SEMPose: A Single End-to-end Network for Multi-object Pose Estimation

Xin Liu[a,1], Hao Wang[a], Shibei Xue[a,*] and Dezong Zhao[b]

[a]*Department of Automation, Shanghai Jiao Tong University, Shanghai, 200240, China*

[b]*James Watt School of Engineering, University of Glasgow, Glasgow, G12 8QQ, United Kingdom*

ARTICLE INFO

ABSTRACT

In computer vision, estimating the six-degree-of-freedom pose from an RGB image is a fundamental task. However, this task becomes highly challenging in multi-object scenes. Currently, the best methods typically employ an indirect strategy, which identifies 2D and 3D correspondences, and then solves with the Perspective-n-Points method. Yet, this approach cannot be trained end-to-end. Direct methods, on the other hand, suffer from lower accuracy due to challenges such as varying object sizes and occlusions. To address these issues, we propose SEMPose, an end-to-end multi-object pose estimation network. SEMPose utilizes a well-designed texture-shape guided feature pyramid network, effectively tackling the challenge of object size variations. Additionally, it employs an iterative refinement head structure, progressively regressing rotation and translation separately to enhance estimation accuracy. During training, we alleviate the impact of occlusion by selecting positive samples from visible parts. Experimental results demonstrate that SEMPose can perform inference at 32 FPS without requiring inputs other than the RGB image. It can accurately estimate the poses of multiple objects in real time, with inference time unaffected by the number of target objects. On the LM-O and YCB-V datasets, our method outperforms other RGB-based single-model methods, achieving higher accuracy. Even when compared with multi-model methods and approaches that use additional refinement, our results remain competitive.

1. Introduction

In the field of machine vision, six degrees of freedom (6D) pose estimation plays a crucial role. This technology can measure an object's position (coordinates along the x, y, and z axes) and orientation (roll, pitch, and yaw angles). Therefore, it can help robots accurately understand objects' spatial posture, which is crucial in robots grasping, moving, or manipulating objects[1]. During these processes, it is common to encounter scenes with multiple objects to estimate. These objects often vary in size and may occlude each other. However, despite significant advancements in 6D pose estimation technology, existing methods still struggle to effectively handle the multi-object scenes.

Specifically, in recent years, with the deepening research in deep learning [3, 6, 26], methods for 6D pose estimation have been continuously emerging[2, 4, 5, 7, 8, 9]. At the same time, physically-based rendering techniques have narrowed the gap between synthetic and real images, improving the model's generalization ability in real-world scenarios [28]. In this context, deep learning-based methods have made significant advancements in the task of 6D pose estimation, surpassing traditional methods based on point-to-point features in terms of both accuracy and speed [27]. Among these methods, the most direct approach is to directly regress the 6D pose of objects from input images, without the need for additional steps or models for feature point detection or matching [5, 7, 9, 21]. These methods are suitable for both single-object and multi-object scenarios. And they are easy to deploy and train. However, they still lag behind state-of-the-art methods in terms of accuracy due to occlusions and varying object sizes[30, 29]. Indirect methods achieve higher accuracy by utilizing the idea of correspondence [23, 24, 12, 11, 32]. These methods first predict the key points of objects in 2D images and then match them with corresponding 3D key points. Finally, these correspondences are input into the Perspective-n-Points (PnP) algorithm or RANSAC algorithm to obtain the object's pose. This two-stage approach has improved accuracy. However, these methods also have drawbacks: on the one hand, due to the non-differentiable nature of PnP and RANSAC, these methods cannot be directly trained and deployed end-to-end; on the other hand, even if only one object is processed, the iterative process of RANSAC can be very time-consuming[32].

In addition, there are some other challenges in existing methods when it comes to multi-object pose estimation tasks. On the one hand, existing methods mostly design networks for specific types of objects, making it difficult to generalize to other types of objects. Therefore, to improve accuracy on the YCB-V dataset, many methods train 21 pose estimation models for 21 different types of objects [8, 14]. This means high system complexity and resource consumption when deployed in practice. On the other hand, in multi-object scenarios, there are differences in size and scale between objects, which can lead to imbalance and bias during network training[30].

To address these issues, we propose SEMPose, a **S**ingle **E**nd-to-end network for **M**ulti-object **P**ose estimation. Our

network requires no additional information besides RGB images, such as 3D models[33], depth images[34], object symmetry information[10], or ground-truth RoI information[9].

Specifically, we adopt a backbone-neck-head structure that enables the entire network to be trained end-to-end. To address the issue of varying object scales in multi-object scenes, we designed a texture-shape guided feature pyramid structure to hierarchically capture the fused features of objects with different sizes. For the issue of object occlusions, we use the unoccluded parts of objects to guide positive sample selection during sampling. What's more, to improve the accuracy of pose prediction, we employ different strategies for predicting rotation and translation, and design rotation iteration head and translation iteration head. This enables our network to learn coordinate features closely related to translation and fully utilize contextual information for a more comprehensive perspective in prediction.

Overall, our contributions are as follows:

- We propose an end-to-end pose estimation network that can simultaneously handle multiple objects. Our approach requires only one model for multiple objects, and the runtime is independent of the number of objects. Additionally, we train the network using only raw RGB data.

- We propose a texture-shape guided feature pyramid structure that can handle objects of varying sizes and enhance feature extraction. Additionally, we decouple the prediction of rotation and translation, and propose iteration heads. This enables our network to improve the accuracy of rotation and translation predictions. Compared to GDR-Net, we reduce the average translation estimation error by 2.47cm and the average rotation estimation error by 8.87°.

- In methods using only RGB images, we achieve state-of-the-art performance on the LM-O and YCB-V datasets. Even compared to multi-model methods and those utilizing 3D models, SEMPose can achieve similar or even superior performance.

2. Related Works

2.1. Indirect Methods.

The mainstream methods for 6D pose estimation adopt an indirect regression approach. Some of these methods [8, 23, 24, 12, 11, 32] are based on correspondence principles. They first detect feature points of the target object in an image, then match these feature points with corresponding features in a known model. Subsequently, they utilize algorithms such as RANSAC or Perspective-n-points (PnP) to compute the 6D pose of the target object based on the matched point pairs. For example, PVNet [8] predicts pixel-level vectors pointing to key points, then uses these vectors to determine the key points' positions through a RANSAC-based voting mechanism. Finally, PnP is used for calculation. Some methods [23, 24] predict belief maps and

affinity maps for each object category. Belief maps represent the likelihood of keypoint locations, while affinity maps represent the correlation between 3D bounding box vertices and the corresponding center point. Then, 2D key points are extracted from the predicted belief maps, and finally, the PnP algorithm is used to calculate the pose. Although these methods can effectively reduce the impact of occlusions, the non-differentiable nature of RANSAC/PnP prevents end-to-end training and deployment; moreover, an increase in the number of objects in the scene significantly raises the demand for computational resources.

