie with their axis in the image plane.

Table 3 also shows that for the two flat parts, the planar bracket, and the connector plate, the error on the synthetic validation images is higher than on the real MVTec ITODD images, while the cylinder and the T-connector show the opposite. We attribute this to the synthetic validation set being the harder test for flat parts. Its poses are sampled uniformly over $SO(3)$ (Section 4.2), so it contains many views in which a flat part is seen edge-on and little of its shape is visible; the composed images also contain strongly occluded instances, down to 30% of their visible area. On the real support, in contrast, the flat parts rest in physically stable poses, lie flat, show their largest surface to the camera, and are rarely occluded. The cylinder has no such uninformative side view: its axis is visible from almost every direction, only the small fraction of views along the axis is ambiguous, and its stable real pose on its side is common in the synthetic data as well. For the cylinder and the asymmetric T-connector, the synthetic-to-real domain gap therefore dominates the real-image error. Both errors are computed with the same error definition (Section 4.4), so the difference reflects the different pose distributions rather than the evaluation.

Complementing the quantitative results in Table 3, Figure 2 shows example predictions on validation images for all four parts. Figure 3 shows further qualitative prediction examples corresponding to the quantitative results of Table 3. The images in Figure 3 are drawn from the BOP19 test subset described in Section 4.5, for which MVTec ITODD does not publish ground-truth poses, so only a qualitative assessment is possible here. Training included partial occlusion because composed instances could overlap, although instances with less than 30% of their original visible area were excluded as training targets. In qualitative inspection, no large coordinate-system error was apparent among the detected partially occluded parts: the axis pair tracks the cylinder's visible long axis, and the axis triads on the T-connector, planar bracket, and connector plate instances follow each part's stem, arm, or elongation direction even in cluttered, multi-instance scenes with partial occlusion between neighboring parts. Detection confidence varies across instances, from

as low as 0.65–0.70 for small or partially occluded instances up to above 0.95 for well-separated, unoccluded ones. Missed detections occur across all four categories, even for instances that are only slightly occluded by a neighboring part. Several Figure 3 scenes also contain substantial occlusion; in some of these qualitative examples, an occluded instance has no associated detection or segmentation; the observed qualitative failures therefore occur before an orientation is produced. Occlusion may nevertheless hide visual details used by the network to infer orientation, so orientation accuracy could also degrade. Because BOP19 does not release the corresponding ground-truth poses, neither the missed-detection rate nor orientation error as a function of occlusion severity can be measured on these images.

5.3. Ablation of the Rotation-Regression Loss Weight

To assess the choice of the rotation-regression loss weight of 10, we trained YOLO26-Medium at 100 px on the T-connector across successive orders of magnitude in the weights $\{0.1, 1, 10, 100, 1000\}$, repeating each weight three times with independent random seeds while holding architecture, data composition, optimizer, and training schedule fixed. Table 5 reports the resulting synthetic validation loss, synthetic angular error, and real MVTec angular error (mean and median, ± sample standard deviation).

Table 5. Loss-weight ablation: Three-seed mean ± sample standard deviation, YOLO26-Medium, 100 px, for the T-connector and the connector plate. Bold marks the best value per column and part. *Italic* MVTec entries mark weights for which at least one seed missed a real instance; their error is computed over the matched instances only and is not marked as best.

