

SO6D: Symmetry-aware Octree Encoding for Textureless Objects' 6D Pose Estimation

Chenrui Wu

wuchenrui@usst.edu.cn

University of Shanghai for Science and Technology

Xinyu Shi

University of Shanghai for Science and Technology

Yao Huang

University of Shanghai for Science and Technology

Research Article

Keywords: Symmetric object, 6D pose estimation, Octree encoding, Hough voting, squares fitting

Posted Date: October 15th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-8317802/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

SO6D: Symmetry-aware Octree Encoding for Textureless Objects' 6D Pose Estimation

Chenrui Wu, Xinyu Shi and Yao Huang

Department of Mechanical Engineering, University of Shanghai for Science and Technology, JunGong Road, Shanghai, 200003, China.

*Corresponding author(s). E-mail(s): wuchenrui@usst.edu.cn;
Contributing authors: 232191431@st.usst.edu.cn; yhuang@usst.edu.cn;

Abstract

The symmetrical properties in industrial components causes challenges in computer vision applications. While current dense feature matching methods excel in 6D pose estimation for asymmetric objects, they struggle with symmetric ones. This paper presents a novel approach combining octree encoding and Hough voting for robust 6D pose estimation of symmetric objects. Our method applies symmetric-aware octree encoding to object vertices to ensure vertices with consistent features have the same encoding. A deep neural network is trained to learn the encoding for each vertex. During inference stage, Hough voting is leveraged to predict the object's center based on the predicted encodings. Then linear least square regression is used to obtain the rotational axis or symmetric plane of the object. The final pose is refined through a symmetric-aware iterative closest point (ICP) algorithm. Experiments on YCB-V dataset and T-LESS dataset demonstrate our approach's superior performance, achieving state-of-the-art results in pose accuracy.

Keywords: Symmetric object, 6D pose estimation, Octree encoding, Hough voting, squares fitting

1 Introduction

Pose estimation is fundamental for smart manufacturing applications including precision assembly, intelligent sorting, and automated stacking. Determining an object's complete 6D pose consists of three-dimensional position and three-dimensional orientation, and this is a core challenge in computer vision and robotics, particularly under variable illumination, partial occlusion, and complex arrangements.

Deep learning methods for the estimation of the 6D pose can be classified by sensor input: RGB methods[1–5] are cost-effective but lack depth information, while RGB-D methods[6–11] leverage 2.5D data for superior performance. As depth

sensors become increasingly affordable, this paper focuses on 2.5D input-based approaches.

Existing 2.5D methods broadly fall into three categories. **Direct regression** methods train networks to predict rotation and translation directly. However, object symmetries create one-to-many mappings between images and valid poses. ES6D[12] addresses this by analyzing the symmetry classes and introducing symmetry-sensitive distance metrics, while Cosypose[4] minimizes distance to the closest symmetric candidate. Probabilistic approaches model rotation distributions through normalizing flows[13] or implicit representations[14]. Recent diffusion-based methods[15–17] use multimodal modeling

capabilities to generate poses, although they still struggle with occlusion. **Dense voting** methods such as PVNet[1], PVN3D[7], and FFB6D[8] predict the direction or distance from each visible point to the center or key points of the object. Although object centers are symmetry-invariant, keypoint distributions exhibit ambiguity under symmetry. GDRNet[5] partitions the models into patches that vote for local positions before predicting the centers, but cannot overcome symmetry-induced ambiguities. These inconsistent supervision signals hinder convergence and accuracy, particularly for occluded objects, where voting errors cause pose drift.

Dense matching methods learn the correspondences between point clouds and model points. SurfEmb[18] and GCPose[11] learn implicit mappings via point-wise similarity, while ZebraPose[2] and HiPose[9] assign explicit binary codes to vertices, achieving state-of-the-art results on non-symmetric objects. However, unique-vertex encoding conflicts with symmetric object ambiguity, degrading performance. Although less affected by occlusion than voting methods, similar features in symmetric regions introduce interference during matching, requiring robust outlier rejection like RANSAC.

For pose refinement, ICP[19] aligns estimated and scene point clouds but suffers from local minima due to the nearest-neighbor correspondence. PFA[20] predicts dense correspondence fields non-iteratively, though it remains ambiguous for symmetric objects and requires additional networks, increasing complexity.