Another group of methods is based on the idea of template matching[33, 35]. These methods first construct a 2D template database using a 3D model, and then compare the input image with all templates in the database using a sliding window algorithm to estimate the pose. The accuracy of this approach improves with an increase in the number of templates, but it also has several significant drawbacks: firstly, obtaining accurate 3D models can be challenging in practical applications; secondly, as the number of objects to be estimated in the scene increases, the size of the template database will also increase, leading to a significant decrease in processing speed; thirdly, changes in texture due to occlusions or lighting can greatly decrease accuracy. Overall, these indirect methods exhibit several limitations in scenarios involving multiple objects.
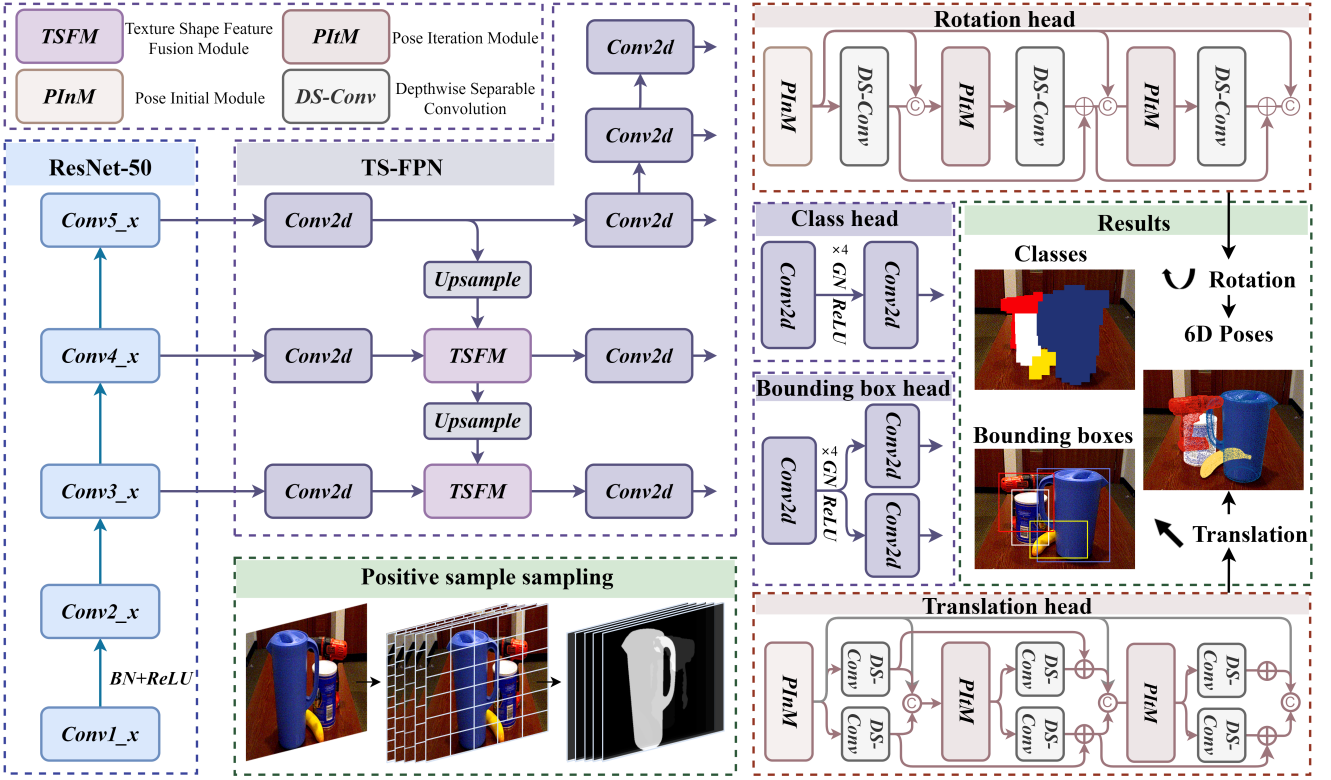
2.2. Direct Methods.

The most straightforward way to estimate 6D pose is to directly regress the 6D pose[4, 5, 7, 9, 21]. For instance, some methods [4, 7, 21] evaluate the matching loss by calculating the average squared distance between points on the model under the ground truth pose and the estimated pose. This approach assesses the discrepancies in model points across the two poses to enhance the accuracy of 6D pose estimation. PoET[9] utilizes a neural network to generate multi-scale feature maps from monocular RGB images and detect objects. These features and bounding boxes are then processed through a Transformer, ultimately predicting the 6D pose of each object using separate rotation and translation heads. GDR-Net[5] first predicts intermediate geometric feature maps, including dense correspondence maps and surface area attention maps. A 2D convolutional Patch-PnP module then directly regresses the 6D pose from these geometric features.

These methods are mostly end-to-end, which is beneficial for training and deployment. However, because convolutional neural networks directly regress the 6D pose from images, they face challenges in learning varying degrees of translation dependencies and struggle with effectively handling occlusions[17, 29]. Therefore, there is significant room for improvement in the accuracy of these methods.

3. Proposed Method

Given an RGB image I containing N objects $\mathcal{O} = \{\mathcal{O}_i \mid i = 1, 2, \cdots, N\}$, our goal is to simultaneously estimate their 6D poses $\mathbf{P}_i = [\mathbf{R}_i | \mathbf{t}_i]$, $i = 1, 2, \cdots, N$, where,

Figure 1: Framework of SEMPose. Given an input RGB image *I*, SEMPose first uses Resnet50 to extract basic features. Then, SEMPose employs a Texture-Shape Guided Feature Pyramid Network (TS-FPN) to fuse these features and produce feature maps at five different scales. Finally, SEMPose uses four heads to predict the categories, bounding boxes, rotations, and translations of the objects. After post-processing, the poses are obtained. Additionally, the network's training process relies on a positive sample sampling strategy guided by the visible parts.

$\mathbf{R}_i \in SO(3)$ is the rotation matrix, and $\mathbf{t}_i \in \mathbb{R}^3$ is the translation vector. The entire 6D pose $\mathbf{P}_i$ represents a rigid transformation from the camera coordinate system to the object coordinate system.

Figure 1 provides an overview schematic of our proposed method. Our SEMPose consists of three parts: the backbone, neck, and heads. The core components include our designed TS-FPN, pose estimation heads, and a positive sample selection strategy guided by visible parts.
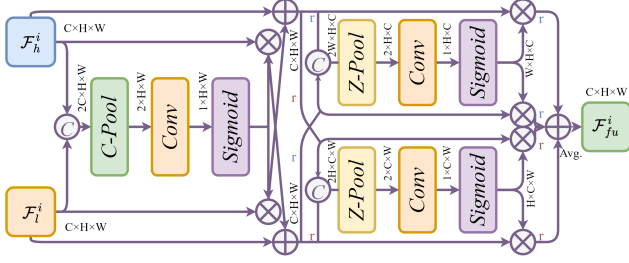
In the following, we will first (Sec. 3.1) provide a detailed introduction to our TS-FPN. Then (Sec. 3.2), we will explain our regression strategy for rotation and translation. Following this (Sec. 3.3), we will describe how we designed the network to implement this strategy. Finally (Sec. 3.4), we will introduce our network's approach to handling occlusion, specifically by sampling from the visible parts.

3.1. Texture-Shape Guided Feature Pyramid Network

In neural networks, lower layers typically capture basic features of input images, such as edges, colors, and texture information. These features vary with the rotation of objects. Therefore, by capturing these high-frequency details, neural networks can better understand the orientation of objects, leading to more accurate estimates of rotation. As the network deepens, the feature maps abstract the input data more

deeply, depicting more complex edge structures and specific shapes. This low-frequency information often reflects the overall shape and larger structural features of objects, which is crucial for estimating the translation. This is because the global shape and position information of objects help the network determine the center and position of the object. For example, when an object moves, changes are mainly related to low-frequency information, while high-frequency information such as edges and textures remains relatively stable.

