The Visual Computer (2024) 40:5421–5434
https://doi.org/10.1007/s00371-023-03113-4

ORIGINAL ARTICLE

6D object pose estimation based on dense convolutional object center voting with improved accuracy and efficiency

Faheem Ullah[1] · Wu Wei[1] · Zhun Fan[2] · Qiuda Yu[1]

Accepted: 1 September 2023 / Published online: 25 November 2023

Abstract
6D object pose estimation is an important application of computer vision and a basic module in robotic manipulation, but dealing with occlusion in a cluttered environment, handling symmetries, and textureless surfaces, are real issues. Other issues with such systems are accuracy and efficiency. The recent two-stage methods perform well in terms of accuracy; however, a linear increase in their runtimes occurs due to the increase in the number of objects in a scene. This paper proposes a fully convolutional and parallel architecture that obtains the 3D translation and orientation for object poses from the same pixel-wise dense estimation. It exploits the same voting block for inliers for multiple instances, final 3D translation estimation, and quaternions aggregation. Only the center point estimation of the objects decreases the model's running time, while still useful for occlusions and texturelessness. Symmetries and varieties are handled with a loss function based on shape matching and the pose of the object. Our proposed approach has fewer parameters and takes less time to train and evaluate, achieving great accuracy. Experiments on LINEMOD and occlusion LINEMOD datasets using ADD (-S) and 2D projection evaluation metrics show that the proposed method outperforms state-of-the-art approaches for 6D pose estimation.

Keywords 6D object pose estimation · Pixel-wise labeling · Pixel-wise center localization · Pixel-wise quaternion rotation

1 Introduction

This section introduces the proposed 6D object pose estimation approach for robot manipulation. Due to the current research progress in robotics and artificial intelligence, 6D poses have become one of the hottest research topics. It is also used for augmented reality, robot navigation, self-driving cars, bin picking, and 6D object tracking. As mentioned in the abstract section, the accuracy of 6D object pose estimation suffers from occlusion, object symmetries, and textureless surfaces of objects. Robust, efficient, and scalable end-to-end deep learning architecture is in high demand to handle these issues. We present such architecture in this research paper. Given an RGB image as input, the 6D pose of the object is a rigid body transformation from the object coordinate system $\mathbb{O}$ to the camera coordinate system $\mathbb{C}$. Here, we assume that the 3D model of the object is available and the object's coordinate system is defined in the 3D space of the model. The 6D pose, which is 3D rotation $\mathcal{R}$ and 3D translation T, is a rigid object transformation ($\mathcal{R}$; T) from the coordinates of the rigid object to the coordinates of the camera. Transformation here can be shown as a rigid transformation matrix $[\mathcal{R}, \mathrm{T}] \in \mathrm{SE}(3)$ where $\mathcal{R} \in \mathrm{SO}(3)$ and $\mathrm{T} \in \mathcal{R}^3$. We propose a network that predicts 6D poses for each object in the input RGB image. Efficient semantic segmentation has been achieved by modifying the EfficientDet [1] which is used to extract class labels and bounding boxes, but we modify it to extract semantic labels for each pixel and further calculate the objects' center points and depths which is necessary for 3D translation, and finally calculates quaternions pixel-wise for 3D rotation.

Pixel-wise labeling is used where the network classifies image pixels into object classes. Several approaches use

✉ Wu Wei
weiwu@scut.edu.cn

Faheem Ullah
ds.faheem@hotmail.com

Zhun Fan
zfan@stu.edu.cn

Qiuda Yu
yuqiuda@163.com

[1] School of Automation Science and Engineering South, China University of Technology, Guangzhou, China

[2] College of Engineering, Shantou University, Shantou, Guangdong, China

object detection with bounding boxes to detect objects for 6D pose estimation [2–4], but semantic labeling provides richer information regarding each object and handles the occlusion better. Well-known approaches for 6D object pose estimation which are based on semantic segmentation are applied by [5–8]. For further details of the backbone network, we refer readers to the EfficientDet paper [1], which has been used and modified for pixel-wise labeling for our architecture. EfficientDet uses EfficientNet [9] as the backbone network.

Figure 1 illustrates the proposed system's basic idea, which predicts semantic segmentation of the objects in the input image, the 2D pixel coordinates of the object center and its depth, pixel-wise quaternions, and finally calculates the 6D poses for objects. It creates a vector field representation for center point localization by predicting a unit vector from each pixel toward the center of the object. Using the semantic labels, image pixels are associated with an object vote on the object center location based on RANSAC [10]. Assuming known camera intrinsic parameters, the estimation of the 2D center of the object and its distance from the camera enables us to recover its 3D translation T without calculating any 2D bounding boxes for objects. It exploits the same semantic segmentation branch to estimate 3D rotation $\mathcal{R}$ by regressing convolutional features pixel-wise to quaternions directly. A similar quaternion estimation approach is used by ConvPoseCNN [5]. We will show that the 2D center point voting followed by translation estimation T and rotation regression to estimate $\mathcal{R}$ can be applied to both textured and textureless objects and is robust to occlusions as the network is trained to vote based on the centers of objects. The network uses a dense-based approach and is fully convolutional because, unlike PoseCNN, it estimates quaternions convolutionally, and the RANSAC only finds the center location inside a convolutional layer.

This end-to-end architecture achieves precise results while using much fewer parameters and results in shorter training times. Voting based on RANSAC is useful for pruning outlier predictions and we exploit this property for detecting multiple instances of the same object. Our network can handle multiple instances of the same object occurring via RANSAC inliers. Another advantage of using RANSAC is that we use a weighted RANSAC clustering technique for aggregating the selected quaternions to a final orientation estimation. Thus, the 3D translation and 3D rotation complete the process of 6D pose estimation. The proposed network contributions are given as:

- It is an accurate, efficient, end-to-end, fully convolutional, and scalable network for 6D object pose estimation which is robust to occlusions, texturelessness, and symmetries.
- It can handle multiple instances of the same object, and perform weighted RANSAC clustering for aggregating the selected quaternions to a final 3D orientation.
- It uses a modified ShapeMatch loss function by including the translation T along with the rotation $\mathcal{R}$.
- It shares parameters in several layers, so it has fewer parameters and takes less training and evaluation timing.

All the above-mentioned contributions together improve the performance of our architecture. The rest of the paper is organized into the following sections: Sect. 2, related work, presents a thorough literature review of this research. Section 3, the method used, presents the step-by-step methodology and implementation details of this research. Section 4, results and discussion, shows the experiments, qualitative and quantitative results, and discussion to prove the performance of our architecture. Finally, Sect. 5, the conclusion and future work, concludes the paper by showing future directions.

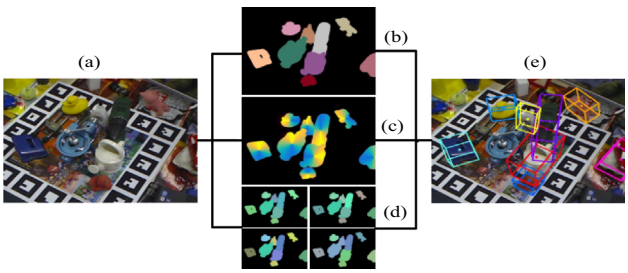


Fig. 1 **a** An RGB image passes through the 6D pose network which performs **b** pixel-wise labeling, **c** calculating objects' center points and its depths, and **d** orientation as 4-dim quaternions prediction and finally calculates **e** the 6D object poses

2 Related work

Traditionally two main approaches are used for 6D object pose estimation: the template-based [11–13] and the feature-based [14–17] approaches. Template-based methods are thought to be robust toward detecting textureless objects but struggle to handle occluded objects. The feature-based methods are thought to be robust toward occlusions but struggle to detect textureless objects. The recent deep learning-based 6D object pose estimation methods using RGB images provide end-to-end architectures. These approaches are categorized in this section in the following ways.

2.1 Direct pose regression methods

Various deep learning approaches like [18–21] regress pixels directly to the 3D object coordinates for 2D–3D correspondences, but 3D coordinate regression encounters ambiguities when dealing with symmetric objects. PoseCNN [4] also

falls in this category but it can handle occlusion and symmetry well. It uses Hough voting to estimate the 2D center location of the object. 2D object center localization can be achieved directly, however, would fail in case the object center is hidden due to occlusion. PoseCNN decouples the 6D pose estimation process and calculates the pixel-wise labeling, 3D translation, and 3D orientation separately. The orientation branch contains RoI pooling [22] for 2D bounding box estimation for objects. Several researchers use 2D object detection methods for 6D object pose estimation.

