Robotics and Autonomous Systems, vol. 168, article 104490, Elsevier, 2023.

YOLOPose V2: Understanding and Improving Transformer-based 6D Pose Estimation

Arul Selvam Periyasamy, Arash Amini, Vladimir Tsaturyan, and Sven Behnke

Autonomous Intelligent Systems, University of Bonn, Germany
periyasa@ais.uni-bonn.de

Abstract. 6D object pose estimation is a crucial prerequisite for autonomous robot manipulation applications. The state-of-the-art models for pose estimation are convolutional neural network (CNN)-based. Lately, Transformers, an architecture originally proposed for natural language processing, is achieving state-of-the-art results in many computer vision tasks as well. Equipped with the multi-head self-attention mechanism, Transformers enable simple single-stage end-to-end architectures for learning object detection and 6D object pose estimation jointly. In this work, we propose YOLOPose (short form for You Only Look Once Pose estimation), a Transformer-based multi-object 6D pose estimation method based on keypoint regression and an improved variant of the YOLOPose model. In contrast to the standard heatmaps for predicting keypoints in an image, we directly regress the keypoints. Additionally, we employ a learnable orientation estimation module to predict the orientation from the keypoints. Along with a separate translation estimation module, our model is end-to-end differentiable. Our method is suitable for real-time applications and achieves results comparable to state-of-the-art methods. We analyze the role of object queries in our architecture and reveal that the object queries specialize in detecting objects in specific image regions. Furthermore, we quantify the accuracy trade-off of using datasets of smaller sizes to train our model.

Autonomous robotic object manipulation in real-world scenarios depends on high-quality 6D object pose estimation. Such object poses are also crucial in many other applications like augmented reality, autonomous navigation, and industrial bin picking. In recent years, with the advent of convolutional neural networks (CNNs), significant progress has been made to boost the performance of object pose estimation methods. Due to the complex nature of the task, the standard methods favor multi-stage approaches, i.e., feature extraction followed by object detection and/or instance segmentation, target object crop extraction, and, finally, 6D object pose estimation. In contrast, Carion et al. [8] introduced DETR, a Transformer-based single-stage architecture for object detection. In our previous work [1], we extended the DETR model with the T6D-Direct architecture to perform multi-object 6D pose direct regression. Taking advantage of the *pleasingly parallel* nature of the Transformer architecture, the T6D-Direct model predicts 6D pose for all the objects in an image in one forward-pass. Despite the advantages of the architecture and its impressive performance, the

overall 6D pose estimation accuracy of T6D-Direct, which directly regresses translation and orientation components of the 6D object poses, is inferior to state-of-the-art CNN-based methods, especially in rotation estimation. Instead of directly regressing the translation and orientation components, the keypoint-based methods predict the 2D pixel projection of 3D keypoints and use the perspective-n-point (PnP) algorithm to recover the 6D pose. In this work, we extend our T6D-Direct approach to utilize keypoints as 2D projected sparse correspondences. Our proposed model performs keypoint direct regression instead of the standard heatmaps for predicting the spatial position of the keypoints in a given RGB image and uses a multi-layer perceptron (MLP) to learn the orientation component of 6D object pose from the keypoints. Another independent MLP serves as the translation direct estimator. In short, our contributions include:

1. a Transformer-based real-time single-stage model for multi-object monocular 6D pose estimation using keypoint regression,
2. a learnable rotation estimation module to estimate object orientation from a set of keypoints to develop an end-to-end differentiable architecture for pose estimation, and
3. achieving results comparable to the state-of-the-art pose estimators on the YCB-Video dataset as well as yielding the fastest inference time.

This article extends our conference paper [2] that received the Best Paper Award at the 17th International Conference on Intelligent Autonomous Systems, 2021. we make the following additional contributions:

1. analyzing the role of object queries in the YOLOPose architecture,
2. improving the accuracy of the YOLOPose model by deriving new variants with additional inputs to the pose estimation MLPs,
3. quantifying the robustness of the learned PnP module compared to the analytical PnP algorithm, and
4. quantifying the accuracy trade-off of using datasets of smaller sizes to train our model.

1 Related Work

1.1 RGB Object Pose Estimation

The recent significant progress in the task of 6D object pose estimation from RGB images is driven—like for many computer vision tasks—by deep learning methods. The current methods for object pose estimation from RGB images can be broadly classified into three major categories, namely direct regression methods, keypoint-based methods, and refinement-based methods. Direct regression methods formulate the task of pose estimation as a regression of continuous translation and rotation components, whereas keypoint-based methods predict the location of projection of some of the specific keypoints or the 3D coordinates

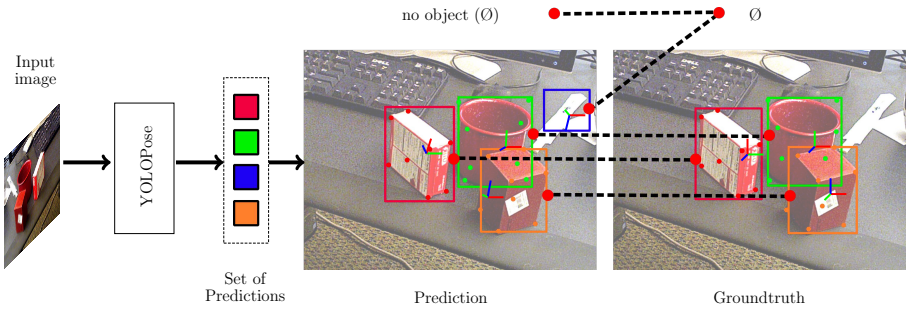


Fig. 1. Proposed YOLOPose approach. Our model predicts a set with a fixed cardinality. Each element in the set corresponds to an object prediction and after predicting all the objects in the given input image, the rest of the elements are padded with Ø as no object predictions. The predicted and the ground-truth sets are matched using bipartite matching and the model is trained to minimize the Hungarian loss between the matched pairs. Our model is end-to-end differentiable.

of the visible pixels of an object in an image and use the PnP algorithm to retrieve the 6D pose from the estimated 2D-3D correspondences. Often, the PnP algorithm is used in conjunction with RANSAC for improving the robustness of the pose estimation.

Some examples of direct regression methods include [1, 41, 57, 59]. Keypoint-based include [22, 23, 40, 44, 53]. One important detail to note regarding these methods is that except for [1, 7, 23, 54] all the other methods use multi-staged CNNs. The first stage performs object detection and/or semantic or instance segmentation to detect the objects in the given RGB image. Using the object detections from the first stage, a crop containing the target object is extracted. In the second stage, these models predict the 6D pose of the target object. To enable end-to-end differentiability of the CNN models, these models employ region of interest (ROI) pooling, anchor box proposal, or non-maximum suppression (NMS) procedures [21, 45, 46]. In terms of the 6D pose prediction accuracy, keypoint-based methods perform considerably better than the direct regression methods [19], though this performance gap is shrinking [1].

The third category of pose estimation methods are the refinement-based methods. These methods formulate the task of pose estimation as iterative pose refinement, i.e., the target object is rendered according to the current pose estimate, and a model is trained to estimate a pose update that minimizes the pose error between the ground-truth and the current pose prediction. Refinement-based methods [26, 30, 37, 42] achieve the highest pose prediction accuracy among three categories [19]. They need, however, a good object pose initialization within the basin of attraction of the final pose estimate.

1.2 RGB-D Object Pose Estimation

Although we deal with the problem of RGB pose estimation in this work, it is highly relevant to review the RGB-D methods as well. RGB-D deep learning methods for pose estimation fuse visual features from the RGB input extracted by a CNN model and geometric features from the point cloud or depth input. The predominant methods for extracting point-wise geometric features from the point cloud input include PointNet [43], PointNet++ [43], and Point Transformer [61]. Xu et al. [60] learned to estimate 3D bounding box corners by fusing visual and geometric features. Wang et al. [56] learned dense point-wise embeddings from which the pose parameters are regressed in an iterative pose refinement procedure. He et al. [16] lifted the pixel-wise 2D keypoint offset learning proposed by Peng et al. [40] for RGB images to 3D point clouds by learning point-wise 3D keypoint offset and using a deep Hough voting network. He et al. [15] jointly learned keypoint detection and instance segmentation and estimated 6D pose from the predicted keypoint and segmentation using a least-squares fitting scheme from multi-view RGB-D input. Overall, RGB-D methods leverage the geometric features in the point cloud or depth input and achieve better accuracy than RGB-only methods. Despite the advantages of the RGB-D data, the RGB-D sensors have limitations in terms of resolution and frame rate. Reflectance and transparency properties of the objects also pose challenges for RGB-D sensors [25, 32, 35, 36]. Additionally, calibrating RGB and depth sensors in large industrial settings is often time-consuming [3, 49, 50]. Moreover, RGB methods are comparatively simpler and require less computational power and processing time. This motivates us in focusing on monocular RGB pose estimation.