Weight	Best val. Loss	Synthetic val. Error (°)	Synthetic Test Error (°)	MVTec Mean Error (°)	MVTec Median Error (°)
T-connector (asymmetric)					
0.1	11.97 ± 0.16	41.00 ± 3.64	46.37 ± 6.65	28.42 ± 11.42	20.88 ± 5.12
1	12.19 ± 0.10	24.00 ± 1.23	27.83 ± 1.69	21.01 ± 4.13	21.56 ± 3.44
10	13.14 ± 0.99	15.52 ± 2.57	17.42 ± 3.50	**19.80 ± 0.62**	19.81 ± 1.74
100	24.63 ± 0.63	**12.03 ± 0.57**	**16.51 ± 0.57**	26.27 ± 11.44	**16.76 ± 1.26**
1000	164.36 ± 16.96	15.34 ± 1.49	19.92 ± 1.43	28.21 ± 13.17	21.16 ± 2.94
Connector plate (two mirror planes)					
0.1	14.87 ± 0.50	30.73 ± 1.25	32.60 ± 1.27	*15.81 ± 4.07*	*17.05 ± 5.84*
1	**14.43 ± 0.74**	26.26 ± 1.89	27.67 ± 2.39	*11.62 ± 1.02*	*9.17 ± 2.48*
10	14.87 ± 0.74	18.47 ± 1.70	20.08 ± 2.38	**13.25 ± 1.23**	**12.26 ± 6.06**
100	19.92 ± 1.24	12.94 ± 1.07	14.26 ± 1.41	16.40 ± 0.84	19.30 ± 5.68
1000	57.61 ± 2.35	**10.56 ± 0.15**	**11.51 ± 0.30**	*12.15 ± 4.34*	*12.53 ± 5.96*

Weight 100 reaches the best synthetic accuracy and the best MVTec median, but its MVTec-mean variance across seeds is by far the largest (±11.44°), driven by a single outlier replicate (39.5° against 19.5° and 19.8° for the other two seeds). Weight 10—the value used throughout this paper—has the lowest MVTec-mean variance of any tested weight (±0.62°) and the best MVTec mean overall, while remaining close to weight 100 and weight 1000's synthetic-error range rather than the far worse weights 0.1 and 1. Every configuration matched all six MVTec poses regardless of weight. We therefore retain weight 10 because it best balances competitive accuracy against seed-to-seed reliability on real images, rather than because it uniquely minimizes any single synthetic metric.

To test the same choice on a symmetric part, we repeated the ablation on the connector plate, whose two undirected axes both use the outer-product target (12 values), with the same weights and three seeds (lower part of Table 5). The synthetic error again decreases steadily with the weight, from 30.73° at weight 0.1 to 10.56° at weight 1000. On the five real instances, the differences between the weights are of the same size as the variation between

seeds. The lowest MVTec errors, at weights 1 and 1000, come from runs in which one seed missed one or two instances, whereas weight 10 matched all five instances for every seed, with a mean error of $13.25 \pm 1.23^\circ$. The connector plate therefore supports the choice of weight 10 on the same grounds as the T-connector.

5.4. Ablation of the Composition Loader

To assess the on-the-fly composition loader, we compared it with a Mosaic-like composition, our own implementation of the Mosaic augmentation of the YOLO family [30], in which four whole training images, each with its own background, are placed around a random center. Unlike the original, the center is drawn so that every part stays whole, the canvas is not cropped, and no horizontal flip is applied, because flipping mirrors the part and therefore changes its rotation label. All other settings are those of the YOLO26-Medium T-connector runs in Table 4 (100 px, same evaluation), and the comparison uses their seeds 0–4. Table 6 reports the results.

Table 6. Composition-loader ablation: T-connector, YOLO26-Medium, 100 px, five seeds (mean $\pm$ sample standard deviation over the seeds), with the columns of Table 5. The synthetic errors of each row are measured on validation and test images composed by its own loader and are therefore not comparable between the rows. MVTec predictions are matched to ground truth by box IoU ($\geq$0.5); *italic* entries mark the row in which one of the 30 real instances was not detected.

Training Images	Best val. Loss	Synthetic val. Error (°)	Synthetic Test Error (°)	MVTec Mean Error (°)	MVTec Median Error (°)
Composition loader (ours)	13.20 ± 0.52	15.16 ± 1.59	18.88 ± 1.48	$\mathbf{20.0 \pm 4.2}$	$\mathbf{19.9 \pm 4.0}$
Mosaic-like (our implementation)	5.52 ± 0.61	7.73 ± 4.52	10.09 ± 5.90	*35.3 ± 4.8*	*31.0 ± 3.8*

On the real MVTec instances, the composition loader gives a mean error of $20.0 \pm 4.2^\circ$ against $35.3 \pm 4.8^\circ$ for the Mosaic-like composition, and it is better for every seed. On their own synthetic test images, the order is reversed: the Mosaic-like composition reaches $10.1 \pm 5.9^\circ$ and the loader $18.9 \pm 1.5^\circ$. It places whole, unoccluded parts on the backgrounds of the rendered images, so its training and test images are easier to fit, but the error grows by about 25° from its synthetic test images to the real scenes. The loader pastes cut-out, randomly rotated and rescaled parts, with partial occlusion, onto real photographs (Figure 1b), which is closer to the cluttered real scenes of Figures 2 and 3; its synthetic and real errors differ by only about 1°. The standard Ultralytics copy-paste augmentation was not used, because it mirrors the pasted instances by default, which inverts the handedness of the rotation label.