2.2 2D object detection-based methods

Although the two-staged detectors [22, 23] show more accuracy in certain circumstances, one-stage 2D object detectors such as SSD [24], YOLO [25], and EfficientDet [1] are more efficient. The method presented in [1] also handles the scalability issue well. [26] is a 2D object detection method for 6D pose estimation using deep learning. SSD6D [27] uses the idea of 2D object detection of SSD and classifies localized objects with discrete poses. YOLO6D [3] uses YOLO and detects directly the 2D projections of the 3D bounding box vertices. BB8 [2] also uses bounding boxes to detect objects for further estimating 6D object pose estimation. These 2D object detection-based methods for 6D object pose estimation are considered template matching using deep learning. The recent EfficientPose [28] extends EfficientDet for 6D object pose estimation which shows improved efficiency and also shows better accuracy on LINEMOD data using ADD (-S) evaluation metric [13].

The region of interest (RoI) pooling of the 2D object detection methods focuses on the hypotheses of the individual object at a time due to which contextual information is lost; hence, the 6D object poses estimation based on 2D object detection may lose performance in situations of occlusions. [24, 25] use fully convolutional architectures for object detection which are simpler and more efficient. Because of a fixed region size and the size-invariant property of RoI pooling, the fully convolutional architectures typically outperform RoI-based architectures in terms of model size, training, and inference time.

The fully convolutional or dense architectures replace the RoI pooling-based orientation estimation with fully convolutional and pixel-wise quaternion orientation prediction in 6D pose estimation. Although [19], which is a detection-based 6D pose estimation approach, achieves state-of-the-art accuracy and efficiency in object 6D pose estimation, the semantic segmentation-based fully convolutional architectures are robust anyway as these architectures gain more information about the scene and the objects' pixels, cut off the parameters by a huge margin, and learn the various geometric shapes of objects well.

2.3 Dense prediction methods

Unlike PoseCNN [4], PVNet [7] avoids the RoI pooled orientation prediction using 2D direction vectors to densely predict a fixed number of keypoints. Each keypoint is found using separate RANSAC-based voting followed by estimating the final pose using a perspective-n-points (PnP) solver utilizing the known keypoints' 2D–3D correspondences. DPVL [8] and PVDRL [29] improved the results by further considering the distance between pixel and keypoint which is useful for the selection of accurate hypotheses in the RANSAC process and it avoids the deviation of hypothesis due to direction unit vectors errors when a pixel is far from a keypoint. ZebraPose [30] also uses a dense approach with a PnP solver. [31] make the 6D poses of several cameras and the template model optimal where a strong RANSAC-based selection process chooses well-matched 2D–3D feature points in the discrete stage of their discrete–continuous optimization in accordance with the current model/camera poses. [32] estimate 3D object poses without correspondence from noisy depth data, and [33] introduce a novel issue that predicts poses from 3D skeleton sequence positions. By taking advantage of spatial–temporal linkages for pose estimation using graph convolutional networks, Cai et al. [34] incorporated spatial dependencies and temporal consistency.

Other recent 6D objects pose estimation methods using RGB [3, 19–21, 35] to achieve state-of-the-art accuracy which applies a 2D keypoints detection approach on the objects' surfaces and further solves the PnP problem for the final 6D object poses. A number of these approaches are also efficient when estimating the pose of a single object, but their runtime increase linearly when the number of objects increases due to solving the perspective-n-points (PnP) problem for each object individually for the final 6D object poses.

The proposed approach in this research paper makes dense predictions for an object's center point localization. It is a hybrid of dense prediction and keypoint-based, which combines the advantages of both of these methods, hence useful in the occluded environment and can handle symmetric, textured, and textureless objects. The 3D translation and orientation are obtained from the same pixel-wise estimation. It is fully convolutional and exploits the same voting block for inliers for multiple instances and quaternions aggregation, hence very accurate and efficient. Our method calculates orientation regression directly. The additional overhead of RANSAC or Hough transforms and PnP solving is not needed, hence useful in resource-constrained settings. PoseCNN first predicts the translation and RoI and finally, for each RoI, it estimates object orientation sequentially. Our proposed architecture is unified and more parallel which can parallelly perform pixel-wise labeling, vector field prediction, and unit orientation quaternions, while PoseCNN and ConvPoseCNN [5] predict all of these in separate branches.

ConvPoseCNN uses the Hough layer for object center location estimation and handling multiple instance occurrence of the same object using Hough inliers and then uses weighted RANSAC clustering for aggregating the selected quaternions to a final orientation estimation. Our architecture takes advantage of the RANSAC layer for handling all the above three tasks. Our architecture is capable of simultaneously and independently performing the rotation estimation for several objects, which means that it runs parallelly with translation estimation and is not dependent on translation. Using a clustering method, we convert our pixel-wise estimation into a final orientation. We demonstrate how our pixel-wise prediction from fully convolutional architecture provides accurate results while utilizing significantly fewer parameters.

3 Method

The proposed network predicts for each pixel, the labels and unit vectors pointing to the center point of the object. Unlike PVNet [7] which predicts 8 keypoints on the object's surface, our network similar to PoseCNN [4] considers and predicts only the center point of the object. The segmentation and the vector field together vote for the center position and depth of the object in a RANSAC layer and estimate the 3D translation. Our network estimates quaternions fully convolutionally and pixel-wise, parallel to 3D translation, and estimates the final orientation. Quaternions are regressed directly using a linear output layer. The pixel-wise labeling, pixels voting for a central point of the object with RANSAC voting for 3D translation, and quaternions for 3D rotation, all use the same convolutional layers and share the same parameters. Fully convolutional layers have fewer parameters than fully connected layers, and hence, our architecture is more lightweight than PoseCNN and ConvPoseCNN. The following subsections explain the process step by step and Fig. 2 presents the network architecture.

3.1 Semantic labeling of object pixels

While EfficientDet model is primarily intended for object detection, we modified it for semantic segmentation tasks. We adopt the EfficientDet [1] model and added the P_2 layer to its BiFPN to preserve feature levels from P_2 to P_7 in BiFPN. We combine and fuse all from P_2 to P_7 of the BiFPN output for the final per-pixel classification. EfficientDet exploits the EfficientNet's [9] multi-scale feature fusion. The multi-scale features fusion was used by Lin et al. [36]. Inspired by these approaches, we also fuse multi-scale features efficiently after which the upsampling is performed to achieve the pixel-wise labeled image of the input size. We use the EfficientDet-D4 model with an EfficientNet-B4 backbone which is pre-trained on ImageNet. In this case, it has a size similar to ResNet-50. All the levels of BiFPN are up-sampled and brought at the same scale feature maps which are then concatenated channel-wise. Then, transposed convolution is used for upsampling followed by batch normalization and the swish activation [37] to transform the added feature maps to the original image size. This way the prediction has a direct relation to each pixel of the input image. For final pixel-wise labeling, three blocks of depth-wise convolutions [38] are applied again followed by the batch norm and the swish activation. Finally, a further convolution is applied to reduce channels to the number of classes. The softmax cross-entropy loss is applied for training semantic labels.

To the final feature map, a 1×1 convolution is also applied to obtain the unit vectors along with class probabilities and also predicts unit orientation quaternions pixel-wise. So there are three output values of the base network, i.e., pixel-wise labeling, unit vectors, and unit quaternions.

3.2 3D translation network

The network predicts the 2D center point $\mathcal{C} = (c_x, c_y)^T$ of the object in pixel coordinates and then the distance or depth T_z from the camera to the object. Adopting the PoseCNN approach, we do not regress the image features directly to the 3D translation vector $\mathrm{T} = (T_x, T_y, T_z)^T$. Regressing the image features directly to T is not a generalizable approach and cannot handle multiple instances of the same object, hence it is avoided. After localizing the center $\mathcal{C}$ in the image, estimating the depth T_z and the camera intrinsic parameters, the T_x and T_y can be recovered via the projection equation of a pinhole camera as follows:

$$\begin{bmatrix} T_x \\ \\ T_y \end{bmatrix} = \begin{bmatrix} \frac{(c_x - p_x).T_z}{f_x} \\ \frac{(c_y - p_y).T_z}{f_y} \end{bmatrix} \tag{1}$$

where $\mathcal{P} = \left(p_x, p_y\right)^T$ is the principal point, while f_x and f_y are called focal lengths. Here, $\mathcal{C}$ is the 2D center of the object if the object origin $\mathcal{O}$ is the centroid of the object.