Traditional Feature Pyramid Networks[6], by integrating the high-resolution information from lower levels with the semantically rich information from higher levels, can better handle targets of different scales. This has achieved good results in the field of object detection, but remains insufficient for pose estimation[15, 16]. Therefore, we propose a Texture-Shape-guided Feature Pyramid Networks (TS-FPN) to further integrate high and low-frequency features. Compared to FPN, in the top-down process, we do not simply use an upsampling and addition approach. As shown in Figure 2, our TS-FPN concatenates the upsampled low-frequency features $\mathcal{F}_l^i \in \mathbb{R}^{C_i \times H_i \times W_i}$ with the high-frequency features $\mathcal{F}_h^i \in \mathbb{R}^{C_i \times H_i \times W_i}$ along the channel dimension. It then captures the spatial dependencies of H and W to

Figure 2: The structure diagram of texture shape feature fusion module.

generate attention weights. These attention weights are applied to $\mathcal{F}_l^i$ and $\mathcal{F}_h^i$ separately, and then the weighted high-frequency/low-frequency features are added back to $\mathcal{F}_l^i/\mathcal{F}_h^i$ to obtain enhanced features $\hat{\mathcal{F}}_h^i/\hat{\mathcal{F}}_l^i$. This enhancement process can be described as:

$$\hat{\mathcal{F}}_h^i = \mathcal{F}_h^i \oplus (\mathcal{F}_l^i \otimes \sigma(Conv(C\text{-}Pool(C(\mathcal{F}_h^i, \mathcal{F}_l^i))))),$$
$$\hat{\mathcal{F}}_l^i = \mathcal{F}_l^i \oplus (\mathcal{F}_h^i \otimes \sigma(Conv(C\text{-}Pool(C(\mathcal{F}_h^i, \mathcal{F}_l^i))))), \quad (1)$$

where $\oplus$ refers to element-wise addition, $\otimes$ denotes element-wise multiplication, $\sigma(\cdot)$ represents the sigmoid function, $Conv(\cdot)$ stands for convolution with a kernel size of 7, $C\text{-}Pool(\cdot)$ indicates a channel pool, and $C(\cdot)$ means the concatenation along the channel dimension.

After capturing the spatial dependencies, we rotate $\hat{\mathcal{F}}_h^i/\hat{\mathcal{F}}_l^i$ 90° counterclockwise along the H-axis/W-axis[25]. We then concatenate the features processed in the same way and generate attention weights between C and W, and between C and H, in parallel channel attention. These weights are used to reweight the corresponding enhanced features, followed by the inverse rotation. Finally, we take the average to obtain the final fused features $\mathcal{F}_{fu}^i$. The cross-channel feature fusion can be described as:

$$\tilde{\mathcal{F}}_1^i = \hat{\mathcal{F}}_{h,r_H^+}^i \otimes \sigma(Conv(Z\text{-}Pool(C(\hat{\mathcal{F}}_{h,r_H^+}^i, \hat{\mathcal{F}}_{l,r_H^+}^i)))),$$
$$\tilde{\mathcal{F}}_2^i = \hat{\mathcal{F}}_{l,r_H^+}^i \otimes \sigma(Conv(Z\text{-}Pool(C(\hat{\mathcal{F}}_{h,r_H^+}^i, \hat{\mathcal{F}}_{l,r_H^+}^i)))),$$
$$\tilde{\mathcal{F}}_3^i = \hat{\mathcal{F}}_{h,r_W^+}^i \otimes \sigma(Conv(Z\text{-}Pool(C(\hat{\mathcal{F}}_{h,r_W^+}^i, \hat{\mathcal{F}}_{l,r_W^+}^i)))), \quad (2)$$
$$\tilde{\mathcal{F}}_4^i = \hat{\mathcal{F}}_{l,r_W^+}^i \otimes \sigma(Conv(Z\text{-}Pool(C(\hat{\mathcal{F}}_{h,r_W^+}^i, \hat{\mathcal{F}}_{l,r_W^+}^i)))),$$

$$\mathcal{F}_{fu}^i = \frac{1}{4}(\tilde{\mathcal{F}}_{1,r_H^-}^i + \tilde{\mathcal{F}}_{2,r_H^-}^i + \tilde{\mathcal{F}}_{3,r_W^-}^i + \tilde{\mathcal{F}}_{4,r_W^-}^i), \quad (3)$$

where $Z\text{-}Pool(\cdot)$ represents the max and average pooling along the 0th dimension[25], and the subscripts r_H^+, r_W^+, r_H^-, and r_W^- indicate 90° rotations: r_H^+ for counterclockwise along the H, r_W^+ for counterclockwise along the W, r_H^- for clockwise along the H, and r_W^- for counterclockwise along the W.

Finally, our TS-FPN performs downsampling on the output side. It ultimately outputs feature maps at five scales, fused with shape and texture information, for further object detection and pose estimation.

3.2. Pose Regression Strategy

After obtaining the multi-scale features extracted by TS-FPN, we need to utilize these features for 6D pose estimation. For the rotation, our primary focus is on the object's appearance in the image, as rotation can significantly alter the object's appearance[36]. Correspondingly, the estimation of the translation focuses more on the distance of the object's center point relative to the camera, as this directly affects the size of the object in the image[37]. Therefore, the rotation and translation should be estimated using different strategies.

In the regression of rotation, all representations in four-dimensional or lower real Euclidean spaces are discontinuous. This means the widely used 3D and 4D representations (Euler angles and quaternions) are discontinuous, which poses challenges for neural network learning[22]. To address this, we adopted a higher-dimensional 6D representation. Compared to traditional representations, its main advantages are simplicity and continuity in most cases, and it has been proven effective in many studies[5, 22]. In the 6D representation, a rotation is represented by two 3D vectors, which correspond to two orthogonal directions in 3D space. Our strategy is to directly regress the 6D vector $\boldsymbol{r}_{6d} = (r_1, r_2, r_3, r_4, r_5, r_6)$ through the rotation head. Then, we can recover two 3D vectors from the 6D vector, i.e., $\boldsymbol{a}_1 = (r_1, r_2, r_3), \boldsymbol{a}_2 = (r_4, r_5, r_6)$. Next, we construct three standard orthogonal bases $\boldsymbol{e}_1, \boldsymbol{e}_2$, and $\boldsymbol{e}_3$ from these two 3D vectors. Finally, we use these three base vectors as the columns of the rotation matrix to obtain the final rotation matrix $\mathbf{R} = [\boldsymbol{e}_1^T, \boldsymbol{e}_2^T, \boldsymbol{e}_3^T]$. This process can be represented as:

$$\boldsymbol{e}_1 = \Phi(\boldsymbol{a}_1), \ \boldsymbol{e}_2 = \Phi(\boldsymbol{e}_1 \times \boldsymbol{e}_2), \ \boldsymbol{e}_3 = \boldsymbol{e}_1 \times \boldsymbol{e}_2, \quad (4)$$

where $\Phi(\cdot)$ denotes vector normalization, while the symbol $\times$ denotes the vector cross product.