In this paper, we propose a novel symmetric-aware octree encoding approach for dense matching. By analyzing object symmetries—whether axisymmetric or planar symmetric—we encode 3D model vertices using octree structures ensuring that identical features receive identical encodings, enabling networks to learn without ambiguity. During inference, we estimate object centers via 3D Hough voting, compute symmetry planes or normal vectors through linear least squares, and refine poses using symmetric-aware ICP.

Our main contributions are: (1) A symmetric-aware octree encoding method ensuring convergence for symmetric objects by assigning consistent encodings to identical features. (2) A hybrid post-processing pipeline that combines dense voting and matching advantages—using 3D Hough

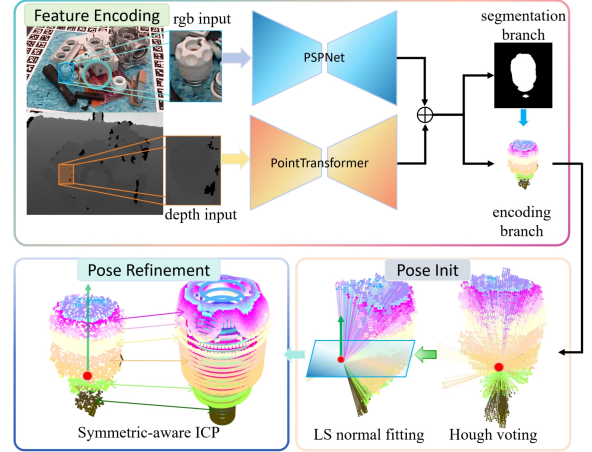


Fig. 1 Pipeline of SO6D consists of three main components: point cloud encoding, initial pose estimation, and pose optimization. The implementation details are introduced in Sections 2.3, 2.4, and 2.5 respectively.

voting for center estimation, linear least squares for symmetry plane computation, and symmetric-aware ICP for refinement. (3) Comprehensive evaluation of T-LESS and YCB-V datasets demonstrating superiority over existing approaches, with detailed analysis of network architectures, encoding depths, and computational efficiency.

2 Method

In this section, we provide a detailed description of our proposed method. Inspired by other dense feature point matching methods[9], we propose a symmetry-aware octree encoding method. This encoding has the advantage of simultaneously possessing feature matching capabilities and positional representation capabilities, allowing us to adopt both dense voting and dense matching approaches to achieve more accurate pose estimation.

2.1 Problem Definition

For each symmetric object appearing in an RGB-D image, our goal is to estimate its own coordinate system relative to the camera coordinate system’s translation and rotation. Translation can be represented as $t \in \mathbb{R}^3$, and rotation can be represented as $R \in SO(3)$. Suppose that the target object’s model $\mathcal{M}$ is given, which contains n vertices $v_i \in \mathbb{R}^3$, where i is the index. We need to encode each vertex using a given algorithm and then train a

neural network in a supervised manner, with the input being the RGB-D image and the output being the encoded value corresponding to each point in the image. This encoded value represents the correspondence between each point p_j in the RGB-D image and the vertex v_i in the model. According to the correspondence, we can calculate the relative pose using a method such as least squares:

$$\min_{R \in SO(3), t \in \mathbb{R}^3} \sum_{(i,j) \in \mathcal{C}} \|p_j - (Rv_i + t)\|^2, \quad (1)$$

where $\mathcal{C}$ denotes the set of corresponding point pairs, and $(i, j) \in \mathcal{C}$ indicates that the i -th model vertex v_i corresponds to the j -th image point p_j . Calculating (1) requires a prerequisite condition, namely that $\mathcal{C}$ is a correspondence of one-to-one. However, symmetric objects clearly do not satisfy this requirement, which is the main reason why current dense matching methods[2, 9, 11] perform well on non-symmetric objects but fail to handle symmetric objects.

2.2 Pipeline of SO6D

The pipeline of the proposed method is shown in Fig.1. As in [2, 9], we first detect the object and crop the corresponding RGB-D region, which is then encoded with an octree-based representation explicitly designed to handle object symmetries (Section 2.3). During supervised training, ground-truth poses and segmentation masks are used to assign octree encodings to object points in the RGB-D image I_{rgb-d} . The image is processed by a network $\mathcal{F} = \Phi_{fe}(I_{rgb-d})$ that combines a Point Transformer and PSPNet for the extraction of features, although these backbones are modular and can be replaced by more advanced alternatives. The fused features are passed to two parallel heads: a segmentation branch Φ_{seg} for the foreground/background prediction and an encoding branch Φ_e for the prediction of the octree code. At inference time, the segmentation head first selects object points, whose predicted octree encodings are then used for pose estimation. An initial pose T_{init} is computed from these encodings in a symmetry-sensitive manner (Section 2.4), and the final pose is obtained through an iterative refinement procedure described in Section 2.5.