The pixels of the object vote for the 2D center location of the object where for each pixel, a unit vector is calculated pointing to the center direction. A voting procedure based on RANSAC is used that takes as input, the pixel-wise semantic labeling and the center regression results. After obtaining voting scores, the location with the maximum score is selected as the center of the object and calculates its depth. We do not calculate 2D bounding boxes for objects. 2D object center localization can be achieved directly, however, would fail in case the object center is hidden due to occlusion.

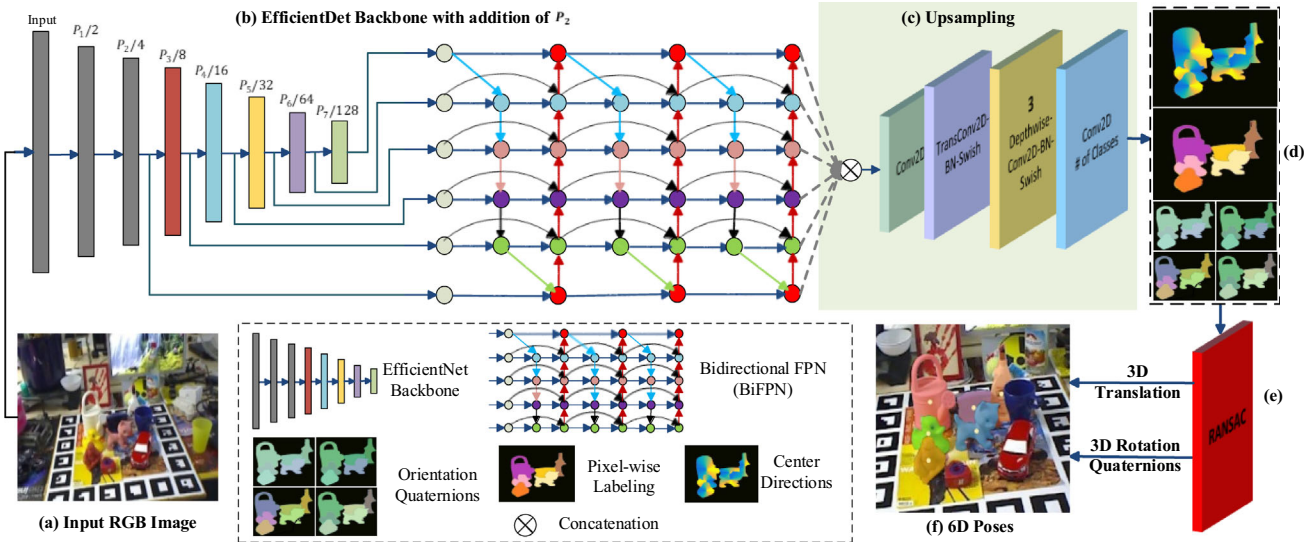


Fig. 2 The proposed end-to-end architecture for 6D objects poses estimation is presented in the figure. EfficientDet (**b**) uses EfficientNet as a backbone with the BiFPN layer for object detection. We modify it for semantic segmentation by adding the P_2 layer to the BiFPN and concatenate all the output values of the BiFPN which passes to the (**c**) upsampling module to achieve the original image size back. This network produces output as pixel-wise labeling (semantic segmentation) for each object, a unit vector field pointing towards the center of each object, and unit orientation quaternions. The RANSAC (**e**) decides the center point for each object and calculates the distance or depth from the camera to the object which then calculates 3D translation. Finally, weighted RANSAC clustering predicts orientation quaternion pixel-wise for each object, i.e., 3D rotation. The 3D translation and 3D rotation together complete the process of 6D pose estimation (**f**)

This RANSAC layer also helps in clustering the selected quaternions to a final estimation of orientation during the 3D rotation presented in Sect. 3.3.

Once a set of object centers is generated, the pixels that vote for the center of an object are considered the inliers for the center, and thus, the depth of the center T_z is calculated to be the average of the depths estimated by the inliers. Finally, using Eq. (1) and the depth information, the 3D translation T is estimated.

3.3 3D rotation network

Our end-to-end architecture predicts orientation quaternions in a fully convolutional way using the predicted pixel-wise segmentation to identify which quaternions belong to which object. Quaternions are regressed directly using the same architectural parameters of the segmentation branch. This has several advantages over [39] and [4] architectures, i.e., avoiding fully connected layers and the extra dense prediction branches for objects center position with depths and orientation quaternions. Additionally, we use RANSAC inliers for handling the occurrence of multiple instances of the same object. The same RANSAC voting layer which is used for translation estimation has been exploited for separating for each object hypothesis, the predictions into inlier sets.

To aggregate the selected quaternions to a final orientation estimation, we use a weighted RANSAC clustering technique defined by ConvPoseCNN [5] as this approach is less affected by outlier predictions; hence less chance to suffer from skewed results. The weighted RANSAC clustering technique is given as:

Assume that $\mathcal{Q} = \{\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3, \ldots, \mathcal{Q}_n\}$ are the quaternions and $\mathcal{W} = \{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3, \ldots, \mathcal{W}_n\}$ are the weights for these quaternions for all objects in an image. So, $\mathcal{Q}_i = \{q_1, q_2, q_3, \ldots, q_n\}$ be the quaternions and $\mathcal{W}_i = \{w_1, w_2, w_3, \ldots, w_n\}$ be the weights of these quaternions associated with a single object. Considering a specific object, the algorithm iteratively selects a random quaternion $q_r \in \mathcal{Q}_i$ with a probability proportional to its weight and then determines the inliers set $\mathcal{Q}_{in}$ as $\mathcal{Q}_{in} = \{q_i \in \mathcal{Q}_i | d(q_i, q_r) < \theta\}$, where $d(q_i, q_r)$ is the angular distance between a quaternion and random quaternion. Finally, the q_r with the largest $\sum_{q_i \in \mathcal{Q}_{in}} w_i$ is selected as the resultant quaternion.

Our architecture takes advantage of the RANSAC layer for object center locations estimation, handling multiple instances of occurrence of the same object using RANSAC inliers, and aggregating the selected quaternions to a final orientation estimation via weighted RANSAC clustering technique. Therefore, ConvPoseCNN [5] uses the Hough layer for the first two tasks and uses weighted RANSAC clustering for the third task. This difference is shown in Fig. 3.

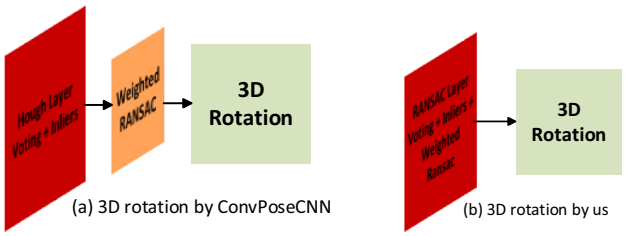


Fig. 3 Shows the difference between the 3D orientation estimation of ConvPoseCNN and our proposed method. ConvPoseCNN uses Hough inliers and then applies weighted RANSAC, while our method uses the RANSAC inliers and hence removes the extra step

3.4 Loss function

We use the ShapeMatch loss ($\mathcal{SM}Loss$) proposed by Xiang et al. (2018) [4] for rotation training that calculates the distance between 3D points set of the object rotated by the predicted quaternions $\tilde{q}$ and the ground truth quaternions q and handles symmetric and non-symmetric objects. The $\mathcal{SM}Loss$ is based on two loss functions, i.e., the pose loss ($\mathcal{P}Loss$) and the symmetric loss ($\mathcal{S}Loss$).

$$\mathcal{SM}\text{Loss}(\tilde{q}, q) = \begin{cases} \mathcal{S}\text{Loss}(\tilde{q}, q) & \text{if symmetric,} \\ \mathcal{P}\text{Loss}(\tilde{q}, q) & \text{otherwise} \end{cases} \tag{2}$$

To turn the loss function to transform the object of interest with the estimated and the ground truth 6D pose, and then computing the average point distances between the transformed model points, we add the translation T also along with the rotation $\mathcal{R}$ in both the $\mathcal{P}$Loss and $\mathcal{S}$Loss functions. This strategy is useful to immediately tune the model on the performance metric similar to ADD and ADD (-S) metrics (the ADD and ADD (-S) metrics are presented in Sect. 4.2). It is also useful in avoiding the need for an extra hyperparameter for balancing the partial losses when calculating the rotation and translation losses independently. The $\mathcal{P}$Loss and $\mathcal{S}$Loss functions are given below in Eqs. 3 and 4, respectively.