After obtaining the rotation matrix, we use a geodesic loss shown in Equ. 5 to train the rotation head[20]. This formula quantifies the difference in angles between the predicted rotation and the actual rotation.

$$\mathcal{L}_{rot\_6d} = \arccos\left(\frac{tr\left(\mathbf{R}_{gt}^T \mathbf{R}_{pred}\right) - 1}{2}\right). \quad (5)$$

To validate the effectiveness of this representation, we also experimented with the quaternion representation. Quaternions have two different representations for the same rotation. Therefore, we normalized both the predicted and actual quaternions by ensuring that the real part is non-negative. Additionally, we utilized a loss function shown in Equ. 6, which is essentially similar to the geodesic loss[20]. The comparison of the effectiveness of the two representation methods is provided later in Sec. 4.

$$\mathcal{L}_{rot\_quat} = 2\arccos\left(\left|\langle \mathbf{q}_{gt}, \mathbf{q}_{pred} \rangle\right|\right). \quad (6)$$

For the translation, due to the translational invariance of convolutional operations, the network's prediction of absolute coordinates is not as effective as relative coordinates[21].
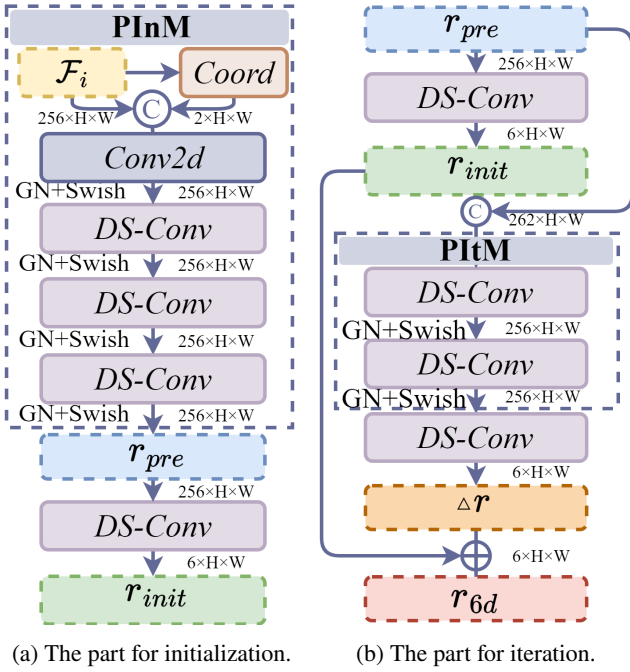
(a) The part for initialization.　　(b) The part for iteration.

Figure 3: The structure of the rotation head.

Therefore, we did not directly regress the translation vector $\mathbf{t} = \left(t_x, t_y, t_z\right)^T$ like in PoET. First, the transformation between the camera coordinate system and the pixel coordinate system can be expressed as $t_z \left(c_x, c_y, 1\right)^T = K \left(t_x, t_y, t_z\right)^T$, where c_x and c_y represent the coordinates of the projected 3D point on the image, and K represents the camera intrinsic parameters. Based on this, we decompose $\left(t_x, t_y, t_z\right)^T$ into the projected points $\left(c_x, c_y\right)^T$ and t_z. Then, we decompose $\left(c_x, c_y\right)^T$ into the anchor point coordinates $\left(a_x, a_y\right)^T$ and the relative coordinates $(\Delta x, \Delta y)^T$. Therefore, we can obtain the translation by predicting the offset $(\Delta x, \Delta y)^T$ and t_z, i.e.,

$$\begin{cases} t_x = \frac{(a_x + \Delta x - p_x) \cdot t_z}{f_x}, \\ t_y = \frac{(a_y + \Delta y - p_y) \cdot t_z}{f_y}, \\ t_z = t_z, \end{cases} \quad (7)$$

where f_x, f_y, p_x and p_y are obtained from the camera's intrinsic matrix $K = \begin{pmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{pmatrix}$. During training, we use the translation losses as

$$\mathcal{L}_{tran_1} = \left\| \left(t_x, t_y\right)_{pred}, \left(t_x, t_y\right)_{gt} \right\|_2, \quad (8)$$

$$\mathcal{L}_{tran_2} = \left\| t_{z,pred}, t_{z,gt} \right\|_1. \quad (9)$$

3.3. Pose Estimation Heads

In the previous section, we have established the regression strategies for rotation and translation. When implementing these strategies into the network structure, we need to regress $\mathbf{r}_{6d}$ and $\left(\Delta x, \Delta y, t_z\right)^T$ separately. The most straightforward approach is similar to the heads in the object detection task[15, 16]. Specifically, it involves the use of class head and bounding box head, as illustrated in Figure 1. Four 3x3 convolutions are employed to further refine the feature maps produced by the TS-FPN. Lastly, a convolutional layer outputs the prediction results. However, compared to the object detection task, the 6D pose estimation task requires higher accuracy. Additionally, regressing the translation relies on coordinate features, which are difficult to extract with ordinary convolutions. Hence, we do not use a structure similar to the class head to regress rotation and translation. Inspired by EfficientPose[21], we adopt an iterative refinement approach to approximate the final result. Based on this, we design the rotation head and translation head. Taking the rotation head as an example, its structure is illustrated in Figure 3.

The approximation approach is divided into two steps. Firstly, the pose initialization module (PInM) is utilized to refine the feature maps $\mathcal{F}_i \in \mathbb{R}^{C \times H \times W}, i \in \{1, 2, 3, 4, 5\}$ from TS-FPN. This refinement process results in obtaining $\mathbf{r}_{pre}$, which is then further processed to obtain $\mathbf{r}_{init}$. Specifically, the x and y coordinates of $\mathcal{F}_i$ are normalized to the range $[0, 1]$ and then mapped to $[-1, 1]$. This process generates two channels with coordinate features, which are then concatenated with $\mathcal{F}_i$. The concatenated data passes through a standard convolutional layer. Subsequently, it is processed through three depthwise separable convolutional layers, along with group normalization and the Swish activation function, resulting in $\mathbf{r}_{pre}$. $\mathbf{r}_{pre}$, retaining the preliminary raw features, is utilized for further iterations. Finally, the number of channels is reduced to 6 using a depthwise separable convolutional layer to obtain $\mathbf{r}_{init}$. Thus, we have completed the addition of coordinate features and the initial utilization of overall features. Our first step can be represented as:
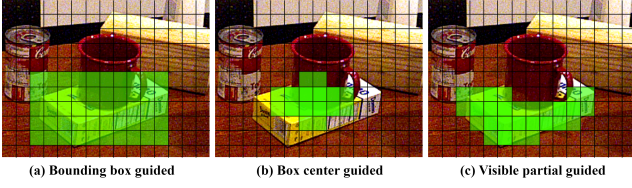
$$\mathbf{r}_{pre} = DS\text{-}Conv^3 \left(Conv \left(C \left(\mathcal{F}_i, Coord \left(\mathcal{F}_i \right) \right) \right) \right), \quad (10)$$

$$\mathbf{r}_{init} = DS\text{-}Conv \left(\mathbf{r}_{pre} \right), \quad (11)$$
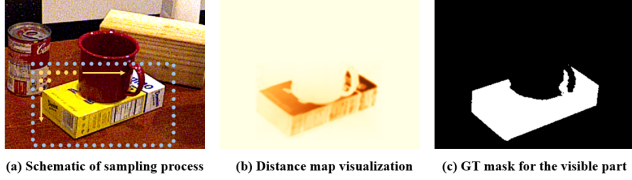
where $DS\text{-}Conv \left(\cdot \right)$ refers to a depthwise separable convolution and $Coord \left(\cdot \right)$ indicates the operation of extracting coordinate information described above.

