IEEE International Conference on Robotic Computing (IRC), Laguna Hills, USA, December 2023.

Efficient Multi-Object Pose Estimation using Multi-Resolution Deformable Attention and Query Aggregation

Arul Selvam Periyasamy*
Autonomous Intelligent Systems
University of Bonn
Bonn, Germany
Email: periyasa@ais.uni-bonn.de

Vladimir Tsaturyan*
Autonomous Intelligent Systems
University of Bonn
Bonn, Germany
Email: vladimir@gmail.com

Sven Behnke
Autonomous Intelligent Systems
University of Bonn
Bonn, Germany
Email: behnke@cs.uni-bonn.de

***Abstract*—Object pose estimation is a long-standing problem in computer vision. Recently, attention-based vision transformer models have achieved state-of-the-art results in many computer vision applications. Exploiting the permutation-invariant nature of the attention mechanism, a family of vision transformer models formulate multi-object pose estimation as a set prediction problem. However, existing vision transformer models for multi-object pose estimation rely exclusively on the attention mechanism. Convolutional neural networks, on the other hand, hard-wire various inductive biases into their architecture. In this paper, we investigate incorporating inductive biases in vision transformer models for multi-object pose estimation, which facilitates learning long-range dependencies while circumventing the costly global attention. In particular, we use multi-resolution deformable attention, where the attention operation is performed only between a few deformed reference points. Furthermore, we propose a query aggregation mechanism that enables increasing the number of object queries without increasing the computational complexity. We evaluate the proposed model on the challenging YCB-Video dataset and report state-of-the-art results.**

I. Introduction

Object pose estimation is the task of predicting the position and the orientation of objects with respect to the sensor coordinate frame. It plays a vital role in many autonomous robotic systems and augmented and virtual reality applications. Occlusion, object reflectance properties, and lighting conditions increase the complexity of the task. Despite the recent deep-learning-driven progress, object pose estimation remains challenging. RGB-D methods, in general, perform better than the RGB-only methods. The depth information facilitates easier learning of the geometric features compared to the RGB input. However, transparent and reflecting objects present significant challenges for RGB-D cameras. The resolution and frame rate of the RGB-D sensors are limited, compared to RGB cameras. Additionally, in large-scale real-world deployment, RGB-D sensors need calibration, which is time and resource-consuming. Motivated by these limitations of the RGB-D sensors, we focus on RGB methods in this paper.

In the last decade, convolutional neural networks (CNNs) have become the standard machine learning tool for solving many computer vision tasks. The success of CNNs is largely attributed to the inductive biases incorporated into their architectural design [1]–[4]. Lately, transformer-based architectures have found great success in many computer vision tasks. The core component of the transformer model is the multi-head attention mechanism that allows learning dependencies in the input data over a long range. Carion *et al.* [5] introduced DETR, an end-to-end differentiable architecture for object detection combining a CNN model for feature extraction and a transformer-based encoder-decoder model for generating a set of object predictions. Inspired by their

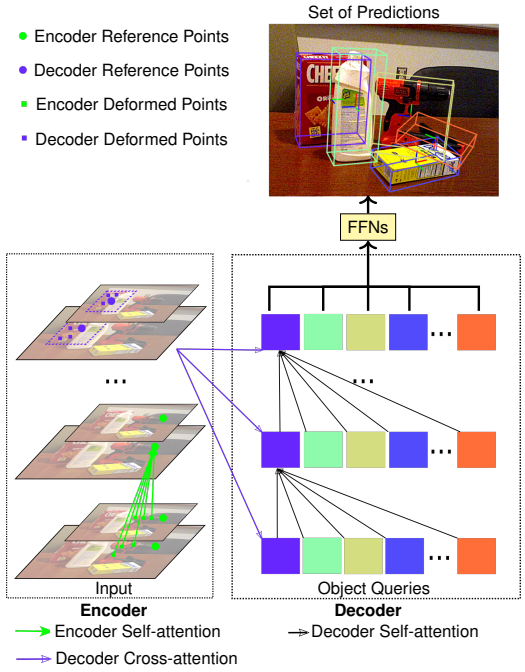

Fig. 1. Multi-resolution deformable attention model. Our architecture utilizes an attention-based encoder-decoder design. In the encoder, image patch features of multiple resolutions are extracted using linear projection, and the self-attention mechanism is used to generate encoder embeddings. In the decoder, cross-attention is performed between the encoder embeddings and learned embeddings known as *object queries* to generate object embeddings. The object queries are initialized randomly at the beginning of training and learned jointly with the training objective. During inference, the object queries remain fixed. In contrast to standard attention, in which attention is computed between all image patch features, in deformable attention, the attention operation is performed only between the deformed reference points. From the object embeddings, object predictions are generated using feed-forward neural networks (FFNs) in parallel. Object pose predictions generated by our model are visualized using 3D bounding boxes.