$$\mathcal{P}\text{Loss}(\tilde{q}, q) = \frac{1}{2m} \sum_{x \in \mathcal{M}} \left(\mathcal{R}(\tilde{q})x + \tilde{\mathrm{T}}\right) - (\mathcal{R}(q)x + \mathrm{T})^2 \tag{3}$$

$$\mathcal{S}\text{Loss}(\tilde{q}, q) = \frac{1}{2m} \sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} \left(\mathcal{R}(\tilde{q})x_1 + \tilde{\mathrm{T}}\right) - (\mathcal{R}(q)x_2 + \mathrm{T})^2 \tag{4}$$

where $\mathcal{M} = \{ x_i \in \mathcal{R}^3 | i = 1, 2, 3, \ldots, m\}$ is the 3D model points set and m the number of points. $\mathcal{R}(\tilde{q})$ is the rotation matrix computed from the estimated quaternion and $\mathcal{R}(q)$ is the rotation matrix computed from the ground truth quaternion. Here, we use two loss functions, the $\mathcal{P}$Loss and the $\mathcal{S}$Loss, as the $\mathcal{P}$Loss performs well only for non-symmetric objects but is not suitable for handling symmetric objects properly because it penalizes the rotations that are equal in terms of the object's 3D shape symmetry. The $\mathcal{S}$Loss function avoids this issue and is suitable for handling symmetric objects. It takes into account the shortest mean distance for each point to any point in the other transformed point set like ADD (-S) metric in Sect. 4.2, rather than just measuring the distance between the matching points of the two transformed point sets [29]. Again this is useful to train and optimize the model on the ADD (-S) like performance metric which helps prevent unnecessary penalties during training when dealing with symmetric objects.

$\mathcal{Q}$Loss is a loss function originally used by SilhoNet [40], also known as pixel-wise ℓ_2 has been exploited for efficiency reasons for orientation regression which is the log distance function between the predicted and ground truth quaternions. Calculating the $\mathcal{Q}$Loss between two quaternions $\tilde{q}$ and q given in [40] is as follows:

$$\mathcal{Q}\text{Loss}(\tilde{q}, q) = \log(\varepsilon + 1 - |\tilde{q}.q|) \tag{5}$$

where ε is the stability constant having a small value which is e^{-4} in this case. The total loss is achieved by the linear combination of all losses as follows:

$$\mathcal{L} = \alpha \mathcal{L}_{\text{seg}} + \beta \mathcal{L}_{\text{T}} + \gamma \mathcal{L}_{\mathcal{R}} \tag{6}$$

where $\mathcal{L}$ is a total loss, $\mathcal{L}_{\text{seg}}$, $\mathcal{L}_{\text{T}}$, and $\mathcal{L}_R$ are the segmentation, translation, and rotation losses, respectively, while α, β, and γ are the normalization factors and are hyperparameters. A momentum-based SGD has been used for training the total loss.

3.5 Implementation details

As the LINEMOD and Occlusion LINEMOD datasets have limited annotated data due to which the network leads to overfitting. To avoid overfitting, we apply augmentation to the dataset. The processing performed includes rotation, horizontal flip, color jittering, width and height shift, zoom, channel shift, and shear. For each object, we synthesized 10,000 images and rendered 10,000 images similar to PVNet [7] and DPVL [8]. We use a momentum-based SGD for training with a learning rate of 0.001 and momentum of 0.9, for the total loss $\alpha = \beta = 1$, for the $\mathcal{Q}$Loss, the $\gamma = 1$, and for the $\mathcal{SM}$Loss, the $\gamma = 100$. The model has been trained for 200 epochs. We trained our network on the LINEMOD dataset and performed no post-refinement operations. Our method achieves real-time performance on a GTX 2080 Ti GPU which is 27 fps.

4 Results and discussions

The experiments, results, comparisons, and discussions of our method against other related well-known methods for 6D object pose estimation are explored and presented in this section using state-of-the-art datasets and evaluation metrics for evaluating this architecture.

4.1 Datasets

All experiments are performed on the two main benchmarking datasets for 6D object pose estimation LINEMODE and OCCLUSION LINEMOD [41].

4.1.1 LINEMOD

This dataset has 15,783 images of 13 objects, and each object has about 1200 images with masks. For each object, there is also provided a high-quality 3D model. It contains images of small objects in a cluttered scene under different illumination conditions.

4.1.2 OCCLUSION LINEMOD

Similarly, the Occlusion LINEMOD dataset is a subset of the LINEMOD dataset that contains only the objects that are occluded. It consists of a total of 1214 images of only 8 objects with occlusions among them which is more challenging. A number of research studies have reported these datasets for comparative analysis of the 6D object pose estimation.

4.2 Evaluation metrics

For evaluation purposes, we use the ADD metric [13] which is a standard metric for calculating the average Euclidean distance between predicted and ground truth pose using the transformed 3D model points, and the 2D projection metric [39] which measures in the pixel domain, the closeness of the 2D projected vertices to the ground truth.

4.2.1 ADD metric

Considering that the ground truth translation T and rotation $\mathcal{R}$, and the estimated translation $\widetilde{\mathrm{T}}$ and rotation $\widetilde{\mathcal{R}}$ are provided, the average of the pairwise distances between the 3D model points transformed according to the ground truth pose, and the estimated pose is computed using the average distance as:

$$\mathrm{ADD} = \frac{1}{m} \sum_{x \epsilon \mathcal{M}} \|(\mathcal{R}x + \mathrm{T}) - (\widetilde{\mathcal{R}}x + \widetilde{\mathrm{T}})\|_2 \tag{7}$$

where m is the number of points and $\mathcal{M}$ is the set of 3D model points. The correct pose is considered if the ADD score is $< 10\%$ of the target diameter of the object. This is the predefined threshold with which the average distance is compared. But due to the rotational invariant nature of the symmetric objects regarding the appearance such as "Egg box" and "Glue" in the LINEMOD and occlusion LINEMOD datasets, the ADD metric penalizes these objects. Therefore, the ADD (S), which is the mean distance using the closest point distance, is calculated for handling symmetric objects:

$$\mathrm{ADD}(S) = \frac{1}{m} \sum_{x_1 \epsilon \mathcal{M}} min_{x_2 \in \mathcal{M}} \|(\mathcal{R}x_1 + \mathrm{T}) - \left(\widetilde{\mathcal{R}}x_2 + \widetilde{\mathrm{T}}\right)\|_2 \tag{8}$$

4.2.2 2D Projection metric

The 2D projection metric measures the closeness between the 2D projected vertices in the pixel domain and the ground truth. The correct pose is considered if the mean 2D projection error is $< 5px$.

$$2D.\mathrm{Proj} = \frac{1}{m} \sum_{x \epsilon \mathcal{M}} \|\mathcal{K}(\mathcal{R}x + \mathrm{T}) - \mathcal{K}(\widetilde{\mathcal{R}}x + \widetilde{\mathrm{T}})\|_2 \tag{9}$$

where m is the total number of points on the 3D object model $\mathcal{M} = \{ x_i \in \mathcal{R}^3 | i = 1, 2, 3, \ldots, m\}$, x is a point or a set of points on the surface of the 3D object model, $\mathcal{R}$ and T are the rotation and translation, respectively, $\mathcal{R}x + \mathrm{T}$ is the target pose that transforms the point with SE(3) transformation and vice versa, and $\mathcal{K}$ is the intrinsic parameter of camera.

4.3 Comparisons with state-of-the-art

Comparisons of our proposed method with the state-of-the-art methods of 6D object poses based on RGB input are presented in this section and the qualitative results of our method are also presented. All the comparisons are carried out against PVNet [7], PoseCNN [4], SSD6D [27], YOLO6D [3], BB8 [2], CDPN [19], DPOD [21], Pix2Pose [20], CSA6D [42], PVDRL [29], RePOSE [43], ZebraPose [30], BiCo-Net [48], and EfficientPose [28]. The ADD (-S) and 2D-projection metrics are used for the evaluation of the results on LINEMOD and occlusion LINEMOD datasets. Since our proposed method estimates accurate poses of objects from an RGB image, any pose refinement methods are not adopted for further improvement of the 6D object pose estimation results. As certain methods do not report 2D-projection-based results, those are not included in Table 2.

4.3.1 Comparisons on LINEMOD Dataset

Table 1 presents results on the LINEMOD dataset using ADD (-S) metric and shows the performance of our proposed method compared to the previous approaches. BB8 uses as keypoints the eight corners of the 3D bounding box and the object center and regresses their coordinates directly. The YOLO6D and SSD6D also localize keypoints via regression. DPOD, Pix2Pose, and CDPN regress 3D object coordinate. PVNet and PVDRL apply a voting strategy, localizing eight keypoints on the surface of objects and also the center of each object from the predicted vector fields. DPOD regresses dense 2D–3D correspondences.