Secondly, we concatenate $\mathbf{r}_{pre}$ and $\mathbf{r}_{init}$ along the channel dimension. Then we feed the concatenated feature map into a pose iteration module (PItM) consisting of two depthwise separable convolutions. After that, we use another depthwise separable convolution to compress the channels, producing $\Delta \mathbf{r}$. At last, we add $\Delta \mathbf{r}$ to $\mathbf{r}_{init}$, obtaining the optimized $\mathbf{r}_{6d}$. The advantage of this structure is that it not only considers the initial rotation estimation but also integrates features from the early layers of the network. This helps to utilize more contextual information and offers a more comprehensive perspective for making predictions. Thus, we have completed one iteration. Our second step can be represented as:

$$\Delta \mathbf{r} = DS\text{-}Conv^3 \left(C \left(\mathbf{r}_{pre}, \mathbf{r}_{init} \right) \right), \quad (12)$$

(a) Bounding box guided   (b) Box center guided   (c) Visible partial guided

Figure 4: Schematic diagrams of the three different sampling strategies [15, 16, 29]. A darker green color indicates a higher probability of sampling.

(a) Schematic of sampling process   (b) Distance map visualization   (c) GT mask for the visible part

Figure 5: The schematic diagram of the sampling process from visible parts.

$$\boldsymbol{r}_{6d} = \boldsymbol{r}_{init} + \Delta\boldsymbol{r}. \tag{13}$$

Our translation head and rotation head are essentially the same. The only difference is that we use two branches in the translation head, as shown in Figure 1. This corresponds to $(\Delta x, \Delta y)^T$ and $\mathbf{t}_z$ in Sec. 3.2. At the end of the translation head, we concatenate the two branches along the channel dimension to output $(\Delta x, \Delta y, t_z)^T$. Further, we can convert it back to the translation vector $\mathbf{t}$.

Additionally, Our rotation and translation heads share weights across various scales, similar to FCOS's approach[15]. The weight sharing leads to a considerable reduction in the number of parameters. This method ensures that as long as positive samples are present on one scale, weights on other scales can also be updated, despite having fewer positive samples. This helps mitigate the problem of uneven sample distribution across levels of heads.

3.4. Positive Sample Selection

In this section, we will describe our approach to addressing occlusion in multi-object scenes. In single-stage object detectors, positive samples refer to sampling points linked to annotated instances of objects. These are utilized to depict the features of target categories during training. Typically, strategies for positive sample selection consider either all points within the GT bounding box or just those near the box center as positive samples[15, 16]. Hai et al. propose that it's more logical to use points from visible sections as positive samples for rigid objects[29]. In scenes with multiple objects, occlusion among objects is common, and the objects we study are all rigid. Therefore, during the training phase, we sample positive samples from the parts of the object that are visible. Different sampling strategies are illustrated in Figure 4.

To implement the above sampling strategies, we first uniformly set boundary points $\mathbb{B} = \{b_1, b_2, \cdots, b_m\}$ on the ground truth bounding box of the target object. Then, we

calculate the foreground-background discrepancy $\mathcal{V}(p, b)$ for each pixel point p within the bounding box relative to the boundary point b. Finally, based on the foreground-background discrepancy, we determine the parts of the object that can be seen. The overall process is illustrated in the Figure 5. We can see that the distance map and the ground truth visibility mask are essentially consistent. This means that during the process of positive sample sampling, we can achieve an effect similar to segmentation.

The principle of the above process is that boundary points are generally not part of the target object. The larger the discrepancy between a pixel point and a boundary point, the more likely the pixel point is part of the target object. Specifically, we use the minimum barrier distance $\mathcal{D}(C, l)$ related to the color difference [31] and the Euclidean distance $d(p, d)$ to represent $\mathcal{V}(p, b)$. This can be formulated as,

$$\mathcal{V}(p, b) = \min_{b \in \mathbb{B}} \min_{l \in \mathbb{L}_{\{p,b\}}} \left[\left(\frac{\mathcal{D}(C, l)}{255} \right)^2 + \alpha \cdot d(p, d) \right], \tag{14}$$

where $\mathbb{L}_{\{p,b\}}$ represents the set of paths between p and b, l is one of these paths, C represents the color brightness of p and α is a balance coefficient, set to 0.1 here, used to adjust the weight of color difference and distance. And $\mathcal{D}(C, l)$ can be obtained by

$$\mathcal{D}(C, l) = \max_{i=1}^{3} \left[\max_{p_j \in l} C_i(p_j) - \min_{p_j \in l} C_i(p_j) \right]. \tag{15}$$

In the formula, the outer i represents one of the three channels of RGB, and the inner p_j is a point on the path l. Furthermore, for the feature maps extracted by TS-FPN, we calculate the visibility probability for each cell by

$$\mathcal{P}(c) = \frac{\overline{\mathcal{V}}(c)}{\max_{c \in \mathbb{C}} \overline{\mathcal{V}}(c)}, \tag{16}$$

where $\overline{\mathcal{V}}(c)$ represents the average foreground-background discrepancy of pixels within a cell and $\mathbb{C}$ is the set of all cells on the feature maps. In our experiments, we set cells with $\mathcal{P}(c) > 0.25$ as positive samples. For more parameter settings, please refer to [29].

4. Experiments

In this section, we will first provide the implementation details of our experiment, descriptions of the datasets used, and the evaluation metrics. Then, we will compare our method with the state-of-the-art methods on two public datasets to demonstrate the superiority of SEMPose. Finally, we conduct ablation studies to prove the effectiveness of our specific designs.

4.1. Experimental Setup

Implementation Details. Our algorithm is designed based on PyTorch. We choose ResNet-50 [3] as the backbone network, utilize our TS-FPN for the neck, and design four

Table 1
Comparison with State of the Art on LM-O. We report the Recall of ADD(-S) in %. P.E. indicates whether to use 1 model or N models to estimate N types of objects. (*) marks the symmetrical objects. (-) refers to unavailable results[7, 9, 5, 8, 12, 13, 14].

Method	w/o Refinement				w/ Refinement				
	PoseCNN	PoET	GDR-Net	**Ours**	PVNet	GDR-Net	DPOD	DeepIM	RNNPose
P.E.	1	1	1	1	N	N	1	1	N
Ape	9.6	10.2	**44.9**	33.2	15.8	46.8	-	59.2	37.2
Can	45.2	31.8	79.7	**85.2**	63.3	90.8	-	63.5	88.1
Cat	0.9	9.0	**30.6**	30.1	16.7	40.5	-	26.2	29.2
Driller	41.4	33.9	67.8	**88.6**	65.7	82.6	-	55.6	88.1
Duck	19.6	15.4	**40.0**	29.0	25.2	46.9	-	52.4	49.2
Eggbox*	22.0	44.7	49.8	**77.0**	50.2	54.2	-	63.0	67.0
Glue*	38.5	58.7	73.7	67.3	49.6	75.8	-	71.7	63.8
HoleP.	22.1	24.7	62.7	**71.4**	36.1	60.1	-	52.5	62.8
Mean	24.9	28.5	56.1	**60.2**	40.8	62.2	47.3	55.5	60.7

heads corresponding to different tasks. Our TS-FPN is constructed from the last three layers of ResNet to extract features, generating multi-scale feature maps with 256 channels. In our network's training phase, we configure the batch size at 16 and opt for AdamW as the optimization algorithm, setting the initial learning rate to 4e-4. The optimizer's beta parameters are specified as $(0.9, 0.999)$, coupled with a weight decay factor of 5e-4 and an epsilon of 1e-8. The OneCycle policy governs the learning rate adjustments[18], peaking at 4e-4. The training spans over 150 epochs, with the learning rate experiencing a linear ascent to its peak within the initial 0.5% of the steps, followed by a linear decline throughout the subsequent steps. In the optimization configuration, we perform gradient clipping with a maximum norm of 35 using the L2 norm type. Most of our experiments are conducted on an RTX 4090D GPU and an AMD 3.1GHz CPU. Additionally, to ensure consistency with other methods, we measure the inference time on an RTX 3090 GPU and an Intel 2.8GHz CPU.