2.3 Symmetry-Aware Octree Encoding

HiPose[9] used k-NN to recursively bisect the points on $\mathcal{M}$, without considering the impact of object symmetry. Additionally, the encoding information of the vertices themselves is not correlated with their 3D coordinates. Taking into account these two points, we propose to use an octree to encode the vertices. We primarily consider two types of symmetry: axis symmetry and plane symmetry, as shown in Fig. 2. Our core objective is to ensure that the symmetric vertices share identical feature representations. For axis-symmetric models, assume that the symmetry axis of model $\mathcal{M}$ is the z-axis. Let $v_i = (x_i, y_i, z_i) \in \mathbb{R}^3$ be a vertex in the Cartesian coordinates of the object. Let $\Pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ be the transformation that maps a vertex (x_i, y_i, z_i) to its cylindrical projection (ρ_i, z_i) , where $\rho_i = \sqrt{x_i^2 + y_i^2}$ represents the radial distance from the z-axis. Then we encode the coordinates using a binary octree method $E : \mathbb{R}^2 \rightarrow \mathbb{N}^2$, where the encoding functions for the ρ and z coordinates are defined as:

$$E_\rho(\rho) = \sum_{l=1}^L b_l^\rho \cdot 2^{L-l}, \quad b_l^\rho \in \{0, 1\} \quad (2)$$

$$E_z(z) = \sum_{l=1}^L b_l^z \cdot 2^{L-l}, \quad b_l^z \in \{0, 1\}, \quad (3)$$

where L denotes the maximum depth of the octree, and at each level l , the binary digits b_l^ρ and b_l^z are determined by:

$$b_l^\rho = \mathbb{I}(\rho > \rho_{mid}), \quad b_l^z = \mathbb{I}(z > z_{mid}), \quad (4)$$

where $\mathbb{I}(\cdot)$ is the indicator function, and (ρ_{mid}, z_{mid}) represents the midpoints of the current subdivision at level l . The final encoding is given by the tuple $\mathcal{E}_a = (E_\rho(\rho), E_z(z))$.

For plane-symmetric objects, the vertices equidistant from the symmetry plane should share identical encoding. Considering a Euclidean coordinate system with the object's center as origin and a symmetry plane $z = 0$, a vertex $p_j = (x_j, y_j, z_j)$ and its symmetric counterpart $p'_j = (x_j, y_j, -z_j)$ must map to the same encoding. To achieve this, we transform the coordinate values taking the absolute value of the z-coordinate: $p_j = (x_j, y_j, |z_j|)$. This transformation is applied to all

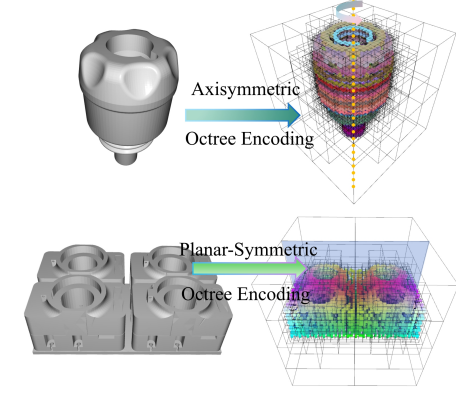


Fig. 2 Illustration of the proposed octree encoding method. Grid cells with the same color share identical encoding, which helps eliminate ambiguity issues during model training.

vertices in the model to maintain planar symmetry. This representation offers two key advantages: (1) It encodes the symmetric features of the model into identical forms. (2) The feature encoding vanishes at the coordinate plane, enabling direct fitting to determine the zero-value of the coordinate axis. The corresponding encode function is $E : \mathbb{R}^3 \rightarrow \mathbb{N}^3$, where the encode functions for the coordinates x , y , and $|z|$ are defined as:

$$E_d(d) = \sum_{l=1}^L b_l^d \cdot 2^{L-l}, \quad b_l^d \in \{0, 1\}, \quad d \in \{x, y, |z|\}, \quad (5)$$

where L denotes the maximum depth of the octree and $b_l^d = \mathbb{I}(d > d_{mid})$. $\mathbb{I}(\cdot)$ is the indicator function and $(x_{mid}, y_{mid}, z_{mid})$ represents the midpoints of the current subdivision at level l . The final encoding is given by the tuple $\mathcal{E}_p = (E_x(x), E_y(y), E_{|z|}(|z|))$. Following the computation of the octree encoding for each vertex v_i on the model $\mathcal{M}$, we transform the model into the camera coordinate system using the ground truth pose T_{gt} provided in the training data set. We then establish point correspondences between the transformed model vertices and points in the RGB-D image using nearest neighbor matching. These correspondences enable us to assign appropriate encodings to each point of the object p_i , which serve as the ground truth encodings $\mathcal{E}_{gt}^i$ during the network training process. The detailed pipeline of the octree encoding neural network is illustrated in Fig. 3.