PoseCNN, ConvPoseCNN, EfficientPose, and our proposed method estimate the center point of the objects only. PoseCNN first localizes object centers via Hough voting and then estimates poses by regression. ConvPoseCNN uses a dense approach for 3D orientation while PoseCNN calculates 2D boxes and then a fully connected network. EfficientPose uses a detection-based approach. Our method is more parallel and uses a dense approach for 3D translation and orientation. Note that BB8, DPOD, SSD6D, and PoseCNN require further refinement of the estimated object poses. In the tables, only the refined results of these approaches are shown. The ConvPoseCNN paper does not report results on LINEMOD and occlusion LINEMOD datasets, so we do not use its results in comparisons. ZebraPose [30] also does not report results on LINEMOD, they only report on occlusion LINEMOD. Table 1 shows that using the LINEMOD dataset, our proposed method outperforms most of the objects applying ADD (-S) scores. In comparison with the state-of-the-art methods RePOSE and EfficientPose, our method successfully improves the performance of the initial pose estimation on average 1.52% and 0.27% accuracy performance, respectively. The EfficientPose has reported very high results on the LINEMOD dataset using ADD (-S) with improved efficiency and our proposed method performs well by a proportion in terms of accuracy; however, our approach outperforms in terms of accurate pose estimation of most of the object classes in LINEMOD which mean that our approach can accurately detect a higher number of objects than EfficientPose. As can be seen in Table 1, the RePOSE and EfficientPose show better results on 5 classes and 4 classes, respectively, while our approach on 6 classes which are (Ape, Cam, Cat, Duck, Egg box, and Phone). This achievement is necessary for real-world applications as we need our systems to accurately perform over more and more objects. Accuracies for textureless objects like "Ape" are improved by 8.28% and 0.07%, and for "Duck" are improved by 11.7% and 1.01% against RePOSE and EfficientDet, respectively, and achieve maximum accuracy for symmetric objects like "egg box" which is 100% although SSD6D and EfficientDet outperforms in terms of "Glue." The graphs (a) and (b) in Fig. 5

Table 1 The performance of the LINEMOD dataset for object pose estimation based on ADD (-S) score. Objects like Glue and Egg Box are symmetric objects, Ape and Duck are textureless objects

Methods	BB8 [2]	SSD6D [27]	YOLO6D [3]	DPOD [21]	Pix2Pose [20]	CDPN [19]	PoseCNN [4]	PVNet [7]	PVDRL [29]	RePOSE [43]	EfficientPose [28]	Ours
Ape	40.4	65	21.62	53.28	58.1	64.38	27.8	43.62	76.27	79.5	87.71	**87.78**
Bench vise	91.8	80	81.8	95.34	91	97.77	68.9	99.9	**100**	**100**	99.71	99.20
Cam	55.7	78	36.57	90.36	60.9	91.67	47.5	86.86	96.8	99.2	97.94	**99.32**
Can	64.1	86	68.8	94.1	84.4	95.87	71.4	95.47	99.38	**99.8**	98.52	98.63
Cat	62.6	70	41.82	60.38	65	83.83	56.7	79.34	87.85	97.9	98.00	**98.22**
Driller	74.7	73	63.51	97.72	73.6	96.23	65.4	96.43	99.4	99.0	**99.90**	99.30
Duck	44.3	66	27.23	66.01	43.8	66.76	42.8	52.58	71.42	80.3	90.99	**92.00**
Egg box	57.8	100	69.58	99.72	96.8	99.72	98.3	99.15	**100**	**100**	**100**	**100**
Glue	41.2	**100**	80.02	93.83	79.4	99.61	95.6	95.66	99.68	98.3	**100**	99.96
Hole puncher	67.2	49	42.63	65.83	74.8	85.82	50.9	81.92	94.72	**96.9**	95.15	96.23
Iron	84.7	78	74.97	99.8	83.4	97.85	65.6	98.88	**100**	**100**	99.69	99.46
Lamp	76.5	73	71.11	88.11	82	97.86	70.3	99.33	99.92	99.8	**100**	99.86
Phone	54	79	47.74	74.24	45	90.75	54.6	92.41	98.64	98.9	97.98	**99.20**
Mean	62.7	79	55.95	82.98	72.4	89.86	62.7	86.27	94.16	96.1	97.35	**97.62**

The bold values indicate the highest score for the different methods given at different objects in the dataset

Table 2 The performance of the LINEMOD dataset for object pose estimation based on 2D projection score

Methods	BB8 [2]	YOLO6D[3]	CDPN [19]	PoseCNN [4]	PVNet [7]	DPVL [8]	PVDRL [29]	CSA6D [42]	Ours
Ape	96.6	92.1	96.86	83	99.23	99.04	99.42	98.60	**99.54**
Bench vise	99.1	95.06	98.35	50	99.81	99.71	99.83	95.80	**99.88**
Cam	86	93.24	98.73	71.9	99.21	99.41	99.55	98.80	**99.65**
Can	91.2	97.44	99.41	69.8	99.9	**100**	**100**	97.40	99.95
Cat	98.8	97.41	99.8	92	99.3	99.7	99.86	99.50	**99.89**
Driller	80.9	79.41	95.34	43.6	96.92	98.12	98.83	95.10	**98.88**
Duck	92.2	94.65	98.59	91.8	98.02	99.06	99.59	98.40	**99.72**
Egg box	91	90.33	98.97	91.1	99.34	99.43	99.84	99.90	**99.97**
Glue	92.3	96.53	99.23	88	98.45	99.51	99.86	99.90	**99.96**
Hole puncher	95.3	92.86	99.71	82.1	**100**	**100**	**100**	98.20	**100**
Iron	84.8	82.94	97.24	41.8	99.18	99.69	99.91	97.80	**99.93**
Lamp	75.8	76.87	95.49	48.4	98.27	99.14	99.53	97.50	**99.68**
Phone	85.3	86.07	97.64	58.8	99.42	99.42	99.65	97.60	**99.76**
Mean	89.3	90.37	98.1	70.2	99	99.4	99.68	98.10	**99.75**

Objects like Glue and Egg Box are symmetric objects, Ape and Duck are textureless objects
The bold values indicate the highest score for the different methods given at different objects in the dataset

show the accuracies of textureless and symmetric objects on LINEMOD using ADD (-S). REDE [44], MixedFusion [45], and PVN3D [46] on LINEMOD with ADD (-S) achieve 98.9, 97.8, and 99.4%, respectively, but they all require depth information. [47] achieves 97.5% but using a different approach that combines two RGB for pose estimation while we estimate pose from a single RGB. Table 2 shows that our proposed method also outperforms state-of-the-art approaches on LINEMOD data in terms of 2D projection metric (except class "Can"). EfficientDet and RePOSE both do not report results using the 2D projection matrix, so we do not add those results to Table 2, instead, we include CSA6D [42] as it reports only 2D projection-based results on the LINEMOD dataset. Again the accuracies for textureless objects like "Ape" are improved by 0.12 and 0.94%, and for "Duck" 0.13 and 1.32% against PVDRL and CSA6D, respectively. Using the 2D projection metric on the LINEMOD dataset achieves better accuracy for symmetric objects also such as "Egg box" which are 0.13 and 0.07%, and for the "Glue" which are 0.1 and 0.06% against PVDRL and CSA6D, respectively. Our architecture on average of 0.07 and 1.65% outperforms PVDRL and CSA6D, respectively, based on the 2D projection metric. The total number of wins by our proposed architecture is 12 out of 13 object classes. The number of wins is the classes where our architecture shows higher accuracy compared to the state-of-the-art. Again the # of wins which is 12 in our case are the classes where our architecture shows higher accuracy compared to the recent related state-of-the-art methods. The graphs (c) and (d) in Fig. 5 show the accuracies of textureless and symmetric objects on LINEMOD using 2D projection.

4.3.2 Comparisons on occlusion LINEMOD dataset

The results of our method compared to the state-of-the-art approaches on LINEMOD using Add (-S) are presented in Table 3 which shows that our method compared to the state-of-the-art approaches achieves the best overall average performance of 79.57%, which outperforms state-of-the-art BiCo-Net [48], ZebraPose [30], and EfficientPose [28] by 10.07, 2.65, and 0.53%, respectively. In terms of textureless objects "Ape" and "Duck," and symmetric objects like "Egg box" and "Glue," the ZebraPose and our methods are comparable. ZebraPose outperforms our method in terms on "Ape" by 0.26% while our method outperforms ZebraPose by 4.71% and EfficientPose by 3.9%. In terms of "Egg box," our method outperforms ZebraPose by 24.13% and EfficientDet by 1.57%, but when it comes to "Glue," ZebraPose outperforms slightly our method which is 2.86%. EfficientPose performs well on "Cat" and "Driller" while Bico-Net on "Hole puncher." To see visually how accurate our network is, the qualitative results on occlusion data are given in Fig. 4, and graphs (e) and (f) in Fig. 5 show the accuracies of textureless and symmetric objects on occlusion LINEMOD using ADD (-S) metric.