1.3 Learned PnP

Given a set of 3D keypoints and their corresponding 2D projections, and the camera intrinsics, the PnP algorithm is used to recover the 6D object pose. The standard PnP algorithm [13] and its variant EPnP [28] are used in combination with RANSAC to improve the robustness against outliers. Both PnP and RANSAC are not trivially differentiable. In order to realize an end-to-end differentiable pipeline for the 6D object pose estimation, Wang et al. [57], and Hu et al. [22] proposed a learning-based PnP module. Similarly, Li et al. [29] introduced a learnable 3D Lifter module to estimate vehicle orientation. Recently, Chen et al. [9] proposed to differentiate PnP using the implicit function theorem. Although a generic differentiable PnP has many potentials, due to the overhead incurred during training, we opt for a simple MLP that estimates the orientation component given the 2D keypoints.

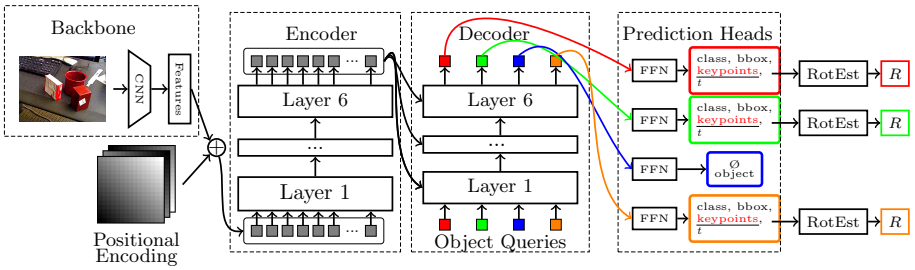


Fig. 2. YOLOPose architecture in detail. Given an RGB input image, we extract features using the standard ResNet model. The extracted features are supplemented with positional encoding and provided as input to the Transformer encoder. The encoder module consists of six standard encoder layers with skip connections. The output of the encoder module is provided to the decoder module along with N object queries. The decoder module also consists of six standard decoder layers with skip connections generating N output embeddings. The output embeddings are processed with FFNs to generate a set of N elements in parallel. Each element in the set is a tuple consisting of the bounding box, the class probability, the translation, and the interpolated bounding box keypoints. A learnable rotation estimation module is employed to estimate object orientation R from the predicted 2D keypoints.

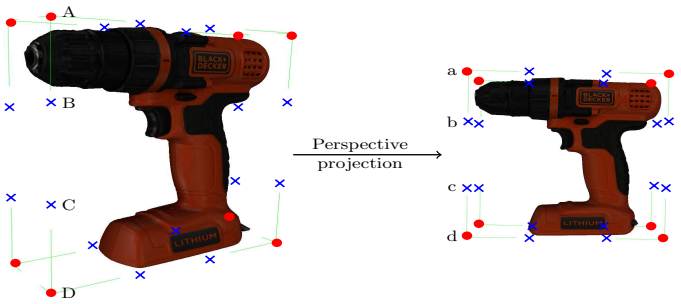


Fig. 3. Interpolated bounding box points. Bounding box points are indicated with red dots, and the interpolated points are indicated with blue crosses. The cross-ratio of every four collinear points is preserved during perspective projection, e.g., the cross-ratio of points A, B, C, and D remains the same in 3D and, after perspective projection, in 2D.

2 Method

2.1 Multi-Object Keypoint Regression as Set Prediction

Object pose estimation is the task of estimating the position and the orientation of an object with respect to the sensor coordinate frame. Occlusion, reflective properties of objects, lighting effects, and camera noise in real-world settings aggravate the complexity of the task. The early methods for pose estimation like template matching [6, 17, 20] and keypoint-based [39, 48, 55] decoupled object pose estimation from object detection and followed a multi-staged pipeline in which 2D bounding boxes are extracted in the first stage and only the crop containing the target object is processed in the second stage for pose estimation. Most of the deep learning methods also followed the same multi-stage approach for pose estimation. However, multi-stage pipelines suffer from two major issues. Firstly, inaccuracies in the first stage impede the final pose estimation accuracy, Secondly, complex modules like NMS, ROI, and anchor boxes are needed to realize end-to-end differentiable pipelines. Multi-object pose estimation methods alleviate the issues with multi-stage pipelines by detecting and localizing all objects in a given image. Following DETR [8] and T6D-Direct [1], we formulate multi-object pose estimation as a set prediction problem. Fig. 1 gives an overview of our approach. Given an RGB input image, our model outputs a set of elements with a fixed cardinality N. Each element in the set is a tuple containing the 2D bounding boxes, the class probability, the translation, and the keypoints. 2D bounding boxes are represented with the center coordinates, height, and width proportional to the image size. The class probability is predicted using a softmax function. To estimate translation $\mathbf{t} = [t_x, t_y, t_z]^T \in \mathbb{R}^3$ as the coordinate of the object origin in the camera coordinate system, we follow the method proposed by PoseCNN [59] which decouples the estimation of $\mathbf{t}$ into directly regressing the object's distance from the camera t_z and the 2D location of projected 3D object's centroid in the image plane $[c_x, c_y]^T$. Finally, having the intrinsic camera matrix, we can recover t_x and t_y. The exact choice of the keypoints is discussed in Section 2.3. The number of objects present in an image varies; therefore, to enable output sets with fixed cardinality, we choose N to be larger than the expected maximum number of objects in an image in the dataset and introduce a no-object class Ø. This Ø class is analogous to the background class used in semantic segmentation models. In addition to predicting the corresponding classes for objects present in the image, our model is trained to predict Ø for the rest of the elements in the set.

2.2 Model Architecture

The proposed YOLOPose architecture is shown in Fig. 2. The model consists of a ResNet backbone followed by Transformer-based encoder-decoder module and MLP prediction heads to predict a set of tuples described in Section 2.1. CNN architectures have several inductive biases designed into them [10, 27]. These strong biases enable CNNs to learn efficient local spatial features in a fixed

neighborhood defined by the receptive field to perform well on many computer vision tasks. In contrast, Transformers, aided by the attention mechanism, are suitable for learning spatial features over the entire image. This makes the Transformer architecture suitable for multi-object pose estimation. In this section, we describe the individual components of the YOLOPose architecture.

Backbone Network We use a ResNet50 backbone for extracting features from the given RGB image. For an image size of height H and width W, the backbone network extracts 2048 low-resolution feature maps of size H/32×W/32. We then use 1×1 convolution to reduce the 2048 feature dimensions to smaller d=256 dimensions. The standard Transformer models are designed to process vectors. Hence, to enable processing the d×H/32×W/32 features, we vectorize them to $d \times \frac{H}{32} \frac{W}{32}$.

Encoder The Transformer encoder module consists of six encoder layers with skip connections. Each layer performs multi-head self-attention of the input vectors. Given pixel with embedding x of dimension d, the embedding is split into h chunks, or "heads" and for each head i, the scaled dot-product attention is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^\top}{\sqrt{d/h}})V,$$

where Q, K, and V are the query, key, and value matrices for the head i, respectively, and are computed by linearly projecting x using projection parameter matrices W^q, W^k, W^v, respectively. The attention outputs of the heads are concatenated and transformed linearly to compute MultiHead self-attention:

$$\text{MultiHead(Q,K,V)} = \underset{i \in h}{\text{concat}}(\text{Attention}(Q_i, K_i, V_i))W^O,$$

where $W^O \in \mathbb{R}^{d \times d}$ is also a projection parameter matrix, and concat denotes concatenation along the embedding dimension. In contrast to the convolution operation, which limits the receptive field to a small neighborhood, self-attention enables a receptive field of the size of the whole image. Note that the convolution operation can be cast as a special case of self-attention Cordonnier et al. [11].

Positional Encodings The multi-head self-attention operation is permutation-invariant. Thus, the Transformer architecture ignores the order of the input vectors. We employ the standard solution of supplementing the input vectors with absolute positional encoding following Carion et al. [8] to provide the Transformer model with spatial information of the pixels. We encode the pixel coordinates as sine and cosine functions of different frequencies:

$$\text{P.E.}_{(pos,p)} = sin(pos/10000^{\frac{2p}{d}}),$$

$$\text{P.E.}_{(pos,p+1)} = cos(pos/10000^{\frac{2p+1}{d}}),$$

where *pos* is the pixel coordinate (either width or height), d is the embedding dimension, and p is the index of the positional encoding. The positional embeddings are added to the backbone feature vectors before feeding them to the Transformer encoder as input.