5.5. Comparison Against SARR on the Same Real ITODD Parts

Kriegler et al.'s [13] SARR representation, introduced as related work in Section 2.3, publicly releases predictions for the annotated MVTec ITODD validation scenes, which they also use for evaluation because only these have public ground-truth poses ([13], Section 4.3). These are the scenes of the quantitative evaluation in Section 4.5; the BOP19 test subset is not used here. Four of their object IDs are exactly the parts evaluated here—`bracket_planar` (5), `connector_plate` (11), `cylinder` (12), and `tee_connector` (26)—with 6, 5, 4, and 6 instances, matching our ground-truth counts exactly. We filtered their predictions to these four IDs and re-scored them using the AR_B construction below. The rotation-regression branch here takes grayscale input only (Section 4.3), and MVTec ITODD itself has no RGB imagery, only grayscale [11]; Kriegler et al. match this with a network they call SARR-Gray, which is therefore our same-modality comparison. We additionally report their depth-input SARR-Depth variants for context, since this work's model has no access to depth.

Kriegler et al. [13] report AR_B, the average recall under the BOP benchmark's VSD, MSSD, and MSPD scores, all of which require object position in addition to orientation (unlike their ARC metric's rotation-only cosine distance, which we do not reproduce here). Since this work predicts orientation only (Section 4.5), we report AR_B using ground-truth translation paired with the predicted rotation—the same convention Kriegler et al. use for all of their reported results, their own orientation-only variants included, as they state "we use ground-truth translation for all results that we report: the networks we trained ourselves as well as results from other methods" ([13], Section 4.3). The AR_B comparison below is therefore rotation-only for every method shown, not just this work's. Table 7 reports these values alongside SARR-Depth and SARR-Gray across their four scopes, on the four parts and instances described above.

Kriegler et al. train their networks at four scopes [13]: one network per object (*object*), one per symmetry class (*symmetry*), one for all ITODD objects (*dataset*), and one for all objects with an additional symmetry-classification task (*dataset**). The object scope corresponds to our setup of one model per part and is therefore the like-for-like comparison; in the other scopes, one network is shared by several objects. Their networks are CenterNet models with a HardNet backbone, trained on 50,000 synthetic physically based renderings, and they estimate the rotation from a 384×384 crop of a single object instance, cut out and masked with its ground-truth object mask ([13], Appendix C.1); SARR-Gray converts these crops to grayscale. Our model instead processes the whole rescaled scene image and has to detect and segment each part itself. This comparison is therefore between complete systems, not between rotation representations alone: the two pipelines differ in training data, backbone, input crop and resolution, and in whether the objects are given by their ground-truth masks or detected, so the differences in Table 7 cannot be attributed solely to the outer-product target. Re-scoring both with the same AR_B implementation removes differences in the metric code, but not the others.

Table 7. AR_B (%, BOP average recall under VSD, MSSD, and MSPD) on the 21 annotated ITODD validation instances of the four shared parts for this work and for Kriegler et al.'s [13] SARR-Depth and SARR-Gray networks across their four symmetry-resolution scopes. Detection counts (matches/GT) given in parentheses.