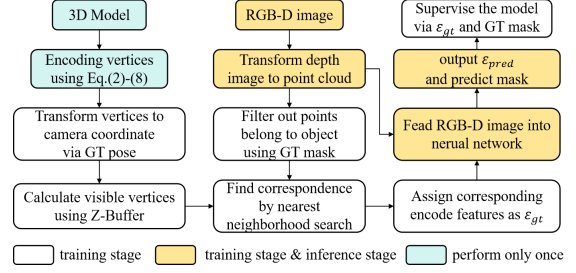


Fig. 3 Pipeline for the octree encoding network.

2.4 Pose Estimation based on Hough Voting

Since the octree encodings is many-to-many, rather than one-to-one, (1) cannot be directly used for pose estimation. Therefore, this section provides a voting method based on Hough to calculate the initial pose T_{init} . The octree encoding inherently preserves the spatial positions in the object coordinate. This positional information allows us to implement the voting-based method for estimating the object's center.

2.4.1 Pose Estimation of Axis-Symmetric Object

For objects with axial symmetry, T_{init} can be determined by identifying the center of the object and its axis of rotation. The predicted encodings $\mathcal{E}_{pred}$ are first converted to cylindrical coordinates $p_i = (\rho, z)$. Subsequently, for each point p_i , we compute its radial distance $r_i = |\vec{p}_i| = \sqrt{\rho^2 + z^2}$. Each point p_i casts votes in a discretized 3D voting space $\mathcal{G}$ based on its predicted radius r_i . The voting process accumulates evidence in $\mathcal{G}$ according to:

$$p_c = \underset{g \in \mathcal{G}}{\operatorname{argmax}} \sum_{i=1}^N H(g, p_i, r_i) \quad (6)$$

$$H(g, p_i, r_i) = \begin{cases} w_i & \text{if } ||g - p_i| - r_i| < \epsilon \\ 0 & \text{otherwise} \end{cases}, \quad (7)$$

where g represents the 3D grid in the voting space, ϵ is a voting tolerance threshold, and w_i is a confidence-based voting weight following a normal distribution:

$$w_i = \exp \left(-\frac{(|g - p_i| - r_i)^2}{2\sigma^2} \right). \quad (8)$$

The grid with the highest accumulation of votes in $\mathcal{G}$ is designated as the center of the object

p_c . The voting computation is implemented in CUDA to enable parallel processing, significantly reducing computational time.

Following the determination of p_c , the estimation of the rotational axis can be formulated as an optimization problem: given a plane L_c passing through point p_c and a set of points $p_i \in \mathbb{R}^3$ with their corresponding plane distances $d_{l,i} = z_i$, we seek to determine the optimal plane normal vector $\mathbf{n}_l$ that minimizes point-to-plane distance errors. The optimization objective function is formulated as follows.

$$\min_{\mathbf{n}_l} \sum_{i=1}^n \|(p_i - p_c)^T \mathbf{n}_l - z_i\|^2. \quad (9)$$

This optimization problem can be reformulated as a linear least squares system:

$$\min_{\mathbf{n}_l} \|\mathbf{A}\mathbf{n}_l - \mathbf{b}\|^2, \quad (10)$$

where

$$\mathbf{A} = \begin{bmatrix} (p_1 - p_c)^T \\ \vdots \\ (p_n - p_c)^T \end{bmatrix}, \mathbf{b} = \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix}. \quad (11)$$

The optimal unit normal vector is obtained through the following:

$$\mathbf{n}_l = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}. \quad (12)$$

Given the calculated center point p_c and the normal vector $\mathbf{n}_l$, T_{init} can be determined by geometric transformation.