Decoder On the decoder side, we compute cross-attention between the encoder output embeddings and N learnable embeddings, referred to as *object queries*, to generate decoder output embeddings, where N is the cardinality of the predicted set. The decoder consists of six decoder layers and the object queries are provided as input to each decoder layer. Unlike the fixed positional encoding used in the encoder, the object queries are learned jointly with the original learning objective—joint object detection and pose estimation, in our case—from the dataset. At the start of the training process, the object queries are initialized randomly, and during inference, the object queries are fixed. In Section 5, we investigate the role of object queries generating object predictions. The embeddings used in our model—both learned and fixed—are 256-dimensional vectors.

FFN From the N decoder output embeddings, we use feed-forward prediction heads to generate a set of N output tuples independently. Each tuple consists of the class probability, the bounding box, the keypoints, and the pose parameters. Prediction heads are fully-connected three-layer MLPs with hidden dimension 256 and ReLU activation in each layer.

2.3 Keypoints Representation

An obvious choice for selecting 3D keypoints is the eight corners of the 3D bounding box [38]. Peng et al. [40] instead used the Farthest Point Sampling (FPS) algorithm to sample eight keypoints on the surface of the object meshes, which are also spread out on the object to help the PnP algorithm find a more stable solution. Li et al. [29] defined the 3D representation of an object as sparse interpolated bounding boxes (IBBs), shown in Fig. 3, and exploited the property of perspective projection that cross-ratio of every four collinear points in 3D (A, B, C, and D as illustrated in Fig. 3) is preserved under perspective projection in 2D [14]. The cross-ratio consistency is enforced by an additional component in the loss function that the model learns to minimize during training. We further investigate these keypoints representations in Section 4 and present our results in Table 4.

2.4 RotEst

The standard solution for the perspective geometry problem of recovering 6D object/camera pose given 2D-3D correspondences and a calibrated camera is the PnP algorithm. The minimum number of correspondences needed for employing PnP is 4. However, the accuracy and the robustness of the estimated pose increase with the number of correspondences. Moreover, PnP is used in conjecture

with RANSAC to increase the robustness. Although PnP is a standard and well-understood solution, incorporating it in neural network pipelines introduces two drawbacks. First, it is not trivially differentiable. Second, PnP combined with RANSAC needs multiple iterations to generate highly accurate pose predictions. These drawbacks hinder us in realizing end-to-end differentiable pipelines with a single step forward pass for pose estimation. To this end, we introduce the RotEst module. For each object, from the estimated pixel coordinates onto which the 32 keypoints (the eight corners of the 3D bounding box and the 24 intermediate bounding box keypoints) are projected, the RotEst module predicts the object orientation represented as the 6D continuous representation in SO(3) [62]. Furthermore, we experimented with providing additional inputs to the FFNs. We created three variants of the YOLOPose model: variants A, B, and C (shown in Fig. 4). In variant A, in addition to the estimated IBB keypoints, we provide the output embedding of the object query to the FFNs. In variant B, IBB keypoints, object query output embedding, and the canonical 3D bounding box points (based on the predicted object class) are provided to the FFNs, whereas in variant C, estimated IBB keypoints and class probabilities are fed as input to FFNs. Note that the size of the embedding used in YOLOPose model is 256. Thus, the number of parameters used in FFNs of the three variants is larger than that of the YOLOPose model. We implement the RotEst module using six fully connected layers with a hidden dimension 1024 and a dropout probability of 0.5.

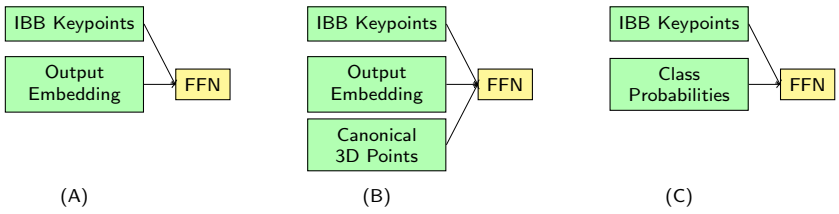


Fig. 4. Variants of the YOLOPose model. All three variants are derived from the YOLOPose model and differ in the inputs provided to the pose estimation FFNs.

2.5 Loss Function

Our model is trained to minimize the Hungarian loss between the predicted and the ground-truth sets. Computing the Hungarian loss involves finding the matching pairs in the two sets. We use bipartite matching [8, 24, 51] to find the permutation of the predicted elements that minimize the matching cost. Given the $\emptyset$ class padded ground-truth set $\mathcal{Y}$ of cardinality N containing labels $y_1, y_2, ..., y_N$, the predicted set denoted by $\hat{\mathcal{Y}}$, we search for the optimal permutation $\hat{\sigma}$ among the possible permutations $\sigma \in \mathfrak{S}_N$ that minimizes the matching

cost $\mathcal{L}_{match}$. Formally,

$$\hat{\sigma} = \underset{\sigma \in \mathfrak{S}_N}{\arg\min} \sum_{i}^{N} \mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)}). \tag{1}$$

Although each element of the set is a tuple containing four components, bounding box, class probability, translation, and keypoints, we use only the bounding box and the class probability components to define the matching cost function. In practice, omitting the other components in the cost function definition does not hinder the model's ability in learning to predict the keypoints and keeps the computational cost of the matching process minimal.

Given the matching ground-truth and predicted sets $\mathcal{Y}$ and $\hat{\mathcal{Y}}_\sigma$, respectively, the Hungarian loss is computed as:

$$\begin{aligned}\mathcal{L}_{Hungarian}(\mathcal{Y}, \hat{\mathcal{Y}}_\sigma) = \sum_{i}^{N} [-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{c_i \neq \emptyset} \mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}(i)}) + \\ \lambda_{kp} \mathbb{1}_{c_i \neq \emptyset} \mathcal{L}_{kp}(k_i, \hat{k}_{\hat{\sigma}(i)}) + \lambda_{pose} \mathbb{1}_{c_i \neq \emptyset} \mathcal{L}_{pose}(R_i, t_i, \hat{R}_{\hat{\sigma}(i)}, \hat{t}_{\hat{\sigma}(i)})].\end{aligned} \tag{2}$$

Class Probability Loss The class probability loss function is the standard negative log-likelihood. Since we choose the cardinality of the set to be higher than the expected maximum number of objects in an image, the $\emptyset$ class appears disproportionately often. Thus, we weigh the loss for the $\emptyset$ class with a factor of 0.1.

Bounding Box Loss The 2D bounding boxes are represented as (c_x, c_y, w, h) where (c_x, c_y) are 2D pixel coordinates and w and h are object width and height, respectively. To train the bounding box prediction head, We use a weighted combination of the Generalized IoU (GIoU) [47] and ℓ_1-loss with 2 and 10 factors, respectively.

$$\mathcal{L}_{box}(b_i, \hat{b}_{\sigma(i)}) = \alpha \mathcal{L}_{iou}(b_i, \hat{b}_{\sigma(i)}) + \beta ||b_i - \hat{b}_{\sigma(i)}||, \tag{3}$$

$$\mathcal{L}_{iou}(b_i, \hat{b}_{\sigma(i)}) = 1 - \left(\frac{|b_i \cap \hat{b}_{\sigma(i)}|}{|b_i \cup \hat{b}_{\sigma(i)}|} - \frac{|B(b_i, \hat{b}_{\sigma(i)}) \setminus b_i \cup \hat{b}_{\sigma(i)}|}{|B(b_i, \hat{b}_{\sigma(i)})|} \right), \tag{4}$$

and $B(b_i, \hat{b}_{\sigma(i)})$ is the largest box containing both the ground truth b_i and the prediction $\hat{b}_{\sigma(i)}$.

Keypoint Loss Having the ground truth K_i and the model output $\hat{K}_{\hat{\sigma}(i)}$, the keypoints loss can be represented as:

$$\mathcal{L}_{kp}(K_i, \hat{K}_{\hat{\sigma}(i)}) = \gamma ||K_i - \hat{K}_{\hat{\sigma}(i)}||_1 + \delta \mathcal{L}_{\mathcal{CR}}, \tag{5}$$

where γ and δ are hyperparameters. The first part of the keypoints loss is the ℓ_1 loss, and for the second part, we employ the cross-ratio loss $\mathcal{L}_{\mathcal{CR}}$ defined in Equation 6 to enforce the cross-ratio consistency in the keypoint loss as proposed by Li et al. [29]. This loss is self-supervised by preserving the cross-ratio of each line to be 4/3. The reason is that after the camera projection of the 3D bounding box on the image plane, the cross-ratio of every four collinear points remains the same.

$$\mathcal{L}_{\mathcal{CR}} = Smooth\ell_1(\mathrm{CR}^2 - \frac{||c-a||^2||d-b||^2}{||c-b||^2||d-a||^2}),\ \mathrm{CR} = \frac{||C-A||\ ||D-B||}{||C-B||\ ||D-A||} = \frac{4}{3}, \tag{6}$$

where CR^2 is chosen since $||.||^2$ can be easily computed using vector inner product. A, B, C, and D are four collinear points and their corresponding predicted 2D projections are a, b, c, and d, respectively.