* Equal contribution

work, we formulate multi-object pose estimation as a set prediction problem. However, instead of relying completely on the computationally costly global attention mechanism, to facilitate efficient architectures, we investigate CNN-like inductive biases in our design.

In this work, we present the following contributions:

- an efficient multi-resolution deformable attention model for multi-object pose estimation,
- a query aggregation mechanism to increase the number of object queries without increasing the computational complexity, and
- a thorough evaluation on the YCB-Video dataset [6] with state-of-the-art results.

II. Related Work

A. Object Pose Estimation

The prominent deep learning methods can be classified as direct regression [6]–[8], keypoint-based [9]–[11], and refinement-based [12]–[14]. The direct regression methods formulate pose estimation as a regression task. Given the image crop consisting of the target object, the direct regression methods predict the 6D pose parameters. In contrast, the keypoint-based methods estimate the pixel coordinates of 3D keypoints in the image and with known 2D-3D correspondence, the 6D pose is recovered employing the perspective-*n*-point (P*n*P) algorithm. The refinement-based methods are orthogonal to both direct regression and keypoint-based methods in terms of the approach. They formulate pose estimation as a problem of pose refinement using the render-and-compare framework. With the notable exception of Capellen *et al.* [15] and Hu *et al.* [11], most of the CNN architectures for object pose estimation decouple pose estimation from object detection and follow a two-stage approach in which the 2D bounding boxes are estimated in the first stage; and in the second stage, only the crop containing the target object is processed to estimate the 6D pose. To realize an end-to-end differentiable architecture for pose estimation, these methods rely on modules like non-maximum suppression (NMS), region of interest (ROI) pooling, anchor box proposal [16]–[18], differentiable implementations of the Hough transform [6], [15], and random sample consensus (RANSAC) [19].

B. Vision Transformers

Vaswani *et al.* [20] introduced the Transformer architecture with a multi-head attention mechanism to model long-range dependencies in natural language processing and achieved significant improvements on a variety of tasks. The success of the transformer architecture inspired many approaches that incorporate attention mechanisms to solve computer vision tasks either by supplementing or by replacing CNN modules. Dosovitskiy *et al.* [21] introduced Vision Transformer (ViT), an architecture without any CNN modules. While ViT achieved impressive results using a simple architecture, the computational cost of attention was higher than CNN modules. Subsequent methods showed that incorporating CNN-like inductive biases into the vision transformer architecture reduced computational costs and improved performance across a wide range of tasks. Liu *et al.* [22] introduced the Swin Transformer model, which limits self-attention to local non-overlapping shifted windows and uses cross-attention for cross-window connections. They also incorporated hierarchical processing into their architecture. Yang *et al.* [23] introduced focal self-attention, an efficient attention mechanism in which the pixels in the closest surrounding tokens are attended at a fine granularity but the pixels far away are processed at a coarse granularity. DETR [5] formulated multi-object detection as a set prediction problem and proposed a transformer-based encoder-decoder architecture for set prediction. Zhu *et al.* [24] proposed deformable attention to reduce computational cost in DETR-like architectures. Amini *et al.* [7] extended DETR for multi-object 6D pose regression. Amini *et al.* [25] and Periyasamy *et al.* [26] introduced YOLOPose, a DETR-like architecture for keypoint-based multi-object pose estimation. In contrast to the multi-stage pose estimation models discussed in Section II-A, they realized a single-stage multi-object pose estimation without using NMS, ROI, or anchor box proposal modules.

III. Method

In this section, we introduce the multi-object pose estimation as a set prediction task formulation and briefly describe the existing YOLOPose model [25] that we use as the baseline. Later, we discuss the improvements to the baseline model incorporating various inductive biases.

Multi-head attention (MHA) introduced by Vaswani *et al.* [20] performs scaled dot-product attention between the query-key pairs. Let $q \in \Omega_q$ be a query element with feature $z_q \in \mathbf{R}^d$ and $k \in \Omega_k$ be a key element with feature $x_k \in \mathbf{R}^d$, where d is the dimension of the features and $\Omega_{q/k}$ are the sets of query and key elements, respectively. Then the multi-head attention feature is computed as:

$$\text{MHA}(z_q, x) \quad = \quad \sum_{m=1}^{M} W_m \left[\sum_{k \in \Omega_k} A_{mqk} \cdot W_m^{'} x_k \right], \quad (1)$$

where $m \in M$ represents the attention head, $W_m^{'} \in \mathbf{R}^{d_v \times d}$ and $W_m \in \mathbf{R}^{d \times d_v}$ are learnable projection parameters, $d_v = d/M$, and A represents the normalized attention weight. MHA is the core component of the transformer architecture.