2.4.2 Pose Estimation of Plane-Symmetric Object

The center computation procedure for plane-symmetric objects is methodologically equivalent to that utilized in axial-symmetric objects. Given the discrete nature of the planar symmetry characteristics defined by the symmetry plane, once the symmetry plane is determined, the feature points maintain one-to-one mapping relationships within the subspaces partitioned by this plane. Following the determination of the center p_c by Hough voting, the symmetry plane is computed using linear least squares optimization. Based on the

computed symmetry plane, we partition the point cloud into distinct subspaces and establish one-to-one correspondences between these points and the vertices within corresponding subspaces of the 3D model. The initial transformation matrix T_{init} is subsequently obtained through the optimization framework defined in (1).

2.5 Pose refinement

After obtaining T_{init} , we iteratively optimize the pose using symmetric-aware ICP optimization. In contrast to conventional ICP algorithms that perform exhaustive nearest neighbor searches across all vertices, our method exploits the octree encodings of points to constrain the search space to subsets of semantically corresponding vertex.

Based on the initial pose estimate T_{init} , we can assume that there exists a small transformation ΔT between the current pose and the final pose. For each point p_j in the point cloud, we first obtain its octree encoding $\mathcal{E}_a^j$, then locate all vertices in the model $\mathcal{M}$ that share the same encoding:

$$\mathcal{V}_p = \{v_i \in \mathcal{M} \mid \mathcal{E}_a(v_i) = \mathcal{E}_a^j\}. \quad (13)$$

Subsequently, we employ a nearest neighbor search to find the vertex v_{i^*} in $\mathcal{V}_p$ that is closest to p_j :

$$v_{i^*} = \arg \min_{v_i \in \mathcal{V}_p} \|T_{init} v_i - p_j\|^2. \quad (14)$$

Based on established correspondences, we compute the transformation matrix T_k that minimizes point-to-point distances at each iteration k :

$$T_k = \arg \min_T \sum_j \|T \cdot v_{i^*} - p_j\|^2. \quad (15)$$

This process is repeated iteratively until convergence or a maximum number of iterations is reached. The final transformation T_{final} is obtained from the last iteration $T_{final} = T_K$. This formulation directly solves for the optimal rigid transformation in each iteration using closed-form methods such as SVD or quaternion-based approaches, where K denotes the final iteration number. This approach effectively leverages the initial pose estimate and establishes reliable point correspondences while accounting for symmetry, thereby achieving robust pose optimization.

3 Experiments

In this section, we first introduce the implementation details of our method, the datasets used for testing, and the evaluation metrics. We then compare our approach with recent pose estimation methods. Finally, we analyze the contribution of each component in our proposed method and examine parameters such as runtime through ablation studies.

3.1 Implementation Details

The octree encoding model Φ_{fe} combines Point-TransformerV3 (PT3)[21] and PSPNet[22], where PT3 extracts point cloud features, and PSPNet extracts image features. Similarly to [2, 5], the network takes a zoomed-in Region of Interest (RoI) image and its corresponding depth map as input. The image is processed through PSPNet, while the point cloud converted from the depth map is fed into PT3. Their outputs are merged and transferred to a 2-layer MLP segmentation prediction network Φ_{seg} and a 2-layer MLP encoding network Φ_{encode} . We supervise both segmentation and encoding using binary cross-entropy loss. The final loss function is defined as:

$$\mathcal{L}_{total} = \mathcal{L}_{mask} + \alpha \mathcal{L}_{code}, \quad (16)$$

where α balances the two losses. We empirically set $\alpha = 0.1$ across all experiments. In particular, for the encoding loss, we only supervise points that belong to the target object during training. A separate model for each object is trained for 30 epochs with a batch size of 8. We use the AdamW optimizer with sinusoidal learning rate scheduling in one-cycle, starting at $1e^{-4}$, following the default PT3 settings.

3.2 Datasets

We evaluate our method on two benchmarks. The YCB-Video dataset contains recordings of 21 YCB objects, 5 of which are symmetric. It is challenging because of varying illumination, strong artifacts, and frequent occlusions. The dataset consists of 92 RGB-D videos showing different object combinations in indoor scenes; following the standard protocol, we use 80 videos for training and 2,949 keyframes from the remaining 12 videos for testing.

The T-LESS dataset is a particularly challenging benchmark with 27 symmetric and 3 asymmetric objects, making it well suited to assessing symmetry-aware methods. We follow previous work and use 10,000 real scan images for evaluation, adopting object detections provided by GDRNPP [5] for fair comparison.