Pose Loss We supervise the rotation R and the translation $\mathbf{t}$ individually via employing PLoss and SLoss from [59] for rotation, and ℓ_1 loss for translation.

$$\mathcal{L}_{pose}(R_i, \mathbf{t}_i, \hat{R}_{\sigma(i)}, \hat{\mathbf{t}}_{\sigma(i)}) = \mathcal{L}_{rot}(R_i, \hat{R}_{\sigma(i)}) + ||\mathbf{t}_i - \hat{\mathbf{t}}_{\sigma(i)}||_1, \tag{7}$$

$$\mathcal{L}_{rot} = \begin{cases} \frac{1}{|\mathcal{M}_i|} \sum\limits_{\mathrm{x}_1 \in \mathcal{M}_i} \min\limits_{\mathrm{x}_2 \in \mathcal{M}_i} ||R_i\mathrm{x}_1 - \hat{R}_{\sigma(i)}\mathrm{x}_2||_1 \text{ if symmetric,} \\ \frac{1}{|\mathcal{M}_i|} \sum\limits_{\mathrm{x} \in \mathcal{M}_i} ||R_i\mathrm{x} - \hat{R}_{\sigma(i)}\mathrm{x}||_1 \text{ otherwise,} \end{cases} \tag{8}$$

where $\mathcal{M}_i$ indicates the set of 3D model points. Here, we subsample 1.5K points from meshes provided with the dataset. R_i is the ground truth rotation, and $\mathbf{t}_i$ is the ground truth translation. $\hat{R}_{\sigma(i)}$ and $\hat{\mathbf{t}}_{\sigma(i)}$ are the predicted rotation and translation, respectively.

3 Evaluation

In this section, we evaluate the performance of our proposed YOLOPose model and its variants and compare it with other state-of-the-art 6D pose estimation methods.

3.1 Dataset

We use the challenging YCB-Video (YCB-V) [59] dataset to evaluate the performance of our model. YCB-V provides bounding box, segmentation, and 6D pose annotations for 133,936 RGB-D images. Since our model is RGB-based, we do not use the provided depth information. The dataset is generated by capturing video sequences of a random subset of objects from a total of 21 objects placed in tabletop configuration. From the 92 video sequences, twelve are used for testing

Fig. 5. Qualitative results on YCB-V test set. Top row: The predicted IBB keypoints overlaid on the input images. Bottom row: Ground truth and predicted object poses are visualized as object contours in green and blue colors, respectively.

and 80 are used for training. The objects used exhibit varying geometric shapes, reflectance properties, and symmetry. Thus, YCB-V is a challenging dataset for benchmarking 6D object pose estimation methods. YCB-V also provides high-quality meshes for all 21 objects. Mesh points from these objects are used in computing the evaluation metrics that we describe in Section 3.2. Hodaň et al. [19] provided a variant of YCB-V[1] as part of the BOP challenge in which the centers of the 3D bounding boxes are aligned with the origin of the model coordinate system and the ground-truth annotations are converted correspondingly. We use the BOP variant of the YCB-V dataset. In addition to the YCB-V dataset images, we use the synthetic dataset provided by PoseCNN [59] for training our model. Moreover, we initialize our model using the pre-trained weights on the COCO dataset [33] for the task of object detection.

3.2 Metrics

Xiang et al. [59] proposed area under the curve (AUC) of ADD and ADD-S metrics for evaluating the accuracy of non-symmetric and symmetric objects, respectively. Given the ground-truth 6D pose annotation with rotation and translation components R and $\mathbf{t}$, and the predicted rotation and translation components $\hat{R}$ and $\hat{\mathbf{t}}$, ADD metric is the average ℓ_2 distance between the subsampled mesh points $\mathcal{M}$ in the ground truth and the predicted pose. In contrast, the symmetry-aware ADD-S metric is the average distance between the closest subsampled mesh points $\mathcal{M}$ in the ground-truth and predicted pose. Following the standard procedure proposed by Xiang et al. [59], we aggregate the results and report the area under the threshold-accuracy curve for distance thresholds from zero to 0.1m.

$$\text{ADD} = \frac{1}{|\mathcal{M}|} \sum_{x \in \mathcal{M}} \|(Rx + \mathbf{t}) - (\hat{R}x + \hat{\mathbf{t}})\|, \tag{9}$$

$$\text{ADD-S} = \frac{1}{|\mathcal{M}|} \sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} \|(Rx_1 + \mathbf{t}) - (\hat{R}x_2 + \hat{\mathbf{t}})\|. \tag{10}$$

[1] `https://bop.felk.cvut.cz/datasets/`

The ADD and ADD-S metrics are combined into one metric by using ADD for non-symmetric objects and ADD-S for symmetric objects. This combined metric is denoted as ADD-(S).

3.3 Hyperparameters

The γ and δ hyperparameters in $\mathcal{L}_{kp}$ (Eq. (5)) are set to 1 and 10, respectively. While computing the Hungarian loss, the pose loss component is weighted down by a factor of 0.05. The cardinality of the predicted set N=20. The model takes images of the size 640 $\times$ 480 as input and is trained using the AdamW optimizer [34] with an initial learning rate of 10^{-4} for 150 epochs. Afterward, the model is trained additionally for 50 epochs, with a reduced learning rate by a factor of 0.1. The batch size is 32. Gradient clipping with a maximal gradient norm of 0.1 is applied.

3.4 Results

Table 1. Comparison of the proposed keypoints-based method YOLOPose with the state-of-the-art methods on YCB-V. The symmetric objects are denoted by *. The best results are shown in bold.

Method	GDR-Net [57]		YOLOPose (Ours)		YOLOPose-A (Ours)		DeepIM [30]		YOLOPose-A (Ours)	
Metric	AUC of ADD-S	AUC of ADD(-S)	AUC of ADD-S	AUC of ADD(-S)	AUC of ADD-S	AUC of ADD(-S)	AUC of ADD-S	AUC of ADD(-S)	AUC of ADD-S @0.1d	AUC of ADD(-S) @0.1d
master_chef_can	**96.6**	71.1	91.3	64.0	91.7	**71.3**	93.1	71.2	71.3	36.6
cracker_box	84.9	63.5	86.8	77.9	**92.0**	83.3	91.0	**83.6**	83.3	71.1
sugar_box	**98.3**	93.2	92.6	87.3	91.5	83.6	96.2	**94.1**	83.6	59.5
tomato_soup_can	**96.1**	**88.9**	90.5	77.8	87.8	72.9	92.4	86.1	72.9	29.8
mustard_bottle	**99.5**	**93.8**	93.6	87.9	96.7	93.4	95.1	91.5	93.4	93.4
tuna_fish_can	95.1	85.1	94.3	74.4	94.9	70.5	**96.1**	**87.7**	70.5	17.4
pudding_box	**94.8**	86.5	92.3	87.9	92.6	**87.0**	90.7	82.7	87.0	70.9
gelatin_box	**95.3**	88.5	90.1	83.4	92.2	85.7	94.3	**91.9**	85.7	23.4
potted_meat_can	82.9	72.9	85.8	**76.7**	85.0	71.4	**86.4**	76.2	71.4	31.2
banana	**96.0**	85.2	95.0	88.2	95.8	**90.0**	91.3	81.2	90.0	83.1
pitcher_base	**98.8**	**94.3**	93.6	88.5	95.2	90.8	94.6	90.1	90.8	90.1
bleach_cleanser	**94.4**	80.5	85.3	73.0	83.1	70.8	90.3	**81.2**	70.8	62.9
bowl*	84.0	84.0	92.3	92.3	**93.4**	**93.4**	81.4	81.4	93.4	87.4
mug	**96.9**	87.6	84.9	69.6	95.5	**90.0**	91.3	81.4	90.0	71.0
power_drill	91.9	78.7	**92.6**	**86.1**	92.5	85.2	92.3	85.5	85.2	73.6
wood_block*	77.3	77.3	84.3	84.3	**93.0**	**93.0**	81.9	81.9	93.0	93.0
scissors	68.4	43.7	**93.3**	**87.0**	80.9	71.2	75.4	60.9	71.2	42.5
large_marker	**87.4**	76.2	84.9	76.6	85.2	**77.0**	86.2	75.6	77.0	14.7
large_clamp*	69.3	69.3	92.0	92.0	**94.7**	**94.7**	74.3	74.3	94.7	94.1
extra_large_clamp*	73.6	73.6	**88.9**	**88.9**	80.7	80.7	73.3	73.3	80.7	65.7
foam_brick*	90.4	90.4	90.7	90.7	**93.8**	**93.8**	81.9	81.9	93.8	78.9
MEAN	89.1	80.2	90.1	82.6	**91.2**	**83.3**	88.1	81.9	83.3	61.4

Table 2. Pose estimation results on YCB-V.