Datasets. We use two core datasets from the BOP dataset, LM-O[19] and YCB-V[7], to evaluate our method. The LM-O dataset provides 6D pose annotations for 8 low-texture objects under various occlusion conditions, making it well-suited for testing an algorithm's occlusion handling capabilities. We train exclusively on Physically-Based Rendering (PBR) data and test on real-world data. YCB-V is a highly challenging dataset with varying lighting conditions, cluttered backgrounds, and image noise. It includes 21 commonly found household objects of different sizes, some of which exhibit high symmetry and varying degrees of occlusion among objects. The original YCB-V dataset comprises 92 video sequences, of which we use 80 sequences for real image training and the remaining 12 sequences for testing. Additionally, we also use the officially provided PBR data for training.

Evaluation Metrics. We use the commonly applied ADD(-S) metric for evaluation. ADD measures the average distance between corresponding model points transformed by ground truth and estimated poses, suitable for

non-symmetric objects. ADD-S measures the average distance between each model point transformed by the estimated pose and the nearest model point transformed by the ground truth pose, suitable for symmetric objects. The specific formulas are as follows:

$$ADD = \underset{x \in \mathcal{M}}{avg} \left\| \left(\mathbf{R}_{gt}x + t_{gt}\right) - \left(\mathbf{R}_{pred}x + t_{pred}\right) \right\|, \quad (17)$$

$$ADD\text{-}S = \underset{x_1 \in \mathcal{M}}{avg} \underset{x_2 \in \mathcal{M}}{\min} \left\| \left(\mathbf{R}_{gt}x + t_{gt}\right) - \left(\mathbf{R}_{pred}x + t_{pred}\right) \right\|, \quad (18)$$

where $\mathcal{M}$ refers to the set of 3D model points, $\mathbf{R}_{gt}$ and t_{gt} represent the ground truth rotation and translation, while $\mathbf{R}_{pred}$ and t_{pred} represent the predicted rotation and translation. When evaluating on YCB-V, we also computed the AUC (area under curve) of ADD-S. Additionally, we calculated the geodesic distance as the rotation error and the L2 distance as the translation error for each object category, with the specific formulas as follows:

$$error_{rot} = \arccos\left(\frac{tr\left(\mathbf{R}_{gt}^T\mathbf{R}_{pred}\right) - 1}{2}\right), \quad (19)$$

$$error_{tran} = \left\| t_{gt} - t_{pred} \right\|. \quad (20)$$

4.2. Comparison with State of the Art
Results on LM-O. Table. 1 shows our comparison results with state-of-the-art methods on LM-O[7, 9, 5, 8, 12, 13, 14]. Among the single-model methods without using additional refinement, our SEMPose achieves the state-of-the-art results; compared to the methods that use 8 models for 8 objects, SEMPose also achieves similar accuracy. Moreover, SEMPose even outperforms some time-consuming methods that employ refinement.
Results on YCB-V. In Table. 2, we present the comparison of SEMPose with other state-of-the-art methods on YCB-V. Among the single-model methods without using refinement,
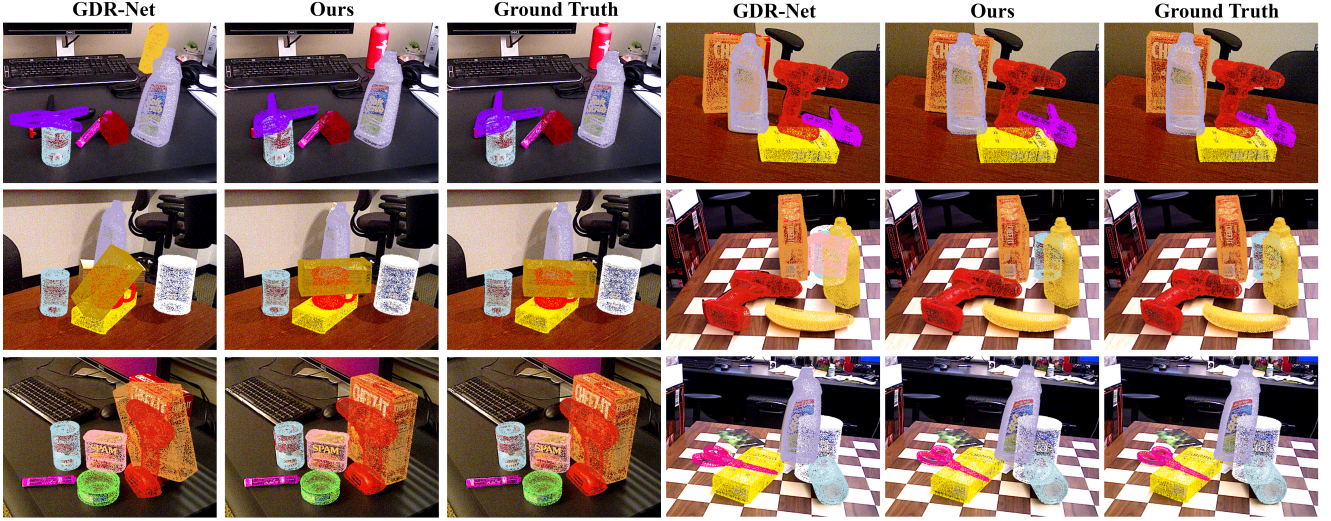
| GDR-Net | Ours | Ground Truth | GDR-Net | Ours | Ground Truth |

Figure 6: Qualitative comparison results on YCB-V.

Table 2
Comparison with State of the Art on YCB-V. We report the results evaluated by AUC of ADD-S and AUC of ADD(-S). ADD-S means using symmetric metrics for all objects, while ADD(-S) is used only for symmetric objects. P.E. indicates whether to use 1 model or N models to estimate N types of objects. (-) refers to unavailable results.

Method	Refinement	P.E.	AUC of ADD-S	AUC of ADD(-S)
PoseCNN[7]		1	75.9	61.3
SilhoNet[10]		1	79.6	-
PoET[9]		1	87.1	70.1
GDR-Net[5]		1	89.1	80.2
Ours		1	**92.2**	**85.2**
PVNet[8]		N	-	73.4
GDR-Net[5]		N	91.6	84.4
DeepIM[13]	✓	1	88.1	81.9
CosyPose[4]	✓	1	89.8	84.5
RNNPose[14]	✓	N	-	83.1

SEMPose continues to achieve the best results. For multi-model methods that use 21 models for prediction, our results remain competitive. Moreover, SEMPose also performs well in terms of accuracy compared to the time-consuming methods that employ refinement.

Additionally, we report a comparison of the average rotation error and the average translation error for each category. The methods listed in the Table. 3 are all single-model approaches for YCB-V. As the table shows, our method achieves lower translational prediction errors compared to other methods; the rotational prediction error is only bigger than that of SilhoNet, which employs rotation correction. Compared to GDR-Net, which performs second best in Table. 2, SEMPose reduces the translation prediction error by 2.47cm and the rotational error by 8.87 °.

In Figure. 6, we present qualitative comparison results on YCB-V. We project the 3D model points onto the image using the predicted poses and differentiate objects with various colors. It is evident that our method effectively handles occlusion, a challenge that other single-model methods struggle with. For both GDR-Net and our SEMPose, a score threshold of 0.4 is used before plotting to remove low-confidence predictions. Despite this, GDR-Net frequently misidentifies background objects as targets or misclassifies targets as other objects. This is a common issue among single-model methods.