3.3 Evaluation Metrics

For YCB-Video, we report the AUC of the ADD-S metric (using the symmetric variant of ADD) with a maximum distance threshold of 10 cm, following [8, 9]. For T-LESS, we follow the BOP benchmark and use MSPD (Maximum Symmetry-Aware Projection Distance), MSSD (Maximum Symmetry-Aware Surface Distance), and VSD (Visible Surface Discrepancy). To better capture symmetry-related performance, we additionally report pose precision at fixed rotation and translation thresholds: $R@2^\circ$, $R@5^\circ$, and $R@10^\circ$ denote the fraction of poses whose rotation error is below 2° , 5° , and 10° , respectively, while $T@2$ cm, $T@5$ cm, and $T@10$ cm measure the proportion of predictions within 2, 5 and 10 cm of the translation of the ground-truth.

3.4 Comparison to State of the Art

In the following, we compare our method to the state of the art using various metrics on YCB-V and T-LESS datasets.

3.4.1 Results on YCB-V

The results of the 6D pose estimation on symmetric objects in YCB-V are shown in Table. 1. The experimental results demonstrate that our method outperforms the HiPose dense matching approach by approximately 6%, effectively addressing the correspondence ambiguity problems inherent in one-to-one feature matching schemes. Compared to the second-best performing method, our approach achieves a performance improvement of 1.4%, primarily due to our symmetric-aware pose refinement strategy. This enhancement can be attributed to our feature matching mechanism, which inherently preserves axial and planar symmetry constraints, thus minimizing the occurrence of incorrect correspondences.

Table 1 Comparison of ADD-S score with different 6D pose estimation methods on symmetric objects in YCB-V(%)

	RGB		RGB-D			
	GDRNPP	ZebraPose	FFB6D	ES6D	HiPose	SO6D
bowl	85.7	37.1	96.3	96.4	84.2	96.8
wood	82.5	84.5	92.6	94.4	89.7	96.1
Lclamp	89.3	85.6	96.8	61.0	81.3	97.1
xl.clamp	93.5	92.5	96.0	59.6	94.3	97.5
foam	90.9	95.3	95.3	96.6	96.1	96.8
Avg.	88.4	79.0	95.4	81.6	91.1	96.8

3.4.2 Results on T-LESS

The comparative results of our method (SO6D) against other state-of-the-art approaches on the T-LESS dataset are presented in Table. 2, evaluated using multiple metrics, including MSPD, MSSD and VSD. Our method demonstrates superior performance across all evaluation metrics.

Specifically, in terms of MSPD, which measures the 2D projection accuracy, SO6D achieves 0.901, surpassing the previous best result of 0.892 by ES6D. For MSSD, which evaluates 3D pose accuracy, our method attains 0.875, showing a notable improvement over HiPose’s 0.866. The most significant enhancement is observed in the VSD metric, where SO6D achieves 0.807, substantially outperforming ES6D’s 0.751, representing a 5.6% improvement.

The Average Recall (AR) across all metrics further validates the effectiveness of our approach, with SO6D achieving 0.861, compared to 0.833 from HiPose, the second-best performer. This comprehensive improvement of 2.8% in AR demonstrates that our octree-based design effectively addresses the symmetry-induced mismatching issues prevalent in the T-LESS dataset.

Table 2 Comparison of MSPD, MSSD, VSD score with different 6D pose estimation methods on T-LESS

Metrics	ZebraPose	ES6D	HiPose	GCPose	SO6D
MSPD	0.889	0.892	0.888	0.757	0.901
MSSD	0.740	0.841	0.866	0.735	0.875
VSD	0.696	0.751	0.746	0.681	0.807
AR	0.775	0.828	0.833	0.724	0.861

Table 3 Comparison of $R@2$, $R@5$, $R@10$, $T@2cm$, $T@5cm$, and $T@10cm$ score with different 6D pose estimation methods on T-LESS(%)

Method	$R@2^\circ$	$R@5^\circ$	$R@10^\circ$	$T@2cm$	$T@5cm$	$T@10cm$
GDRNPP	21.60	71.18	90.56	90.31	96.09	98.10
DiffPose	47.21	86.94	94.78	71.72	92.03	97.15
SO6D	49.38	89.15	95.81	92.14	96.46	98.31

Table. 3 presents a comprehensive evaluation of translational and rotational accuracy metrics across multiple threshold values. The quantitative results demonstrate the efficacy of our proposed method in comparison with state-of-the-art approaches.