Input	Method	ADD(-S)	AUC of ADD-S	AUC of ADD(-S)	Inference Time [ms/frame]
RGB	CosyPose† [26]	-	89.8	**84.5**	395
	PoseCNN [59]	21.3	75.9	61.3	-
	GDR-Net [57]	49.1	89.1	80.2	65
	YOLOPose (Ours)	65.0	90.1	82.6	**17**
	YOLOPose-A (Ours)	**69.0**	**91.2**	83.3	22
RGB-D	PVNet3D [16]	-	95.5	91.8	170
	PVNet3D+ICP [16]	-	96.1	92.3	190
	FFB6D [59]	-	96.6	92.7	**75**
	FFB6D+ICP [59]	-	**97.0**	**93.7**	95

† indicates the refinement-based method.

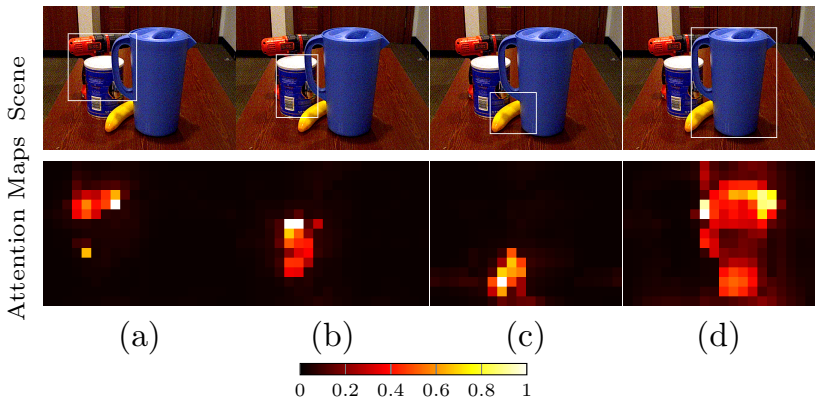


Fig. 6. Top: Object detections predicted by bounding boxes in the given image. Bottom: Decoder cross-attention maps for the object queries corresponding to the predictions in the first row.

In this section, we present the quantitative and qualitative results of the proposed method. We present exemplar qualitative results in Fig. 5. In Table 1, we provide the quantitative per class area under the accuracy curve (AUC) of the ADD-S and ADD(-S) metrics. Both YOLOPose and YOLOpose-A perform well across all object categories and achieve higher AUC scores than the methods in comparison. YOLOPose-A achieves an impressive AUC of ADD-S and ADD-(S) score of 91.2 and 83.3, respectively, which is an improvement of 1.1 and 0.7 over the YOLOPose model. In terms of the individual objects, YOLOPose-A performs significantly better than the mean on *mustard bottle*, *bowl*, *large clamp*, and *foam brick*, while performing worse than the mean on *master chef can*, *tuna fish can*, *bleach cleanser*, and *scissors*. Interestingly, our methods perform well on identical *large clamp* and *extra large clamp*, whereas both the competing methods perform poorly on these objects. Real-world robotic applications require handling objects of different sizes and this necessitates highly accurate pose estimates. The standard procedure of reporting the AUC of ADD-(S) and ADD-S metrics with a fixed threshold of 0.1m does not take the object size into account. To better reflect the performance of our method on smaller objects, we present the AUC of ADD-(S) and ADD-S metric with a threshold of 10 % of the object diameter. We denote this metric as AUC of ADD-(S) and ADD-S @0.1d. The accuracy of the proposed method drops significantly for smaller objects while using the object-specific threshold. In particular, the AUC of ADD-(S)@0.1d score for *tuna fish can*, *gelatin box*, and *large marker* are less than 30. This could be due to the fact that the Pose Loss discussed in Section 2.5 is computed using the subsampled model points and smaller objects contribute less to the overall loss. In Table 2, we also present a comparison of the ADD-S and the mean AUC ADD-S and ADD-(S) scores of the predominant RGB as well as RGB-D methods. Benefiting from the geometric features imparted by the depth information, RGB-D methods outperform RGB-only methods. However, RGB-only methods are catching up with the RGB-D methods fast [52].

The FFNs in our model generate the set predictions from the decoder output embeddings, which are the result of cross-attention between the object queries and the encoder output embeddings. Each encoder output embedding corresponds to a specific image pixel. This allows us to investigate the pixels that contribute the most to each object prediction. In Fig. 6, we visualize the decoder cross-attention corresponding to four different object detections, where the attended regions correspond to the object's spatial position in the image very well. Moreover, looking closely at the pixels with the highest attention score reveals the object parts that contribute most to the object predictions. For example, in Fig. 6(a), the tip and base of the *drill* contribute the most and in Fig. 6(d), the spout and the handle *pitcher base* contribute the most. Note that in Fig. 6(a), the base is severely occluded and the base barely visible. Despite being occluded, the attention mechanism focuses on the base heavily, which demonstrates the significance of the base in *drill* pose estimation.

In Table 3, we present a quantitative comparison of the YOLOPose variant discussed in Section 2.4. Variant A performs the best among the variants. This

can be attributed to the additional object-specific information contained in the output embedding.

Table 3. Quantative comparison of the YOLOPose variants.

Method	AUC of ADD(-S)	AUC of ADD-S	Parameters $\times 10^6$
YOLOPose	82.6	90.1	**48.6**
Variant A	**83.3**	**91.2**	53.2
Variant B	82.8	91.0	53.4
Variant C	82.8	90.9	52.8

3.5 Inference Time Analysis

In terms of inference speed, one of the major advantages of our architecture is that the feed-forward prediction networks (FFN) can be executed in parallel for each object. Thus, irrespective of the number of objects in an image, our model generates pose predictions in parallel. In Table 2, we present the inference time results for 6D pose estimation. YOLOPose-A operates at 45 fps, whereas YOLOpose operates at 59 fps, which is significantly better than the refinement-based methods and the RGB-D methods.

4 Ablation Study

In contrast to the standard approach of predicting the 2D keypoints and using a PnP solver—which is not trivially differentiable—to estimate the 6D object pose, we use the learnable RotEst module to estimate the object orientation from a set of predicted interpolated keypoints. In this section, we analyze the effectiveness of our RotEst module and the choice of the keypoint representation.

4.1 Effectiveness of keypoints representations

We compare various keypoints representations, namely 3D bounding box keypoints (BB), random keypoints sampled using the FPS algorithm, and our representation of choice, the interpolated bounding box keypoints (IBB). We use the OpenCV implementation of the RANSAC-based EPnP algorithm with the same parameters to recover the 6D object pose from the predicted keypoints. Since EPnP does not contain any learnable components, this experiment serves the goal of evaluating the ability of the YOLOPose model to estimate different keypoint representations in isolation. YOLOPose is trained using only the ℓ_1 loss in the case of BB and FPS representations, whereas for the IBB representation, ℓ_1 is combined with the cross-ratio loss described in Section 2.5. Table 4 reports

object pose estimation performance for the different representations. When used in conjecture with the EPnP solver, the FPS keypoints performed worse than all other representations. The reason is that the locations of FPS keypoints are less intuitive, making them more difficult to predict, especially for our proposed model that needs to deal with all objects in the YCB-V dataset. In contrast, the IBB keypoints representation yields the best performance. We conjecture that as the cross-ratio loss based on the prior geometric knowledge preserves the keypoints geometrically, this representation is the appropriate choice for our method where a single model is trained for all objects.

4.2 Effectiveness of RotEst

Table 4. Ablation study on YCB-V.