Modality	Scope	T-Connector	Cylinder	Planar Bracket	Connector Plate	Overall (N = 21)
SARR-Depth	object	*84.06% (6/6)*	*71.83% (4/4)*	25.83% (6/6)	**92.00% (5/5)**	66.98%
	symmetry	79.83% (6/6)	*71.83% (4/4)*	49.28% (6/6)	87.33% (5/5)	*71.36%*
	dataset	67.28% (6/6)	70.83% (4/4)	*68.33% (6/6)*	73.33% (5/5)	69.70%
	dataset*	**84.11% (6/6)**	*71.83% (4/4)*	38.89% (6/6)	*90.00% (5/5)*	70.25%
SARR-Gray	object	61.78% (6/6)	53.75% (4/4)	1.89% (6/6)	12.93% (5/5)	31.51%
	symmetry	3.33% (6/6)	71.67% (4/4)	12.06% (6/6)	35.27% (5/5)	26.45%
	dataset	1.17% (6/6)	9.25% (4/4)	2.78% (6/6)	16.53% (5/5)	6.83%
	dataset*	0.22% (6/6)	0.00% (4/4)	4.22% (6/6)	8.00% (5/5)	3.17%
Ours (grayscale)	–	73.78% (6/6)	**97.25% (4/4)**	**79.78% (6/6)**	82.00% (5/5)	**81.92%**

Bold marks the best result and *italic* the second-best in each column (ties at a rank are all marked). SARR-Depth and SARR-Gray values are scored on the same four parts and instances described above using the BOP AR_B metric. This work's AR_B uses the ground-truth translation paired with the predicted rotation, since the rotation-regression branch evaluated here does not predict translation. Overall is the N-weighted average across the four parts.

Against SARR-Gray, the same-modality comparison, this system scores higher by a large margin on every part. In the object scope, SARR-Gray reaches 61.78% (T-connector), 53.75% (cylinder), 1.89% (planar bracket), and 12.93% (connector plate), against this work's 73.78%, 97.25%, 79.78%, and 82.00% on the same four parts; overall, this work's 81.92% more than doubles SARR-Gray's 31.51%, which is also its best scope. SARR-Gray's scores

are also highly inconsistent across its four symmetry-resolution scopes for the same part—for instance, T-connector ranges from 61.78% (object) down to 0.22% (dataset*)—whereas this work reports a single number per part with no comparable scope choice to make.

Against SARR-Depth, which additionally has access to depth input this work does not, the comparison in the object scope is mixed: this work is better on the cylinder (97.25% vs. 71.83%) and the planar bracket (79.78% vs. 25.83%) but trails on the T-connector (73.78% vs. 84.06%) and the connector plate (82.00% vs. 92.00%). Overall, this work's 81.92% exceeds SARR-Depth's 66.98% in the object scope and also its best scope (71.36%, symmetry). Since all methods here, including SARR-Depth and SARR-Gray, are scored with the same ground-truth translation [13, Sec. 4.3], these AR_B differences reflect rotation quality, with no localization advantage or disadvantage for either side, although rotation quality here still depends on the whole pipeline and not only on the representation; the pattern resembles that of Table 3, where the advantage of the outer-product target over direct regression is largest for the axially symmetric cylinder, but this system-level comparison alone cannot attribute the differences to SARR to the rotation representation.

5.6. Limitations

Several factors qualify the results above. First, the real-world comparisons in Tables 3, 4 and 7 rest on a small number of annotated MVTec ITODD instances per part (4–6, 21 total). Second, all the training data were synthetic, rendered with domain-randomized textures, backgrounds, and illumination; while this transferred well to real photographs for the four parts and symmetry types tested here, synthetic-to-real generalization is demonstrated only within that evaluated scope. The selected CAD parts, renderer, and randomization ranges define the training distribution and may omit real materials, lighting conditions, or backgrounds, creating biases that the small real-image evaluation cannot fully characterize. Third, the predicted outer-product matrix is not constrained to be exactly rank-one and positive semidefinite, and axis recovery assumes a clear dominant eigengap. No decoding failure occurred in the held-out evaluation, but this verifies conditioning only empirically; noisy or out-of-distribution predictions may still require validation and rejection before decoding. Fourth, this work's representation resolves the discrete sign ambiguity introduced by mirror-plane and axial symmetries specifically; it does not address continuous symmetries beyond an ignorable axis, higher-order discrete symmetries beyond the two-fold and box cases considered (Section 3.2), or pose ambiguity arising from occlusion or self-similarity rather than object geometry. Although the outer-product target itself does not depend on occlusion state, the predicted orientation may degrade when occlusion covers visual details learned for orientation inference. Training compositions included partially occluded instances, although instances with less than 30% of their original visible area were excluded as targets. In the qualitative BOP19 results (Section 5.2), no large coordinate-system error was apparent for detected partially occluded parts; the observed failures were instead instances with no detection or segmentation and hence no orientation estimate. Because these images lack public ground-truth poses, orientation error cannot yet be measured as a function of occlusion severity, and heavy occlusion was not evaluated quantitatively. Fifth, the outer-product target is compared with a signed target trained with a plain element-wise loss; symmetry-aware losses on signed targets, such as a minimum over the symmetric equivalents of the target or the 6D representation of Zhou et al. [4] with a symmetry-aware chordal loss, were not evaluated. The results therefore do not separate the contribution of the outer product itself from that of a self-consistent training target. Whether the outer-product target keeps an advantage over such a loss with the same grayscale input and detection pipeline therefore also remains open.