In terms of rotational precision, our method exhibits substantial improvements across all error thresholds. Notably, at the most stringent threshold of 2 degrees, our approach achieves 49.38% recall, surpassing GDRNPP and DiffPose by margins of 27.78% and 2.17%, respectively. This enhancement in fine-grained rotational accuracy demonstrates the robustness of our method in handling complex geometric orientations.

Translational accuracy metrics further validate the superiority of our approach. At the critical 2 cm threshold, our method achieves a recall of 92.14%, representing a substantial improvement over DiffPose (71.72%) and GDRNPP (90.31%). This increased precision in translational estimation is particularly significant for industrial applications that require high-precision manipulation.

Qualitative examples of pose estimation results are shown in Fig. 4. The proposed method demonstrates robust pose computation even under partial object occlusion. Failure cases occur primarily under severe occlusion conditions (when the visible surface area is less than 20% of the total surface of the object).

3.5 Ablation Studies

3.5.1 Submodule Analysis

In this section, we present a comparative analysis of symmetric-aware octree encoding versus conventional octree encoding, along with an evaluation of pose initialization using Hough voting and the commonly used RANSAC method. Pose refinement using symmetric-aware ICP and conventional ICP algorithm is also investigated on the YCB-V dataset.



Fig. 4 Qualitative results on T-LESS dataset.

Table 4 Ablation study on the main modules proposed in SO6D. s-a denotes symmetric-aware and conv. represents conventional.

Octree Encoding	s-a octree					
	conv. octree	✓	✓	✓	✓	✓
Pose Init	Hough voting					
	RANSAC	✓	✓	✓	✓	✓
Pose Refinement	s-a ICP					
	conv. ICP	✓	✓	✓	✓	✓
ADD-S (%)		72.5	87.6	90.4	96.0	96.8

Table. 4 presents a systematic evaluation of the components of our framework. The baseline approach, which uses conventional positional encoding with RANSAC, achieves an ADD-S score of 72.5 for symmetric objects. Incorporating our symmetric-aware octree encoding demonstrates marked improvement, elevating performance to levels comparable with the HiPose method. The integration of our proposed Hough voting mechanism further enhances accuracy, slightly exceeding the performance of the FFB6D benchmark. Finally, the addition of symmetric-aware ICP refinement culminates in a 1.4% performance improvement over FFB6D, validating

the effectiveness of our complete symmetric-aware pipeline.

3.5.2 Hyperparameter Tuning

The key hyperparameters that affect pose estimation accuracy are the depth of the octree encoding d and the radius error tolerance ϵ in the Hough voting. We empirically determined these hyperparameters through an ADD-S metric evaluation on the YCB-V dataset. As shown in Fig. 5(a), experiments with depth of octree encoding demonstrate optimal performance in $d = 6$. Lower depth values result in coarse point classification, compromising precise localization. In contrast, greater depths lead to training difficulties in the neural network, increasing overall training error, and subsequently degrading matching performance. For the error tolerance ϵ in the Hough vote, we evaluated values ranging from 0.1 to 0.4. Detailed experimental results are presented in Fig. 5(b).

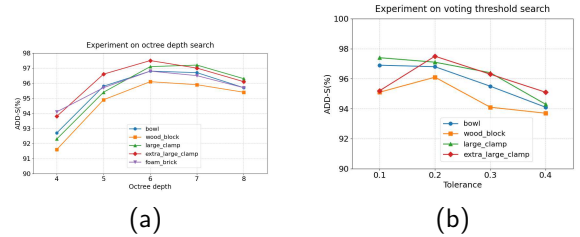


Fig. 5 (a) Octree encoding depth searching experiment. (b) Error tolerance analysis of Hough voting.

3.5.3 Runtime Analysis

We evaluated our method’s runtime performance on a single NVIDIA RTX 3090 GPU paired with an AMD Ryzen Threadripper 2950X CPU. The computational time is as follows: neural network prediction of octree encoding takes 52 ms, Hough voting for object center estimation requires 128 ms, symmetry plane or revolute object normal calculation consumes 74 ms, and symmetric-aware ICP computation takes 852 ms, totaling 1106 ms per inference. Among methods employing iterative ICP computations(as shown in Table. 5), our approach demonstrates relatively competitive computational efficiency.