Method	ADD(-S)	AUC of ADD(-S)
FPS + EPnP	31.4	56.9
handpicked + EPnP	31.5	55.7
IBB + EPnP	**56.0**	**74.7**
IBB + EPnP for R; head for t	63.9	82.3
IBB + heads for R and t	**65.0**	**82.6**

After deciding on the keypoint representation, we compare the performance of the learnable feed-forward rotation and translation estimators against the analytical EPnP algorithm. The factors that impact rotation and translation components are different [31]. The rotation is highly affected by the object's appearance in a given image. In contrast, the translation is more vulnerable to the size and the location of the object in the image. Therefore, we decide to estimate rotation and translation separately. In Table 4, we report the quantitative comparison of the different variants. One can observe that using only the rotation from EPnP and directly regressing the translation improved the accuracy significantly. In general, RotEst performs slightly better than using EPnP orientation and direct translation estimation. Furthermore, the RotEst module and the translation estimators are straightforward MLPs and thus do not add much execution time overhead. This enables YOLOPose to perform inference in real-time. Moreover, to quantify the robustness of the RosEst module compared to the EPnP algorithm against the inaccuracies in keypoint estimation. We exclude the symmetric objects in the comparison. Figure 7, we present the comparison between the AUC of ADD and ADDS scores achieved by using the RotEst module and using the EPnP algorithm for recovering 6D pose from the estimated IBB keypoints. We discretize the average ℓ_2 pixel error in keypoint point estimation into bins of size two and average the AUC scores for all predictions corresponding to each bin. EPnP performs equally well in terms of both

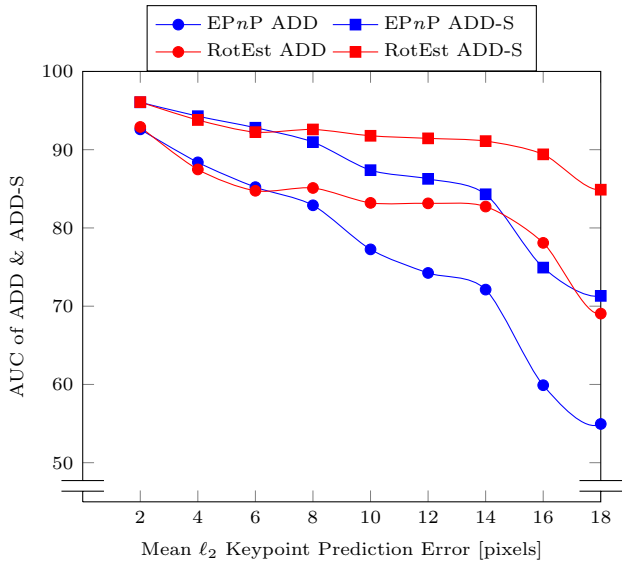


Fig. 7. Comparison of the pose estimation accuracy with respect to the keypoint estimation accuracy between EPnP and RotEst. In the case of highly accurate keypoint estimation, EPnP performs comparably to RotEst. However, the RotEst module is more robust against inaccuracies in keypoint estimation. Overall, RosEst performs better than EPnP.

the AUC of ADDS metrics compared to the RotEst module when the keypoint estimation accuracy is high. In the case of large keypoint estimation errors, the RotEst module demonstrates a significantly higher degree of robustness compared to the EPnP algorithm.

4.3 Dataset Size-Accuracy Trade-off

Vision Transformer models match or outperform CNN models in many computer vision tasks, but they require large datasets for pre-training [5, 12, 58]. Furthermore, obtaining large-scale 3D annotations are significantly harder than 2D annotations. Thus, the 3D datasets are supplemented with easy-to-acquire synthetic datasets. The YOLOPose architecture consists of a CNN backbone model for feature extraction and attention-based encoder-decoder module for set prediction. Learning set prediction is significantly challenging due to the additional overhead of finding the matching pairs between the ground-truth and the predicted sets and results in a low convergence rate. To mitigate this issue, we pre-train our model on the COCO dataset [33] for the task of object detection formulated as set prediction. The COCO dataset comprises 328,000 images with bounding annotations for 80 object categories. The COCO dataset pre-training enables faster convergence while training on the YCB-V dataset. To quantify the dataset size-accuracy trade-off in training our model for the task of joint object detection and pose estimation formulated as set prediction, we train our model with different subsets of the YCB-V dataset of varying sizes. As discussed in Section 3.1, YCB-V consists of 92 video sequences. 80 of which are used for training and the rest of them are used for testing. Additionally, Xiang et al. [59] provide 80,000 synthetic images for training as well. We created five different variants of the training set by using only a subset of the 80 training sequences. The first variant consists of only 16 video sequences and each subsequent variant consists of 16 additional video sequences added to the previous variant progressively. All five variants are supplemented with the complete set of synthetic images. We train one YOLOPoseA model for each of the dataset variants and evaluate the performance of the models on the test set consisting of twelve video sequences. In Fig. 8, we present the AUC of ADD-S and ADD-(S) scores as well as the *cardinality error* (CE), which is defined as the ℓ_1 error between the cardinality of the ground-truth and the predicted set. The model trained with the smallest training set variant consisting of only 16 video sequences achieves an AUC of ADD-S and ADD-(S) score of 83.5 and 75.4, respectively, whereas the model trained using the complete training videos achieves an AUC of ADD-S and ADD-(S) score of 91.2 and 83.3, respectively. The difference between the models trained using the smallest training set variant and the largest is even more significant in terms of the cardinality error—0.23 compared to 0.04. This demonstrates the need for large datasets with a wide range of scene configurations. Presented with smaller datasets with less variability in scene configuration, the YOLOPose model not only performs poorly in terms of pose estimation accuracy but also in terms of object detection accuracy.

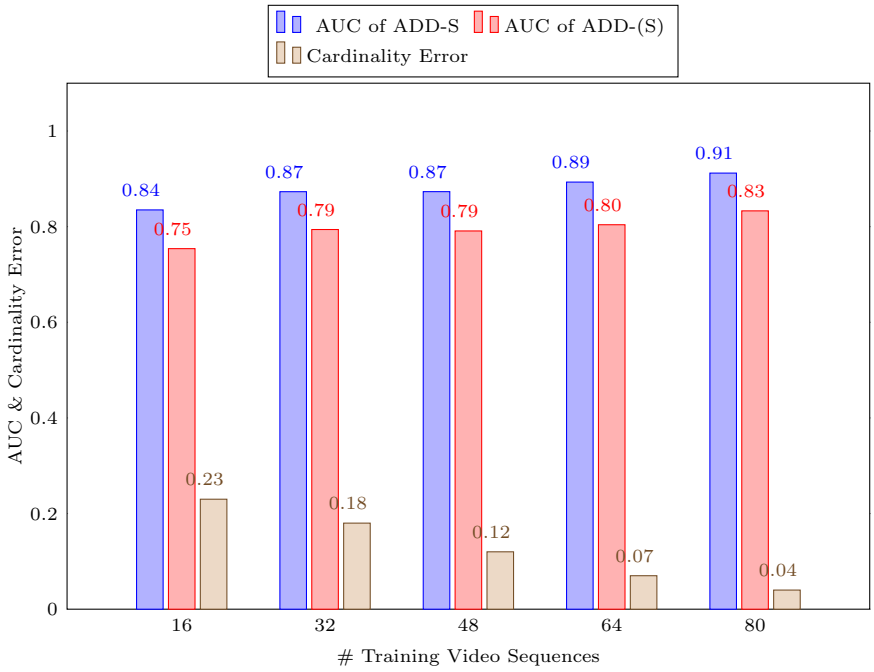
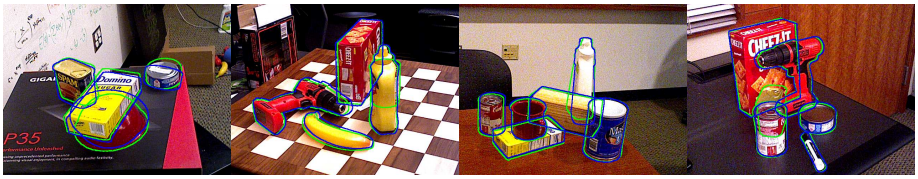


Fig. 8. Comparison of pose estimation and object detection accuracy using a different number of video sequences for training. The AUC scores are normalized to the range [0, 1].



(a) (b) (c) (d)

Fig. 9. Typical failure cases for the YOLOPose model. The pose estimation accuracy of our approach is hampered by occlusion. Ground truth and predicted object poses are visualized as object contours in green and blue colors, respectively.

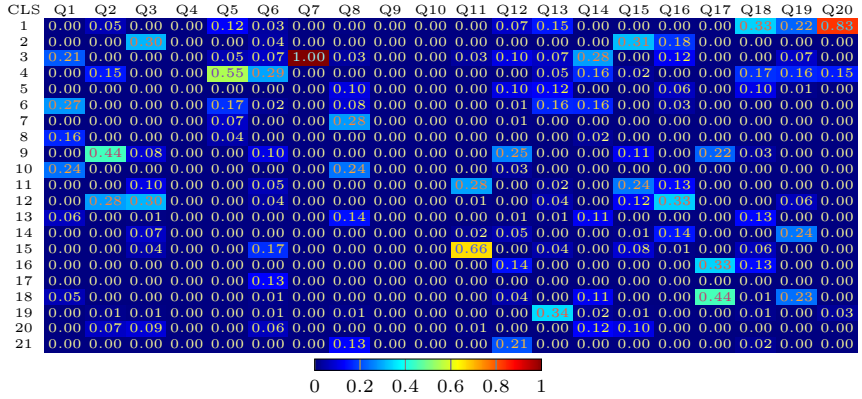


Fig. 10. Correlation between object queries and the detected object classes. Except for queries 7 and 20, the correlation is weak.