4.3. Ablation Study on YCB-V

We conduct ablation studies on the YCB-V dataset to demonstrate the effectiveness of our designs. All results are for the same set described in Sec. 4.1.

The fusion of texture and shape. In Table. In row B0, we show the results without using the texture and shape fusion proposed in Sec. 3.1. After removing the feature fusion, the prediction error for translation increased by 29.6%, and the rotation error increased by 114.0%. At the same time, the AUC of ADD-S and ADD(-S) decreased by 7.0% and 14.1%, respectively. This indicates that by integrating texture and shape features, SEMPose can capture more accurate detail information, which is especially beneficial for rotation prediction.

Pose regression strategies. In rows C0 to C2, we show the outcomes of employing different pose regression strategies. Specifically: In row C0, we use quaternion instead of the 6D representation r_{6d} for representing rotation. In row C1, we shift from regressing $(\Delta x, \Delta y)^T$ and t_z separately to directly regressing $\mathbf{t} = (t_x, t_y, t_z)^T$. In row C2, we utilize the L2 distance between the predicted and actual translation vectors for translation loss. Each adjustment leads to varying degrees of increased errors and reduced accuracy. It's critical to highlight that making adjustments for either rotation or translation alone influences the accuracy of the other value.

Table 3
Comparison of average errors. We report the average errors for translation and rotation. (†) indicates the use of rotation correction from **SILHONET**[]. (*) marks the symmetrical objects. The results of GDR-Net are obtained through the official single model[5, 7, 9, 10].

Method	Average Translation Error [cm]					Average Rotation Error [°]				
	PoseCNN	SilhoNet†	PoET	GDR-Net	**Ours**	PoseCNN	SilhoNet†	PoET	GDR-Net	**Ours**
master chef can	3.29	3.02	2.26	5.40	**1.79**	50.70	**1.21**	80.12	48.16	22.33
cracker box	4.02	5.24	**3.14**	5.09	3.58	**19.69**	19.86	21.87	25.30	35.70
sugar box	3.06	2.10	1.42	5.23	**1.15**	9.29	12.28	**4.40**	20.10	15.14
tomato soup can	3.02	2.40	1.62	6.60	**1.57**	18.23	**1.91**	49.29	30.23	17.00
mustard bottle	1.72	1.65	1.42	2.61	**1.16**	9.94	**5.78**	27.73	9.16	13.78
tuna fish can	2.41	**1.57**	1.79	5.59	1.76	32.80	**1.46**	63.72	30.53	40.70
pudding box	3.69	7.15	1.94	1.78	**1.66**	10.20	20.95	6.87	**6.74**	10.21
gelatin box	2.49	1.09	1.41	1.67	**0.96**	**5.25**	12.52	7.19	7.88	35.35
potted meat can	3.65	4.30	**1.75**	5.14	1.93	28.67	7.27	**6.75**	18.24	14.67
banana	2.43	4.12	1.95	2.02	**0.92**	15.48	16.29	20.40	9.16	**8.25**
pitcher base	4.43	**1.31**	1.55	4.89	1.53	11.98	**6.64**	8.04	30.17	10.16
bleach cleanser	4.86	3.60	2.47	7.52	**3.19**	20.85	51.28	21.93	43.85	**17.49**
bowl*	5.23	3.30	1.76	3.81	**1.97**	75.53	49.95	**25.71**	69.60	38.05
mug	4.00	2.61	**1.85**	2.77	2.52	19.44	18.14	**5.59**	7.75	15.88
power drill	4.59	6.77	2.29	4.80	**1.85**	9.91	30.54	**6.45**	28.27	14.26
wood block*	6.34	5.59	4.75	6.33	**2.68**	23.63	25.52	**14.32**	79.75	42.37
scissors	6.40	9.91	3.72	3.69	**2.06**	43.98	155.53	**6.27**	87.35	30.71
large marker	3.89	3.24	2.75	4.45	**1.76**	92.44	**10.44**	25.91	80.28	41.38
large clamp*	9.79	6.27	**2.33**	7.46	4.50	38.12	**3.54**	4.88	12.01	54.25
extra large clamp*	8.36	4.86	**3.10**	5.83	3.78	34.18	29.18	**26.01**	52.39	38.59
foam brick*	2.48	3.98	3.42	2.36	**1.75**	22.67	13.84	36.34	**4.41**	71.46
Mean	4.16	3.49	2.12	4.53	**2.06**	27.79	**16.04**	27.26	33.39	24.52

Table 4
Ablation Study on YCB-V. Ablation on feature fusion, pose regression methods, pose head structures, and positive sample sampling strategies.

Row	Method	AUC of ADD-S	AUC of ADD(-S)	Avg. T. Error [cm]	Avg. R. Error [◦]
A0	**SEMPose (Ours)**	**92.2**	**85.2**	**2.06**	**24.52**
B0	A0 → without fusion of texture and shape features	85.7	73.2	2.67	52.48
C0	A0: r_{6d} → quaternion	87.7	75.3	2.69	50.45
C1	A0: indirectly regress the translation vector → directly regression	83.3	67.7	3.51	64.91
C2	A0: separate translation loss → combined translation loss	91.7	82.7	2.15	27.58
D0	A0: iterative refinement head → structure similar to the class head	74.1	56.7	5.16	72.37
E0	A0: sampling from visible parts → from center parts	84.2	70.2	3.12	52.64

This interdependence arises because the four heads share a common set of features derived from the backbone and neck of the network. Consequently, modifications targeted at one parameter can inadvertently alter the backbone and neck's parameters, thereby affecting the other value.

Pose heads structure. In row D0, we substitute the iterative refinement heads described in Sec. 3.3 with ones similar to the class head. This adjustment results in a significant loss in accuracy. The reason is that a simple head structure cannot effectively utilize the fused features from TS-FPN. Moreover, conventional convolutional layers do not proficiently extract coordinate features.

Positive sample selection strategies. In row E0, we select positive samples for training from the center of the bounding box, leading to an expected decrease in accuracy. As mentioned in Sec. 3.4, occlusion is very common in pose estimation tasks. Significant occlusion results in the center
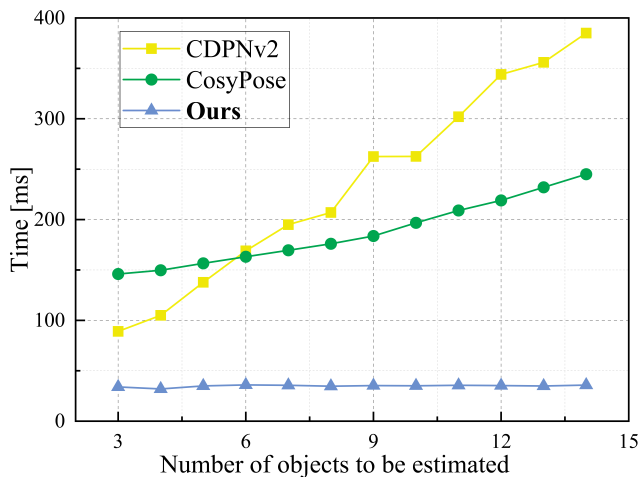
of the bounding box no longer belonging to the target object. Sampling from visible parts can avoid this issue.

4.4. Runtime Analysis

On an RTX 3090 GPU and an Intel 2.8GHz CPU, inputting a 640×480 image, our SEMPose takes an average of 31ms to obtain the 6D poses of all target objects in the picture. This inference speed is highly competitive among mainstream methods. Moreover, as shown in Figure 7, our inference time is independent of the number of objects in the image, which marks a significant difference from other methods.