Table 5 Time Efficiency Comparison of Different Methods

Method	SurfEmb	GDRNPP	ZebraPose	HiPose	FFB6D	SO6D w/o refinement	SO6D
Refinement	custom[18]	CIR[5]	ICP[2]	-	-	-	symmetric-aware ICP
Average Time (s)	9.04	6.37	1.79	0.16	0.19	0.25	1.1

4 Conclusion

In this paper, we present SO6D, a 6D pose estimation method that performs dense feature matching through symmetry-aware octree encoding. This approach addresses the ambiguity issues in the training of features for symmetric objects that current dense feature matching methods face by using explicit symmetry-aware feature representations. Our proposed method demonstrates significant advantages in pose estimation on symmetric object datasets compared to existing approaches. The main limitation of our method is the requirement to train specific models for individual objects. Future work will focus on developing generalizable encoding forms, aiming to achieve strong performance in the estimation of zero-shot 6D poses.

Declarations

Funding

The research leading to these results received funding from the National Natural Science Foundation of China under Grant Agreement No[52105525].

Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

References

- [1] Peng, S., Liu, Y., Huang, Q., Zhou, X., Bao, H.: Pvnnet: Pixel-wise voting network for 6dof pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4561–4570 (2019)
- [2] Su, Y., Saleh, M., Fetzner, T., Rambach, J., Navab, N., Busam, B., Stricker, D., Tombari, F.: ZebraPose: Coarse to fine surface encoding for 6dof object pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6738–6748 (2022)
- [3] Chen, H., Wang, P., Wang, F., Tian, W., Xiong, L., Li, H.: Epro-nnp: Generalized end-to-end probabilistic perspective-n-points for monocular object pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2781–2790 (2022)
- [4] Labbé, Y., Carpentier, J., Aubry, M., Sivic, J.: Cosypose: Consistent multi-view multi-object 6d pose estimation. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII 16, pp. 574–591 (2020). Springer
- [5] 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, pp. 16611–16621 (2021)
- [6] Bai, X., Luo, Z., Zhou, L., Chen, H., Li, L., Hu, Z., Fu, H., Tai, C.-L.: Pointdsc: Robust point cloud registration using deep spatial consistency. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 15859–15869 (2021)
- [7] He, Y., Sun, W., Huang, H., Liu, J., Fan, H., Sun, J.: Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11632–11641 (2020)
- [8] He, Y., Huang, H., Fan, H., Chen, Q., Sun,

- J.: Ffb6d: A full flow bidirectional fusion network for 6d pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3003–3013 (2021)
- [9] Lin, Y., Su, Y., Nathan, P., Inuganti, S., Di, Y., Sundermeyer, M., Manhardt, F., Stricker, D., Rambach, J., Zhang, Y.: Hipose: Hierarchical binary surface encoding and correspondence pruning for rgb-d 6dof object pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10148–10158 (2024)
- [10] Wen, B., Yang, W., Kautz, J., Birchfield, S.: Foundationpose: Unified 6d pose estimation and tracking of novel objects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 17868–17879 (2024)
- [11] 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, pp. 14045–14054 (2023)
- [12] 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, pp. 6718–6727 (2022)
- [13] Liu, Y., Liu, H., Yin, Y., Wang, Y., Chen, B., Wang, H.: Delving into discrete normalizing flows on $so(3)$ manifold for probabilistic rotation modeling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 21264–21273 (2023)
- [14] Höfer, T., Kiefer, B., Messmer, M., Zell, A.: Hyperposepdf-hypernetworks predicting the probability distribution on $so(3)$. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 2369–2379 (2023)
- [15] Hsiao, T.-C., Chen, H.-W., Yang, H.-K., Lee, C.-Y.: Confronting ambiguity in 6d object pose estimation via score-based diffusion on $se(3)$. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 352–362 (2024)
- [16] Tseng, H.-Y., Li, Q., Kim, C., Alsisan, S., Huang, J.-B., Kopf, J.: Consistent view synthesis with pose-guided diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16773–16783 (2023)
- [17] Zhang, J., Wu, M., Dong, H.: Generative category-level object pose estimation via diffusion models. *Advances in Neural Information Processing Systems* **36** (2024)
- [18] 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, pp. 6749–6758 (2022)
- [19] Zhang, J., Yao, Y., Deng, B.: Fast and robust iterative closest point. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **44**(7), 3450–3466 (2021)
- [20] Hu, Y., Fua, P., Salzmann, M.: Perspective flow aggregation for data-limited 6d object pose estimation. In: European Conference on Computer Vision, pp. 89–106 (2022). Springer
- [21] Wu, X., Jiang, L., Wang, P.-S., Liu, Z., Liu, X., Qiao, Y., Ouyang, W., He, T., Zhao, H.: Point transformer v3: Simpler faster stronger. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4840–4851 (2024)
- [22] Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2881–2890 (2017)