6. Conclusions

The sign-invariant outer-product representation used in this work as the regression target for symmetric axes follows Saxena et al. [7]. The central contribution of this work is the construction of that target for each symmetry case with fewer values, a signed vector for each directed axis, and an outer product for each sign-ambiguous axis, with the third axis obtained by a cross-product (6, 9, or 12 values instead of 9 to 39); the integration of that representation into a single-stage YOLO26 detection-segmentation architecture as an added rotation-regression branch, with one architecture and decoding procedure for four axis-symmetry cases (one model trained per part); and its experimental evaluation against direct, signed regression under synthetic-to-real transfer. Rather than regressing a symmetric axis directly as a signed vector, which forces the network to fit an arbitrary, inconsistent sign choice for physically identical poses, the network instead regresses the axis's outer product with itself, a target that is invariant to that sign ambiguity by construction and can be decoded back into an axis, up to sign, through an eigendecomposition.

For the primary YOLO26-Medium experiments, this representation was evaluated against direct, signed regression on three symmetric parts spanning the mirror-plane and axial symmetry cases considered in this work, and consistently improved orientation accuracy on both synthetic validation images and real MVTec ITODD photographs. The gains measured in simulation also transferred to the real images: over ten seeds, the mean MVTec angular error dropped from 58.4° to 5.5° for the cylinder, from 30.2° to 11.1° for the planar bracket, and from 37.6° to 15.4° for the connector plate—roughly a ten-fold reduction for the cylinder and more than a two-fold reduction for the other two parts—and the outer-product model was better in all ten seeds for each part (Wilcoxon signed-rank test, $p = 0.002$; Table 4). The improvement was not limited to the average case: the largest instance error fell just as sharply (e.g., 74.5° $\rightarrow$ 8.6° for the cylinder, averaged over the seeds). These results are consistent with the representation reducing a source of training inconsistency rather than only shifting the mean.

The YOLO26-Large confirmation reproduced the direction of the outer-product improvement on all three symmetric parts, with all 36/36 model-evaluation detections matched. This supports the effect at a second model capacity, although YOLO26-Large was not uniformly better than YOLO26-Medium. In the T-connector loss-weight ablation, weight 10 was the most stable tested setting: it achieved the best MVTec mean and the lowest across-run variation in that mean, while weight 100 gave the best synthetic accuracy and MVTec median.

On the same four real ITODD parts, this work's grayscale-only model was also compared against Kriegler et al.'s SARR networks (Section 5.5): in their object scope, which, like this work, trains one network per part, its overall BOP AR_B of 81.92% more than doubled that of the same-modality SARR-Gray (31.51%) and exceeded that of the depth-input SARR-Depth (66.98%, and 71.36% in its best scope), despite this work having no access to depth. Because the two pipelines also differ in training data, backbone, input crops, and resolution, this is a comparison of complete systems and does not by itself isolate the effect of the rotation representation.

Section 5.6 qualifies these results: chiefly, the small real-image sample; synthetic-only training and associated rendering biases; the scope of the representation (sign ambiguity only); qualitative-only evidence under occlusion; and the missing comparison with symmetry-aware losses on signed targets.