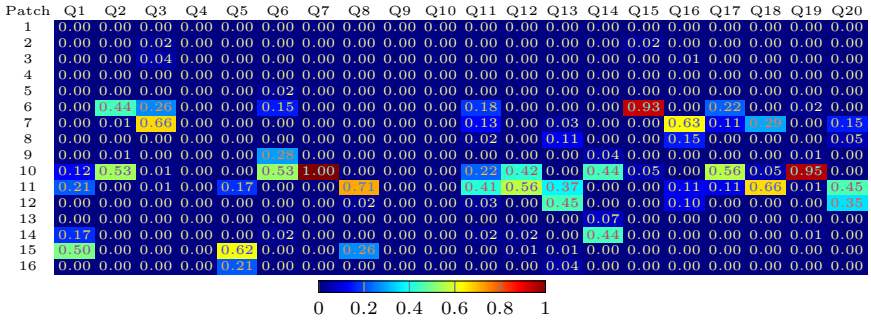


Fig. 11. Correlation between object queries and the image patch in which the object is detected. The images are divided into 4×4 patches. Compared to the correlation between object queries and the detected object classes shown in Fig. 10, the correlation between object queries and image patches is stronger.

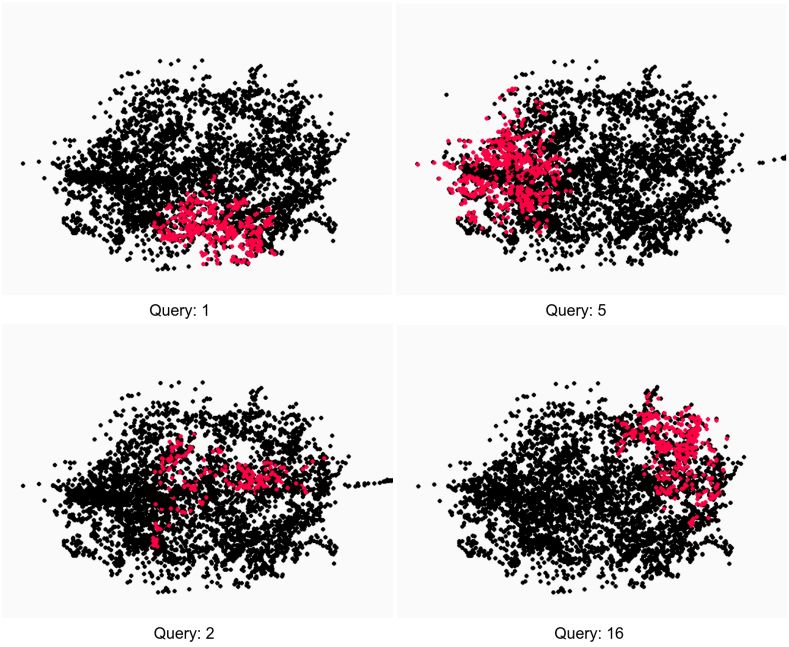


Fig. 12. Visualization of the center of the bounding boxes predicted by an object query. Black dots represent all the spatial positions of the ground-truth bounding boxes normalized to the image size present in the test dataset. Red dots represent the bounding boxes predicted by an object query. Object queries specialize in detecting objects in specific regions of the image.

5 Understanding Object Queries

To understand the role of the learned object queries in the YOLOPose architecture, we analyzed the correlation between the object queries and the detected object class ids as well as the object bounding boxes. Since we use only 20 object queries in the YOLOPose architecture—compared to 100 in the DETR [8] architecture—we can investigate the object queries individually in detail. To this end, we compute the correlation between object queries and class ids, and image patches that form a 4×4 grid. In Fig. 10, we visualize the correlation between the object queries and class ids. Except for queries 7 and 20, the correction is weak. In contrast, the correlation between the object queries and the image patch of the detected object is stronger (see Fig. 11). Note that queries 4, 9, and 10 do not correspond to any objects. This is the case only for the test dataset. In the case of the training dataset, all the object queries correspond to object detections. Moreover, we visualized the spatial location of the center of the bounding boxes predicted by object queries. In Fig. 12, we show exemplar visualizations. The visualizations also reveal that the object queries specialize in object detection in specific regions of the image.

6 Limitations

As shown in Table 1, and in Table 2, YOLOPose achieves pose estimation accuracy comparable to the state-of-the-art methods. Despite the impressive accuracy, occlusion remains a big challenge. In Fig. 9, we show examples of low-accuracy pose predictions—particularly in the case of partially-occluded objects. One of the commonly observed failure cases is the *bowl* object often predicted facing upwards even though *bowl* is placed downwards (See Fig. 9a). This is due to the limitation of the symmetry-aware SLoss (Eq. (8)). The SLoss is defined as the ℓ_2 distance between the closest model points of the object in the predicted and the ground truth poses. For some objects—*bowl*, for example—the 180° flip error is not penalized enough during training.

In terms of the dataset needed for training the YOLOPose model, since we formulate the task of joint object detection and 6D pose estimation as a set prediction problem, our approach needs pose annotation for all objects in the scene. Some of the commonly used datasets for training and evaluating pose estimation models like Linemod-Occluded [4] and Linemod [18] that provide pose annotations for just one object per scene in the training set cannot be used for training the YOLOPose model.

7 Discussion & Conclusion

We presented YOLOPose, a Transformer-based single-stage multi-object pose estimation method using keypoint regression. Our model jointly estimates bounding boxes, class labels, translation vectors, and pixel coordinates of 3D keypoints for all objects in the given input image. Employing the learnable RotEst module

to estimate object orientation from the predicted keypoint coordinates enabled the model to be end-to-end differentiable. We reported results comparable to the state-of-the-art approaches on the widely-used YCB-Video dataset and our model is real-time capable. Moreover, we presented an improved variant of the YOLOPose model in which the pose estimation FFNs input additional query output embeddings to generate improved pose estimates. Furthermore, we presented results on the role of object queries in the YOLOPose model. Based on the correlation matrix, we conclude that the object queries specialize in detecting objects in specific spatial locations.

8 Acknowledgment

This work has been funded by the German Ministry of Education and Research (BMBF), grant no. 01IS21080, project "Learn2Grasp: Learning Human-like Interactive Grasping based on Visual and Haptic Feedback".

Bibliography

[1] Amini, A., Periyasamy, A.S., Behnke, S.: T6D-Direct: Transformers for multi-object 6D object pose estimation. In: German Conference on Pattern Recognition (GCPR) (2021)

[2] Amini, A., Periyasamy, A.S., Behnke, S.: YOLOPose: Transformer-based multi-object 6D pose estimation using keypoint regression. In: International Conference on Intelligent Autonomous Systems (IAS) (2022)

[3] Basso, F., Menegatti, E., Pretto, A.: Robust intrinsic and extrinsic calibration of RGB-D cameras. Transactions on Robotics (T-RO) **34**(5), 1315–1332 (2018)

[4] Brachmann, E.: 6D Object Pose Estimation using 3D Object Coordinates [Data] (2020), doi:10.11588/data/V4MUMX, URL `https://doi.org/10.11588/data/V4MUMX`

[5] Cao, Y.H., Yu, H., Wu, J.: Training vision transformers with only 2040 images. In: European Conference on Computer Vision (ECCV), pp. 220–237 (2022)

[6] Cao, Z., Sheikh, Y., Banerjee, N.K.: Real-time scalable 6DOF pose estimation for textureless objects. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 2441–2448 (2016)

[7] Capellen, C., Schwarz, M., Behnke, S.: ConvPoseCNN: dense convolutional 6D object pose estimation. 15th International Conference on Computer Vision Theory and Applications (VISAPP) pp. 13909–13915 (2019)

[8] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European Conference on Computer Vision (ECCV), pp. 213–229 (2020)

[9] Chen, B., Parra, A., Cao, J., Li, N., Chin, T.J.: End-to-end learnable geometric vision by backpropagating PnP optimization. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8100–8109 (2020)

[10] Cohen, N., Shashua, A.: Inductive bias of deep convolutional networks through pooling geometry. In: International Conference on Learning Representations (ICLR) 2017, Toulon, France (2017)