Table 3 The performance of the Occlusion LINEMOD dataset for object pose estimation based on ADD (-S) score

Methods	YOLO6D [3]	Pix2Pose [20]	PoseCNN [4]	PVNet [7]	PVDRL [29]	RePOSE [43]	BiCo-Net [48]	ZebraPose [30]	EfficientPose [28]	Ours
Ape	2.48	22.00	9.60	15.81	22.44	31.10	55.6	**57.90**	56.57	57.64
Can	17.48	44.70	45.20	63.3	73.31	80.00	83.2	**95.00**	91.12	90.84
Cat	0.68	22.70	0.93	16.68	24.23	25.60	47.3	60.60	**68.58**	68.11
Driller	7.66	44.70	41.40	65.65	75.07	73.10	69.9	94.80	**95.64**	93.70
Duck	1.14	15.00	19.60	25.24	38.60	43.00	58.3	64.50	65.31	**69.21**
Egg box	–	25.20	22.00	50.17	51.43	51.70	78.1	70.90	93.46	**95.03**
Glue	10.08	32.40	38.50	49.62	44.08	54.30	76.9	**88.70**	85.15	85.84
Hole puncher	5.45	49.50	22.10	39.67	50.11	53.60	**87.2**	83.00	76.53	76.20
Mean	6.42	32.00	24.90	40.77	47.40	51.60	69.5	76.92	79.04	**79.57**

Objects like Glue and Egg Box are symmetric objects, Ape and Duck are textureless objects
The bold values indicate the highest score for the different methods given at different objects in the dataset

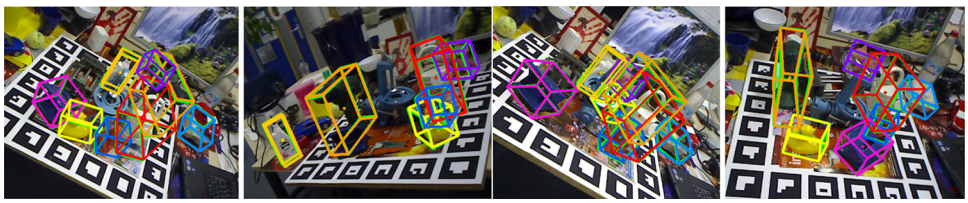


Fig. 4 The qualitative results on occlusion LINEMOD dataset. The ground truths are shown in green 3D bounding boxes and the predictions of the 3D bounding boxes around the objects are shown other colors

4.4 Time comparison

The training of the proposed model is faster and takes only 200 loops/epochs to train. Our network takes an 480 × 640 image as input and runs at 27 fps on a GTX 2080 Ti GPU. PVNet also takes 200 loops to train and runs at 25 fps using similar computing power, while EfficientDet is trained for 5000 epochs. On the occlusion dataset, our network runs on an average of 26 FPS where there are multiple object pose estimations. This speed is real-time efficient for 6D object pose estimation. This efficiency is because of the fewer FLOPs of the EfficientDet architecture and the fully convolutional and parallel architecture of our proposed network.

4.5 Ablation study

Although the two-stage methods produce good results for objects poses, our proposed method is more robust in terms of accuracy and efficiency, as during the two-stage pose estimation methods, solving the perspective-n-points (PnP) problem for several keypoints and then for many objects is an expensive task. Results presented in Sect. 4 show the performance of our method.

RANSAC is a slower process than Hough voting but here if we use the Hough layer for objects center locations estimation, handling multiple instances occurrence of the same object using Hough inliers, and then using weighted RANSAC clustering for aggregating the selected quaternions to a final orientation estimation, become time-consuming. Instead, we use the same RANSAC layer for all these three tasks, which decreases the running time. Our dense architecture outperforms RoI-based architectures in terms of model size, training, and inference time. Each hierarchical layer in our proposed network shares a set of parameters, which makes it lightweight and flexible, and because of its flexibility, it may be incorporated into any encoder–decoder network to reduce model size. More details are given in Sects. 2.2 and 2.3. The weighted RANSAC performance is better than the non-weighted one for the estimation of aggregating the selected quaternions to a final orientation presented in Sect. 3.3. Calculating the $\mathcal{SM}Loss$ pixel-wise is not computationally possible so to make training possible with $\mathcal{SM}Loss$, we first aggregate the dense predictions and then calculate the orientation loss presented in Sect. 3.4. For the $\mathcal{SM}Loss$, via its variations, we performed some experiments. We tested the results using the loss similar to PoseCNN which considers only rotations and then we tried using our new loss that takes into account both the translation and rotation and observed the results presented in Table 4 using occlusion LINEMOD and ADD (-S). Our network model uses far lesser parameters which are less than half of the total parameters of ConvPoseCNN. One reason that ConvPoseCNN has many parameters is because of the VGG16 [49] backbone which has 138 million parameters in total.

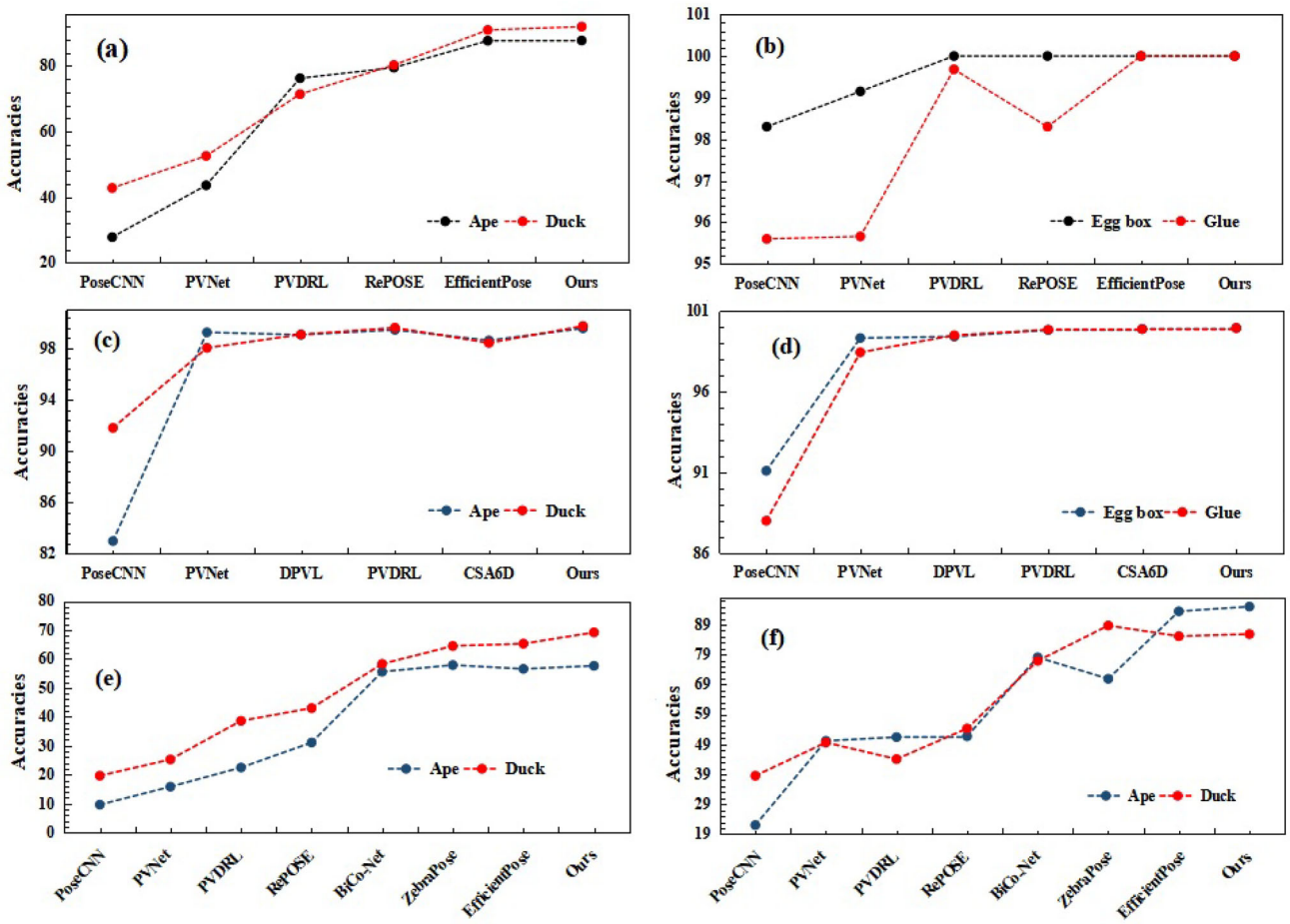


Fig. 5 **a** Accuracies of textureless objects Ape and Duck using LINEMOD dataset and ADD (-S) metric, **b** accuracy of symmetric objects Egg box and glue using LINEMOD and ADD (-S), **c** textureless objects using LINEMOD and 2D projection, **d** symmetric objects using LINEMOD and 2D projection, **e** textureless objects using occlusion LINEMOD and ADD (-S), and **f** is the accuracies of symmetric objects using occlusion LINEMOD and ADD (-S)