Taken together, these results indicate that resolving sign ambiguity at the level of the regression target, rather than through a separate keypoint-matching or point-cloud stage, is viable in a single-image detector: the tested system reached AR_B scores comparable to those of a depth-based, symmetry-specialized system while training only on domain-randomized

synthetic data, although this system-level result does not by itself show which part of the pipeline is responsible. This is a practically relevant outcome for grasping and part-feeding pipelines where annotated real orientation data is expensive to collect and a grayscale-only camera may be all that is available.

Future work should investigate whether the outer-product formulation can be extended to higher-dimensional invariant representations for higher-order discrete symmetries beyond sign flips (including cubic and other regular box-like parts with higher-order rotational symmetry), and how analogous invariant representations could treat continuous-symmetry structures beyond the ignorable-axis case considered here. It should also derive and test theoretically grounded relative loss weights between diagonal and off-diagonal entries of an outer-product target and, for mixed-symmetry targets, between directly regressed axis-vector values and outer-product matrix values. It should also compare the outer-product target with the SARR representation of Kriegler et al. [13] in a controlled setting, with both targets trained in the same detection pipeline on the same data. It should likewise compare the outer-product target with symmetry-aware losses on signed targets, such as an orbit-minimum loss and the 6D representation of Zhou et al. [4] with a symmetry-aware chordal loss, to separate the effect of the target from that of a self-consistent loss. It should also compare the unconstrained outer-product output with a normalized or constrained projector, for example, a predicted vector $\mathbf{v}$ mapped to $\mathbf{v}\mathbf{v}^T/\|\mathbf{v}\|^2$ before the loss. Future work should investigate the remaining real-world errors to determine whether they stem from difficult poses, imperfect axis recovery, or the synthetic-to-real domain gap, and evaluate robustness under occlusion, which this work's real-image evaluation does not isolate. Because the qualitative failures on partially occluded parts occurred mainly when detection or segmentation failed, a natural direction for further investigation is occlusion-focused training and augmentation to improve those stages before assessing orientation on successfully localized parts. The image scale for the real photographs was chosen here per part by a simple search over detection confidence; efficient scale-selection methods, for example, multi-scale inference or estimating the scale from the detections themselves, are left for future work.

The standard YOLO26 segmentation checkpoint used here is initialized from COCO-pretrained RGB weights and receives a three-channel grayscale image; it is not an RGB-D model and no sensor-depth information was used. YOLO26 also provides separate task-specific monocular-depth models [8], but those estimate depth from RGB rather than supplying sensor depth to this segmentation-and-rotation model. Future work should test whether normalized sensor depth, for example, used in place of one of the repeated grayscale color channels, improves rotation accuracy relative to RGB/grayscale alone.

Supplementary Materials: The following supporting information can be downloaded at https://www.mdpi.com/article/10.3390/math14214014/s1, Trained YOLO26-Medium planar-bracket model with the outer-product target, MVTec ITODD download script, evaluation script, and SARR re-scoring script (ZIP archive).

Author Contributions: Conceptualization, R.P. and A.H.; methodology, R.P.; software, R.P.; validation, R.P.; formal analysis, R.P.; investigation, R.P.; resources, A.H.; data curation, R.P.; writing—original draft preparation, R.P.; writing—review and editing, A.H.; visualization, R.P.; supervision, A.H.; project administration, A.H.; funding acquisition, A.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partly funded by the Slovenian Research Agency (ARIS) under a Grant for the Research Program P2–0028.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: This study analyzed publicly available datasets. The MVTec ITODD images and ground-truth pose annotations used for the real-image evaluation [11] are available at https://www.mvtec.com/research-teaching/datasets/mvtec-itodd (accessed on 28 October 2026) and, in the BOP benchmark format, at https://bop.felk.cvut.cz/datasets/ (accessed on 28 October 2026). The background images used for synthetic image composition were taken from the 2017 validation split of the Microsoft COCO dataset [27], available at https://cocodataset.org (accessed on 28 October 2026). The SARR predictions re-scored in Section 5.5 were publicly released by Kriegler et al. [13]. The Supplementary Materials contain one trained model, the YOLO26-Medium planar-bracket model with the outer-product target from Table 3, with its weights stored in half precision; a script that downloads the MVTec ITODD validation images of this part from the BOP benchmark; the evaluation script that recomputes its real-image result; and the script that re-scores the SARR predictions for Table 7. The other trained model weights, the synthetic training and validation images, and the source code generated in this study are available from the corresponding author upon reasonable request.