5. Conclusion and Future Work

In this work, we propose the novel SEMPose, a single end-to-end network for multi-object pose estimation. In

Figure 7: Runtime comparison under the same conditions. The running time of existing methods becomes longer as the number of objects to be estimated increases[2, 4], whereas our method is unaffected.

methods using only RGB images, SEMPose achieves state-of-the-art performance on the LM-O and YCB-V datasets. The key to success is that we employ a shape texture-guided feature processing strategy. This allows our SEMPose to better recognize individual objects from a multi-object scene. On the other hand, our iterative head structure also reduces the estimation error for rotations and translations. This guarantees the estimation accuracy of SEMPose. In addition, we sample positive samples from visible parts, which copes well with the occlusion problem in multi-object scenes. Based on the above, the SEMPose can predict the 6D poses of all target objects in a image accurately and in real time.

In the future, we plan to extend our work to category-level pose estimation, which involves predicting the poses of previously unseen objects within the same category. Additionally, we aim to explore the integration of depth information to further improve the robustness and accuracy of pose estimation, especially in challenging scenarios with severe occlusions and complex backgrounds.

CRediT authorship contribution statement

Xin Liu: Conceptualization, Methodology, Coding, Writing original draft. **Hao Wang:** Coding, Writing, Reviewing, Editing. **Shibei Xue:** Supervision, Writing, Reviewing, Editing. **Dezong Zhao:** Supervision, Writing, Reviewing, Editing.

References

[1] Y. He, J. Gao, and Y. Chen, "Deep learning-based pose prediction for visual servoing of robotic manipulators using image similarity," in *Neurocomputing*, vol. 491. Elsevier, 2022, pp. 343–352.

[2] Z. Li, G. Wang, and X. Ji, "Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7678–7687.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[4] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, "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.* Springer, 2020, pp. 574–591.

[5] G. Wang, F. Manhardt, F. Tombari, and X. Ji, "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*, 2021, pp. 16 611–16 621.

[6] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[7] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: a convolutional neural network for 6d object pose estimation in cluttered scenes," *Robotics: Science and Systems (RSS),*, 2018.

[8] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4561–4570.

[9] T. G. Jantos, M. A. Hamdad, W. Granig, S. Weiss, and J. Steinbrener, "Poet: pose estimation transformer for single-view, multi-object 6d pose estimation," in *Conference on Robot Learning.* PMLR, 2023, pp. 1060–1070.

[10] G. Billings and M. Johnson-Roberson, "Silhonet: An rgb method for 6d object pose estimation," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3727–3734, 2019.

[11] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann, "Segmentation-driven 6d object pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3385–3394.

[12] S. Zakharov, I. Shugurov, and S. Ilic, "Dpod: 6d pose object detector and refiner," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1941–1950.

[13] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "Deepim: Deep iterative matching for 6d pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 683–698.

[14] Y. Xu, K.-Y. Lin, G. Zhang, X. Wang, and H. Li, "Rnnpose: 6-dof object pose estimation via recurrent correspondence field estimation and pose optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[15] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: a simple and strong anchor-free object detector," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 44, no. 04, pp. 1922–1933, 2022.

[16] S. Zhang, C. Chi, Y. Yao, Z. Lei, and S. Z. Li, "Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9759–9768.

[17] R. Liu, J. Lehman, P. Molino, F. Petroski Such, E. Frank, A. Sergeev, and J. Yosinski, "An intriguing failing of convolutional neural networks and the coordconv solution," *Advances in neural information processing systems*, vol. 31, 2018.

[18] L. N. Smith and N. Topin, "Super-convergence: very fast training of neural networks using large learning rates," in *Artificial intelligence and machine learning for multi-domain operations applications*, vol. 11006. SPIE, 2019, pp. 369–386.

[19] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II 13.* Springer, 2014, pp. 536–551.

[20] S. Mahendran, H. Ali, and R. Vidal, "3d pose regression using convolutional neural networks," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 2174–2182.

[21] Y. Bukschat and M. Vetter, "Efficientpose: An efficient, accurate and scalable end-to-end 6d multi object pose estimation approach," *arXiv preprint arXiv:2011.04307*, 2020.

[22] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5745–5753.

[23] Q. Guan, Z. Sheng, and S. Xue, "Hrpose: Real-time high-resolution 6d pose estimation network using knowledge distillation," *Chinese Journal of Electronics*, vol. 32, no. 1, pp. 189–198, 2023.

[24] Q. Guan, W. Li, S. Xue, and D. Li, "High-resolution representation object pose estimation from monocular images," in *2021 China Automation Congress (CAC)*. IEEE, 2021, pp. 980–984.

[25] D. Misra, T. Nalamada, A. U. Arasanipalai, and Q. Hou, "Rotate to attend: Convolutional triplet attention module," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021, pp. 3139–3148.

[26] L. Tang, H. Zhang, H. Xu, and J. Ma, "Rethinking the necessity of image fusion in high-level vision tasks: A practical infrared and visible image fusion network based on progressive semantic injection and scene fidelity," *Information Fusion*, vol. 99, p. 101870, 2023.

[27] M. Sundermeyer, T. Hodaň, Y. Labbe, G. Wang, E. Brachmann, B. Drost, C. Rother, and J. Matas, "Bop challenge 2022 on detection, segmentation and pose estimation of specific rigid objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2784–2793.

[28] M. Denninger, M. Sundermeyer, D. Winkelbauer, D. Olefir, T. Hodan, Y. Zidan, M. Elbadrawy, M. Knauer, H. Katam, and A. Lodhi, "Blenderproc: Reducing the reality gap with photorealistic rendering," in *International Conference on Robotics: Sciene and Systems, RSS 2020*, 2020.

[29] Y. Hai, R. Song, J. Li, M. Salzmann, and Y. Hu, "Rigidity-aware detection for 6d object pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 8927–8936.

[30] S. Thalhammer, P. Hönig, J.-B. Weibel, and M. Vincze, "Open challenges for monocular single-shot 6d object pose estimation," *arXiv preprint arXiv:2302.11827*, 2023.

[31] Y. Hu, Y. Li, R. Song, P. Rao, and Y. Wang, "Minimum barrier superpixel segmentation," *Image and Vision Computing*, vol. 70, pp. 1–10, 2018.

[32] T. Hodan, D. Barath, and J. Matas, "Epos: Estimating 6d pose of objects with symmetries," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 703–11 712.

[33] J. Lin, L. Liu, D. Lu, and K. Jia, "Sam-6d: Segment anything model meets zero-shot 6d object pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 27 906–27 916.

[34] Y. Lin, Y. Su, P. Nathan, S. Inuganti, Y. Di, M. Sundermeyer, F. Manhardt, D. Stricker, J. Rambach, and Y. Zhang, "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*, 2024, pp. 10 148–10 158.

[35] Y. Labbé, L. Manuelli, A. Mousavian, S. Tyree, S. Birchfield, J. Tremblay, J. Carpentier, M. Aubry, D. Fox, and J. Sivic, "Megapose: 6d pose estimation of novel objects via render & compare," in *Conference on Robot Learning*. PMLR, 2023, pp. 715–725.

[36] J. Brownlee, "A gentle introduction to object recognition with deep learning," *Machine Learning Mastery*, vol. 5, p. 10, 2019.

[37] A. B. Amjoud and M. Amrouch, "Object detection using deep learning, cnns and vision transformers: a review," *IEEE Access*, 2023.