Table 4 Comparison on the Occlusion LINEMOD dataset for objects pose estimation based on ADD (-S) scores with and without translation T in $\mathcal{SM}Loss$

Methods	Ours without T in $\mathcal{SM}Loss$	Ours with T in $\mathcal{SM}Loss$
Ape	57.00	**57.64**
Can	90.80	**90.84**
Cat	67.67	**68.11**
Duck	67.98	**69.21**
Hole puncher	75.04	**76.20**

The bold values indicate the highest score for the different methods given at different objects in the dataset

We use a modified version of EfficientDet which is based on EfficientNet-B4 with BiFPN which has about 17 M parameters. EfficientDet-D4 has about 21 M parameters which are comparable to ResNet-50. Another reason that our model has fewer parameters is unlike ConvPoseCNN which estimates all three modules, i.e., pixel-wise labeling, 3D translation, and 3D rotation separately through separate convolutional branches, our network exploits the same base model's output for all three modules which share a lot of parameters and also decreases the calculation time.

EfficientPose evaluates its results using a scaling factor using a single hyperparameter ϕ and presents results with two different values of ϕ which are $\phi = 0$ and $\phi = 3$. We only consider the scaling factor $\phi = 0$ for this research, so we compare our results with the model hyperparameterized with $\phi = 0$ based results of EfficientDet. Our model will produce more accurate results at $\phi = 3$ scaling. In Table 5, we can see that EfficientPose with $\phi = 3$ outperforms both EfficientPose with $\phi = 0$ and our proposed network, but in this case, it loses its efficiency. On occlusion LINEMOD dataset for objects pose estimation based on ADD (-S), they achieve 83.98%

Table 5 Comparison of our proposed network against EfficientDet with $\phi = 0$ and $\phi = 3$ on the Occlusion LINEMOD dataset for object pose estimation based on ADD (-S) scores

Methods	EfficientPose with $\phi = 0$	Ours with $\phi = 0$	EfficientPose with $\phi = 3$
Ape	56.57	**57.64**	59.39*
Can	**91.12**	90.84	93.27*
Cat	**68.58**	68.11	79.78*
Driller	**95.64**	93.70	97.77*
Duck	65.31	**69.21**	72.71*
Egg box	93.46	**95.03**	96.18*
Glue	85.15	**85.84**	90.80*
Hole puncher	**76.53**	76.20	81.95*
Mean	79.04	**79.57**	83.98*

Some objects like Glue and Egg Box are symmetric objects, Ape and Duck are textureless objects
The bold values indicate the highest score for the different methods given at different objects in the dataset
* shows the results of EfficientPose with hyperparameter $\varphi = 3$

results. We could not conduct experiments with $\phi = 3$, but we believe that our proposed system with $\phi = 3$ will outperform EfficientPose with $\phi = 3$. The values with the stars (*) show the highest values between EfficientDet at $\phi = 3$ compared to EfficientPose with $\phi = 0$ and our method with $\phi = 0$.

5 Conclusion

The base network of the proposed architecture predicts pixel-wise labeling of objects, pixel-wise unit vector field toward the center of the objects, and pixel-wise quaternions, so we can identify quaternions related to each object at the same place, and then can determine the final quaternion from those quaternions for each object. The limitation of our method is that although it beats the EfficientDet in terms of both the number of wins and overall average but it could improve the overall average by a small margin compared to EfficientDet. As a whole our method performs well.

Applying post-refinement over this proposed method will improve the results further in terms of accuracy. Furthermore, various experiments can be carried out over orientation quaternions estimation using various techniques to check their effectiveness. Similarly, Hough voting can be applied for center location and inliers for multiple instances of the same object. Scaling our model at scaling factor $\phi = 3$ will produce more accurate results. Furthermore, converting such systems to self-supervised learning will be a good idea, which is a set of scalable approaches and needs a lesser amount of data.

Acknowledgements This work was supported by the Science and Technology Planning Project of Guangdong Province, China, under the grant (2019A050520001).

Data availability The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declarations

Conflict of interest The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

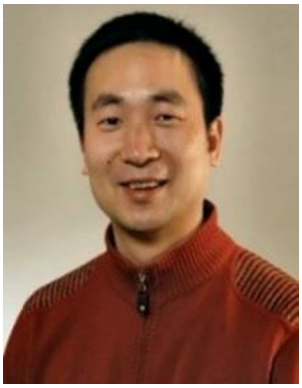
1. Tan, M., Pang, R., Le, Q. V.: EfficientDet : Scalable and Efficient Object Detection, in IEEE/CVF conference on computer vision and pattern recognition, pp. 10781–10790 (2020)
2. Lepetit V.: BB8 : a scalable, accurate, robust to partial occlusion method for predicting. Proc. IEEE Int. Conf. Comput. Vis., pp. 3828–3836, (2017)
3. Tekin, B., Sinha, S. N., & Fua, P.: Real-time seamless single shot 6d object pose prediction. In: Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 292-301). (2018). https://doi.org/10.1109/CVPR.2018.00038.
4. Xiang Y., Schmidt T., Narayanan V., and Fox D.: PoseCNN : a convolutional neural network for 6D object pose estimation in cluttered scenes. (2018)
5. Capellen C., Schwarz M., and Behnke S.: ConvPoseCNN: Dense convolutional 6D object pose estimation. VISIGRAPP 2020 - Proc. 15th Int. Jt. Conf. Comput. Vision, Imaging Comput. Graph. Theory Appl., 5, 162–172, (2020). https://doi.org/10.5220/0008990901620172.
6. Periyasamy, A. S., Capellen, C., Schwarz, M., & Behnke, S.: ConvPoseCNN2: Prediction and Refinement of Dense 6D Object Pose. In: International Joint Conference on Computer Vision, Imaging and Computer Graphics (pp. 353-371). Cham: Springer International Publishing. (2020). https://doi.org/10.1007/978-3-030-94893-1.
7. Peng, S., Liu, Y., Huang, Q., Zhou, X., Bao, H.: Pvnet: 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) https://doi.org/10.1109/CVPR.2019.00469.
8. Yu, X., Zhuang, Z., Koniusz, P., Li, H.: 6dof object pose estimation via differentiable proxy voting loss. (2020). arXiv preprint arXiv:2002.03923.
9. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning (pp. 6105-6114). PMLR (2019)
10. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981). https://doi.org/10.1145/358669.358692
11. Cao, Z., Sheikh, Y., Banerjee, N. K.: Real-time scalable 6DOF pose estimation for textureless objects. In: 2016 IEEE International conference on Robotics and Automation (ICRA) (pp. 2441-2448). IEEE. (2016). https://doi.org/10.1109/ICRA.2016.7487396.
12. Hinterstoisser, S., et al.: Gradient response maps for real-time detection of textureless objects. IEEE Trans. Pattern Anal. Mach. Intell. **34**(5), 876–888 (2012). https://doi.org/10.1109/TPAMI.2011.206