Acknowledgments: During the preparation of this manuscript, the authors used Claude (Anthropic; Claude Sonnet 5 and Claude Opus 5 models), accessed via the Claude Code command-line assistant, for the purposes of language editing and revision of the manuscript text, as well as LaTeX formatting and restructuring of tables, figures, and cross-references. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zucker, I.; Giladi, C. Predicting Part Orientation Distributions in Linear Feeders Using Simulation-Driven Deep Learning. *Machines* **2026**, *14*, 329. https://doi.org/10.3390/machines14030329.
2. Zhang, Y.; Liu, Y.; Zhang, Y.; Wang, W.; Wang, X. 6D Pose Estimation of Industrial Parts Based on Point Cloud Geometric Information Prediction for Robotic Grasping. *Entropy* **2024**, *26*, 1022. https://doi.org/10.3390/e26121022.
3. He, Y.; Wang, S.; Wu, C.; Chen, L. PGA6D: 6D Pose Estimation for Grasping and Assemblying Based on Keypoints Voting. *Mach. Vis. Appl.* **2026**, *37*, 11. https://doi.org/10.1007/s00138-025-01771-z.
4. Zhou, Y.; Barnes, C.; Lu, J.; Yang, J.; Li, H. On the Continuity of Rotation Representations in Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 5745–5753. https://doi.org/10.1109/CVPR.2019.00589.
5. Geist, A.R.; Frey, J.; Zobro, M.; Levina, A.; Martius, G. Learning with 3D Rotations, a Hitchhiker's Guide to SO(3). In Proceedings of the 41st International Conference on Machine Learning (ICML), Vienna, Austria, 21–27 July 2024.
6. Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; Abbeel, P. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 23–30.
7. Saxena, A.; Driemeyer, J.; Ng, A.Y. Learning 3-D Object Orientation from Images. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan, 12–17 May 2009; pp. 794–800. https://doi.org/10.1109/ROBOT.2009.5152855.
8. Ultralytics. Ultralytics YOLO26. Available online: https://docs.ultralytics.com/models/yolo26/ (accessed on 28 October 2026).
9. Jocher, G.; Qiu, J.; Liu, M.; Lyu, S.; Akyon, F.C.; Kalfaoglu, M.E. Ultralytics YOLO26: Unified Real-Time End-to-End Vision Models. *arXiv* **2026**. https://doi.org/10.48550/arXiv.2606.03748.
10. Todorov, E.; Erez, T.; Tassa, Y. MuJoCo: A Physics Engine for Model-Based Control. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Algarve, Portugal, 7–12 October 2012; pp. 5026–5033. https://doi.org/10.1109/IROS.2012.6386109.
11. Drost, B.; Ulrich, M.; Bergmann, P.; Hartinger, P.; Steger, C. Introducing MVTec ITODD: A Dataset for 3D Object Recognition in Industry. In Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW), Venice, Italy, 22–29 October 2017.
12. BOP Benchmark. BOP Benchmark: Model-Based 6D Localization of Seen Objects–ITODD. Available online: https://bop.felk.cvut.cz/leaderboards/pose-estimation-bop19/itodd/ (accessed on 28 October 2026).
13. Kriegler, A.; Beleznai, C.; Gelautz, M. Towards Symmetry-sensitive Pose Estimation: A Rotation Representation for Symmetric Object Classes. *Int. J. Comput. Vis.* **2026**, *134*, 212. https://doi.org/10.1007/s11263-026-02770-x.