[11] Cordonnier, J.B., Loukas, A., Jaggi, M.: On the relationship between self-attention and convolutional layers. In: International Conference on Learning Representations (ICLR) (2020)

[12] Gani, H., Naseer, M., Yaqub, M.: How to train vision transformer on small-scale datasets? In: 33rd British Machine Vision Conference (BMVC), BMVA Press (2022)

[13] Gao, X., Hou, X., Tang, J., Cheng, H.: Complete solution classification for the perspective-three-point problem. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) **25**, 930–943 (2003)

[14] Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, 2 edn. (2004)

[15] He, Y., Huang, H., Fan, H., Chen, Q., Sun, J.: FFB6D: A full flow bidirectional fusion network for 6D pose estimation. In: IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR), pp. 3003–3013 (2021)
[16] He, Y., Sun, W., Huang, H., Liu, J., Fan, H., Sun, J.: PVN3D: A deep point-wise 3D keypoints voting network for 6DoF pose estimation. In: IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR), pp. 11632–11641 (2020)
[17] Hinterstoisser, S., Cagniart, C., Ilic, S., Sturm, P., Navab, N., Fua, P., Lepetit, V.: Gradient response maps for real-time detection of textureless objects. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **34**(5), 876–888 (2011)
[18] Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., Navab, N.: Model-based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In: Asian conference on computer vision (ACCV), pp. 548–562, Springer (2013)
[19] Hodaň, T., Sundermeyer, M., Drost, B., Labbé, Y., Brachmann, E., Michel, F., Rother, C., Matas, J.: BOP challenge 2020 on 6D object localization. In: European Conference on Computer Vision (ECCV), pp. 577–594 (2020)
[20] Holzer, S., Hinterstoisser, S., Ilic, S., Navab, N.: Distance transform templates for object detection and pose estimation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1177–1184 (2009)
[21] Hosang, J., Benenson, R., Schiele, B.: Learning non-maximum suppression. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4507–4515 (2017)
[22] Hu, Y., Fua, P., Wang, W., Salzmann, M.: Single-stage 6D object pose estimation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2930–2939 (2020)
[23] Hu, Y., Hugonot, J., Fua, P., Salzmann, M.: Segmentation-driven 6D object pose estimation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3385–3394 (2019)
[24] Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Research Logistics Quarterly **2**, 83–97 (1955)
[25] Kutulakos, K.N., Steger, E.: A theory of refractive and specular 3D shape by light-path triangulation. International Journal of Computer Vision (IJCV) **76**, 13–29 (2008)
[26] Labbe, Y., Carpentier, J., Aubry, M., Sivic, J.: CosyPose: Consistent multi-view multi-object 6D pose estimation. In: European Conference on Computer Vision (ECCV) (2020)
[27] LeCun, Y., Bengio, Y., et al.: Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks p. 255–258 (1995)
[28] Lepetit, V., Moreno-Noguer, F., Fua, P.: EPnP: An accurate o(n) solution to the PnP problem. International Journal of Computer Vision (IJCV) **81**(2), 155 (2009)
[29] Li, S., Yan, Z., Li, H., Cheng, K.T.: Exploring intermediate representation for monocular vehicle pose estimation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1873–1883 (2021)

[30] Li, Y., Wang, G., Ji, X., Xiang, Y., Fox, D.: DeepIM: Deep iterative matching for 6D pose estimation. In: European Conference on Computer Vision (ECCV), pp. 683–698 (2018)
[31] Li, Z., Wang, G., Ji, X.: CDPN: Coordinates-based disentangled pose network for real-time RGB-based 6-DoF object pose estimation. In: ICCV, pp. 7678–7687 (2019)
[32] Li, Z., Yeh, Y.Y., Chandraker, M.: Through the looking glass: Neural 3D reconstruction of transparent shapes. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1262–1271 (2020)
[33] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: European Conference on Computer Vision (ECCV), pp. 740–755 (2014)
[34] Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (ICLR) (2017)
[35] Lysenkov, I., Eruhimov, V., Bradski, G.: Recognition and pose estimation of rigid transparent objects with a kinect sensor. Robotics **273**(273-280), 2 (2013)
[36] Maeno, K., Nagahara, H., Shimada, A., Taniguchi, R.I.: Light field distortion feature for transparent object recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2013)
[37] Manhardt, F., Kehl, W., Navab, N., Tombari, F.: Deep model-based 6D pose refinement in RGB. In: European Conference on Computer Vision (ECCV), pp. 800–815 (2018)
[38] Oberweger, M., Rad, M., Lepetit, V.: Making deep heatmaps robust to partial occlusions for 3D object pose estimation. In: European Conference on Computer Vision (ECCV) (2018)
[39] Pavlakos, G., Zhou, X., Chan, A., Derpanis, K.G., Daniilidis, K.: 6-DOF object pose from semantic keypoints. In: IEEE International conference on robotics and automation (ICRA), pp. 2011–2018 (2017)
[40] Peng, S., Liu, Y., Huang, Q., Zhou, X., Bao, H.: PVNet: Pixel-wise voting network for 6DOF pose estimation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4561–4570 (2019)
[41] Periyasamy, A.S., Schwarz, M., Behnke, S.: Robust 6D object pose estimation in cluttered scenes using semantic segmentation and pose regression networks. In: International Conference on Intelligent Robots and Systems (IROS) (2018)
[42] Periyasamy, A.S., Schwarz, M., Behnke, S.: Refining 6D object pose predictions using abstract render-and-compare. In: IEEE-RAS International Conference on Humanoid Robots (Humanoids), pp. 739–746 (2019)
[43] Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30, Curran Associates, Inc. (2017)
[44] Rad, M., Lepetit, V.: BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using

depth. In: International Conference on Computer Vision (ICCV), pp. 3828–3836 (2017)
[45] Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7263–7271 (2017)
[46] Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems (NeurIPS), vol. 28 (2015)
[47] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union: A metric and a loss for bounding box regression. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 658–666 (2019)
[48] Rothganger, F., Lazebnik, S., Schmid, C., Ponce, J.: 3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. International journal of computer vision (IJCV) **66**(3), 231–259 (2006)
[49] Schwarz, M., Lenz, C., García, G.M., Koo, S., Periyasamy, A.S., Schreiber, M., Behnke, S.: Fast object learning and dual-arm coordination for cluttered stowing, picking, and packing. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 3347–3354 (2018)
[50] Staranowicz, A.N., Brown, G.R., Morbidi, F., Mariottini, G.L.: Practical and accurate calibration of RGB-D cameras using spheres. Computer Vision and Image Understanding **137**, 102–114 (2015)
[51] Stewart, R., Andriluka, M., Ng, A.Y.: End-to-end people detection in crowded scenes. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2325–2333 (2016)
[52] Sundermeyer, M., Hodan, T., Labbe, Y., Wang, G., Brachmann, E., Drost, B., Rother, C., Matas, J.: BOP challenge 2022 on detection, segmentation and pose estimation of specific rigid objects. preprint arXiv:2302.13075 (2023)
[53] Tekin, B., Sinha, S.N., Fua, P.: Real-time seamless single shot 6D object pose prediction. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
[54] Thalhammer, S., Leitner, M., Patten, T., Vincze, M.: PyraPose: feature pyramids for fast and accurate object pose estimation under domain shift. In: International Conference on Robotics and Automation (ICRA), pp. 13909–13915, IEEE (2021)
[55] Tulsiani, S., Malik, J.: Viewpoints and keypoints. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1510–1519 (2015)
[56] Wang, C., Xu, D., Zhu, Y., Martín-Martín, R., Lu, C., Fei-Fei, L., Savarese, S.: DenseFusion: 6D object pose estimation by iterative dense fusion. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3343–3352 (2019)
[57] Wang, G., Manhardt, F., Tombari, F., Ji, X.: GDR-Net: Geometry-guided direct regression network for monocular 6D object pose estimation. In:

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
[58] Wang, W., Zhang, J., Cao, Y., Shen, Y., Tao, D.: Towards data-efficient detection transformers. In: European Conference on Computer Vision (ECCV), pp. 88–105 (2022)
[59] Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. In: Robotics: Science and Systems (RSS) (2018)
[60] Xu, D., Anguelov, D., Jain, A.: Pointfusion: Deep sensor fusion for 3D bounding box estimation. In: IEEE conference on Computer Vision and Pattern Recognition (CVPR), pp. 244–253 (2018)
[61] Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: IEEE/CVF International Conference on Computer Vision (ICCV), pp. 16259–16268 (2021)
[62] Zhou, Y., Barnes, C., Lu, J., Yang, J., Li, H.: On the continuity of rotation representations in neural networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5745–5753 (2019)