13. Hinterstoisser S., Lepetit V., Ilic S., and Holzer S.: Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In: 12th Int. Conf. Comput. Vis., vol. 3, pp. 548–562, 2012, [Online]. http://link.springer.com/https://doi.org/10.1007/978-3-642-33868-7.
14. Lowe, D.G.: Object recognition from local scale invariant features. IEEE Int Conf Comput Vision (ICCV) **2**(5), 1150–1157 (1999). https://doi.org/10.1016/0262-5075(81)90042-7
15. Pavlakos, G., Zhou, X., Chan, A., Derpanis, K.G., Daniilidis, K.: "6-DoF object pose from semantic keypoints", in. IEEE Int Conf Robot Autom (ICRA) **2017**, 2011–2018 (2017)
16. Rothganger, F., Lazebnik, S., Schmid, C., Ponce, J.: 3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. Int. J. Comput. Vis. **66**(3), 231–259 (2006). https://doi.org/10.1007/s11263-005-3674-1
17. Tulsiani S. and Malik J.: Viewpoints and Keypoints. Cvpr, pp. 1–8, (2015)
18. Krull, A., Brachmann, E., Michel, F., Yang, M. Y., Gumhold, S., Rother, C.: Learning analysis-by-synthesis for 6D pose estimation in RGB-D images. In: Proceedings of the IEEE international conference on computer vision (pp. 954-962). (2015). https://doi.org/10.1109/ICCV.2015.115.
19. Li, Z., Wang, G., & Ji, X.: 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 (pp. 7678-7687). (2019). https://doi.org/10.1109/ICCV.2019.00777.
20. Park, K., Patten, T., & Vincze, M.: Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 7668-7677). (2019). https://doi.org/10.1109/ICCV.2019.00776.
21. Zakharov, S., Shugurov, I., Ilic, S.: Dpod: 6d pose object detector and refiner. In: Proceedings of the IEEE/CVF international conference on computer vision (pp. 1941-1950). (2019). https://doi.org/10.1109/ICCV.2019.00203.
22. Girshick R.: Fast R-CNN (2015). https://doi.org/10.1109/ICCV.2015.169.
23. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN : towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1–14 (2017)
24. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., Berg, A. C.: Ssd: single shot multibox detector. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14 (pp. 21-37). Springer International Publishing. (2016).
25. Redmon, J., Divvala, S., Girshick, R., Farhadi, A. You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788). (2016).
26. Su, H., Qi, C. R., Li, Y.: Guibas, L. J. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In: Proceedings of the IEEE international conference on computer vision (pp. 2686-2694). (2015). https://doi.org/10.1109/ICCV.2015.308.
27. Kehl, W., Manhardt, F., Tombari, F., Ilic, S., & Navab, N. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In: Proceedings of the IEEE İnternational Conference On Computer Vision (pp. 1521-1529), (2017).
28. Bukschat Y. and Vetter M.: EfficientPose: An efficient, accurate and scalable end-to-end 6D multi object pose estimation approach [Online]. (2020). http://arxiv.org/abs/2011.04307.
29. Ullah, F., Wei, W., Daradkeh, Y. I., Javed, M., Rabbi, I., & Al Juaid, H.: A Robust convolutional neural network for 6D Object Pose Estimation from RGB Image with Distance Regularization Voting Loss. Scientific Programming, 2022. (2022).
30. Su, Y., Saleh, M., Fetzer, T., Rambach, J., Navab, N., Busam, B., 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).
31. Yu, Y., Niu, C., Li, J., Xu, K.: Multi-view 2D–3D alignment with hybrid bundle adjustment for visual metrology. Vis. Comput. (2021). https://doi.org/10.1007/s00371-021-02082-w
32. Zabulis, X., Lourakis, M.I.A., Koutlemanis, P.: Correspondence-free pose estimation for 3D objects from noisy. Vis. Comput. (2016). https://doi.org/10.1007/s00371-016-1326-9
33. Liu, X., Yin, J., Liu, H., Yin, Y.: PISEP 2: pseudo-image sequence evolution-based 3D pose prediction. Vis. Comput. (2021). https://doi.org/10.1007/s00371-021-02135-0
34. Cai, Y., Ge, L., Liu, J., Cai, J., Cham, T. J., Yuan, J., Thalmann, N. M.: Exploiting spatial-temporal relationships for 3d pose estimation via graph convolutional networks. In: Proceedings of the IEEE/CVF international conference on computer vision (pp. 2272-2281). (2019).
35. Song, C., Song, J., & Huang, Q.: Hybridpose: 6d object pose estimation under hybrid representations. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 431-440). (2020). https://doi.org/10.1109/CVPR42600.2020.00051.
36. Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2117-2125). (2017).
37. Zoph B., V Le Q.: Searching for activation functions. In: 6th Int. Conf. Learn. Represent. ICLR 2018-Work. Track Proc., pp. 1–13, (2018)
38. Chollet F.: Xception: deep learning with depthwise separable convolutions. In: CVPR, pp. 1251–1258 (2017).
39. Brachmann, E., Michel, F., Krull, A., Yang, M. Y., Gumhold, S.: Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In: Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3364-3372). (2016). https://doi.org/10.1109/CVPR.2016.366.
40. Billings G. and Johnson-roberson M.: SilhoNet: an RGB method for 3D object pose estimation and grasp planning.
41. Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., and Rother, C.: 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 (pp. 536-551). Springer International Publishing. (2014). https://doi.org/10.1007/978-3-319-10605-2_35.
42. Chen, T., Gu, D.: CSA6D-channel-spatial attention networks for 6D object pose estimation. Cognit. Comput. **14**, 702–713 (2022). https://doi.org/10.1007/s12559-021-09966-y
43. Iwase S., Liu X., Khirodkar R., Yokota R., and Kitani K. M.: RePOSE: Fast 6D object pose refinement via deep texture rendering. In: Proc. IEEE Int. Conf. Comput. Vis., pp. 3283–3292, (2021). https://doi.org/10.1109/ICCV48922.2021.00329.
44. Hua, W., Zhou, Z., Wu, J., Huang, H., Wang, Y., Xiong, R.: REDE: end-to-end object 6D pose robust estimation using differentiable outliers elimination. IEEE Robot. Autom. Lett. **6**(2), 2886–2893 (2021). https://doi.org/10.1109/LRA.2021.3062304
45. Feng H., Zhang L., Yang X., and Liu Z.: MixedFusion: 6D Object pose estimation from decoupled RGB-depth features. In: Proc. Int. Conf. Pattern Recognit., pp. 685–691, (2020). https://doi.org/10.1109/ICPR48806.2021.9412494.
46. He Y., Sun W., Huang H., Liu J., Fan H., and Sun J.: PVN3D: A deep point-wise 3D keypoints voting network for 6DoF pose estimation. In: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 11629–11638, (2020). https://doi.org/10.1109/CVPR42600.2020.01165.

47. Wu J., Liu L., Wang Y., and Xiong R.: Towards two-view 6D object pose estimation: a comparative study on fusion strategy. [Online] (2022). http://arxiv.org/abs/2207.00260.
48. Xu Z., Zhang Y., Chen K., and Jia K.: BiCo-net: regress globally, match locally for robust 6D pose estimation. no. c, [Online] (2022). http://arxiv.org/abs/2205.03536.
49. Arge FORL., and Mage CI.: Very deep convolutıonal networks for large-scale ımage recognıtıon. in *ICLR*, pp. 1–14 (2015)

Faheem Ullah received a BS degree in computer science from Gomal University, D.I.Khan, Pakistan, in 2006 and an MS in cognitive science and intelligent systems from the Universiti Malaysia Sarawak, Sarawak, Malaysia, in 2015. He is currently pursuing his Ph.D. in computer vision using deep learning at the School of Automation Science and Engineering, South China University of Technology. His current research interests include deep learning, computer vision, and 6D object pose estimation for robot manipulation. He has experience in databases, web, and mobile augmented reality.

Wu Wei received a Ph.D. from Huazhong University of Science and Technology in 2000. From January 2001 to December 2002, he was a postdoctoral researcher at the Department of Automation, Tsinghua University. From 2003 to 2004, he worked as a researcher at the Institute of Intelligent Systems, The Hong Kong Polytechnic University. He is also a member of the International IEEE Control Systems, Automotive Technology, Signal Processing, System People and Control Systems, Robotics, and Automation Association. He is currently a professor at the School of Automation Science and Engineering, South China University of Technology. His research focuses on robot control.



Zhun Fan received a Ph. D. degree in electrical engineering, from Michigan State University, USA, in 2004. From 2004 to 2007, he worked as an assistant professor at the Technical University of Denmark, Denmark. From 2007 to 2011, he was an associate professor at the Department of Mechanical Engineering and the Department of Management Engineering at the Technical University of Denmark. He is currently a professor at the College of Engineering, Shantou University. His research interest covers artificial intelligence, mechatronics design automation, intelligent robot system, computer vision, and evolutionary computation.

Qiuda Yu received his bachelor's degree from Hangzhou Dianzi University, Zhejiang, China. He is currently pursuing Ph.D. from the School of Automation Science and Engineering, South China University of Technology. His research focuses on visual grasp, visual guided control, and deep learning.