14. Yang, Y.; Cui, Z.; Zhang, Q.; Liu, J. PS6D: Point Cloud Based Symmetry-Aware 6D Object Pose Estimation in Robot Bin-Picking. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Abu Dhabi, United Arab Emirates, 14–18 October 2024; pp. 7167–7174. https://doi.org/10.1109/IROS58592.2024.10802046.
15. Huang, D.T.; Lin, E.T.; Chen, L.; Liu, L.F.; Zeng, L. SD-Net: Symmetric-Aware Keypoint Prediction and Domain Adaptation for 6D Pose Estimation in Bin-Picking Scenarios. *arXiv* **2024**. https://doi.org/10.48550/arXiv.2403.09317.
16. Duffhauss, F.; Koch, S.; Ziesche, H.; Vien, N.A.; Neumann, G. SyMFM6D: Symmetry-Aware Multi-Directional Fusion for Multi-View 6D Object Pose Estimation. *IEEE Robot. Autom. Lett.* **2023**, *8*, 5315–5322. https://doi.org/10.1109/LRA.2023.3293317.
17. Zhao, H.; Wei, S.; Shi, D.; Tan, W.; Li, Z.; Ren, Y.; Wei, X.; Yang, Y.; Pu, S. Learning Symmetry-Aware Geometry Correspondences for 6D Object Pose Estimation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 1–6 October 2023; pp. 14045–14054.
18. Hodan, T.; Barath, D.; Matas, J. EPOS: Estimating 6D Pose of Objects with Symmetries. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020.
19. Haugaard, R.L.; Buch, A.G. SurfEmb: Dense and Continuous Correspondence Distributions for Object Pose Estimation with Learnt Surface Embeddings. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 19–24 June 2022; pp. 6739–6748.
20. Mo, N.; Gan, W.; Yokoya, N.; Chen, S. ES6D: A Computation Efficient and Symmetry-Aware 6D Pose Regression Framework. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 19–24 June 2022; pp. 6708–6717.
21. Wang, G.; Manhardt, F.; Tombari, F.; Ji, X. GDR-Net: Geometry-Guided Direct Regression Network for Monocular 6D Object Pose Estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021.
22. Lin, Y.; Su, Y.; Inuganti, S.; Di, Y.; Ajilforoushan, N.; Yang, H.; Zhang, Y.; Rambach, J. Resolving Symmetry Ambiguity in Correspondence-based Methods for Instance-level Object Pose Estimation. *arXiv* **2024**. https://doi.org/10.48550/arXiv.2405.10557.
23. Cai, D.; Heikkilä, J.; Rahtu, E. SC6D: Symmetry-agnostic and Correspondence-free 6D Object Pose Estimation. In Proceedings of the 2022 International Conference on 3D Vision (3DV), Prague, Czech Republic, 12–16 September 2022; pp. 536–546. https://doi.org/10.1109/3DV57658.2022.00065.
24. Cheng, P.; Yu, H.; Chen, W.; Tang, X.; Liu, J.; Akhtar, N. Symmetry-Aware 9D Pose Estimation with Sim(3)-Consistent Feature and Spherical Inception Convolution. *arXiv* **2026**. https://doi.org/10.48550/arXiv.2606.02219.
25. Tekin, B.; Sinha, S.N.; Fua, P. Real-Time Seamless Single Shot 6D Object Pose Prediction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 292–301. https://doi.org/10.1109/CVPR.2018.00038.
26. Shoemake, K. Uniform Random Rotations. In *Graphics Gems III (IBM Version)*; Kirk, D., Ed.; Morgan Kaufmann: San Francisco, CA, USA, 1992; pp. 124–132. https://doi.org/10.1016/B978-0-08-050755-2.50036-1.
27. Lin, T.Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C.L.; Dollár, P. Microsoft COCO: Common Objects in Context. In Proceedings of the European Conference on Computer Vision (ECCV), Zurich, Switzerland, 6–12 September 2014; pp. 740–755. https://doi.org/10.1007/978-3-319-10602-1_48.
28. Huynh, D.Q. Metrics for 3D Rotations: Comparison and Analysis. *J. Math. Imaging Vis.* **2009**, *35*, 155–164. https://doi.org/10.1007/s10851-009-0161-2.
29. Kisantal, M.; Wojna, Z.; Murawski, J.; Naruniec, J.; Cho, K. Augmentation for Small Object Detection. *arXiv* **2019**. https://doi.org/10.48550/arXiv.1902.07296.
30. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934. https://doi.org/10.48550/arXiv.2004.10934.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.