Formulating pose estimation as a set prediction problem enables multi-object pose estimation in which objects are detected and their 6D pose are estimated in one single forward pass. Given the set of object predictions $\hat{\mathcal{Y}}$ of cardinality N and ground-truth set $\mathcal{Y}$ padded with Ø class, we search for the optimal permutation $\hat{\sigma}$ among the possible permutations $\sigma \in \mathfrak{S}_N$ defined by the matching cost $\mathcal{L}_{match}$. Formally,

$$\hat{\sigma} = \underset{\sigma \in \mathfrak{S}_N}{\arg\min} \sum_{i}^{N} \mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)}). \quad (2)$$

The transformer-based baseline model YOLOPose utilizes a CNN backbone to extract image features and an encoder-decoder architecture followed by feed-forward networks for

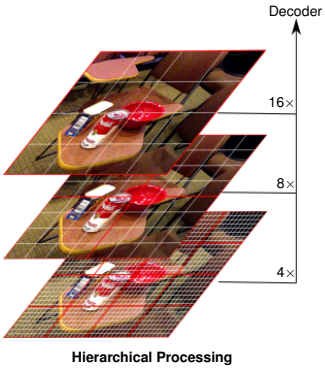

Fig. 2. Hierarchical processing of image features. White grids represent the image patches and red grids represent the self-attention windows. Layers are grouped into blocks. The size of the self-attention window and the number of feature maps remain fixed for all the layers inside each block but increase through the hierarchy. Restricting attention to a local window helps reduce the computational complexity.

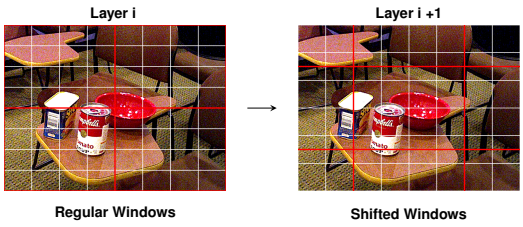

Fig. 3. Shifting window local attention mechanism. Self-attention is restricted to the local window shown in red grids. The windows are shifted between the layers, enabling global interaction within each block.

generating a set of object predictions. Since the transformer architecture is permutation invariant, the image features are supplemented with positional encoding [20]. The encoder performs self-attention between the image features. The decoder performs cross-attention between the encoder output and the learned embeddings known as *object queries*. Object queries are initialized randomly at the beginning of the training and are learned along with the training objective. While performing cross-attention, the encoder output embeddings act as the *key* and the *value*, whereas the object queries act as the *query*. The resulting object embeddings are processed in parallel using multi-layer perceptrons (MLPs) to generate object predictions. During inference, the object queries are fixed. While YOLOPose achieves impressive results, it relies solely on the attention mechanism to learn joint object detection and pose estimation. Thus, it does not benefit from incorporating inductive biases into its architecture. To this end, we discuss various design strategies for imbuing inductive biases into the YOLOPose model in the following sections.

A. Local Hierarchical Shifting Window Attention

The encoder module in YOLOPose utilizes self-attention between the image features extracted by a CNN backbone. This enables aggregating information from all spatial locations in the encoder module. In terms of the design, one of the main shortcomings of YOLOPose is that the backbone module is exclusively CNN-based and the encoder module is exclusively attention-based. Thus, the backbone module does not benefit from the self-attention mechanism. *Vision Transformer models* (ViT) [21] addressed this issue by designing a backbone model based exclusively on the transformer architecture. Raghu *et al.* [27] noted that in ViT, the similarity of the lower layer features and the higher layer features is stronger than in the case of CNN-based ResNet model. Based on this observation, the authors concluded that the self-attention mechanism along with the skip connections enables lower layers in the ViT model to learn global features. However, ViT suffers from two major limitations:

1) feature maps used in the model are low resolution and
2) complexity of the attention mechanism increases quadratically.

These factors limit the suitability of ViT as a backbone model. To address this issue, Liu *et al.* [22] proposed to incorporate hierarchical processing and local sliding window attention in the design of the transformer model. The pixels are divided into crops. All pixels belonging to an image crop are projected linearly to features of dimension d. The hierarchical processing of the features is shown in Fig. 2. Unlike ViT, in which all the image patches in any particular layer interact with all other patches, the interaction is restricted to a local window (shown in Fig. 3). The layers of the model are grouped into blocks. The size of the attention window and the number of feature maps increases progressively higher in the hierarchy. The attention window is shifted between the layers in the same block. This ensures the global interaction of the features within each block. The hierarchical processing of features and limiting the attention to a local window enables linearly scaling computational complexity. We refer to our implementation of the pose estimation model with local hierarchical shifting window attention as Model-A.

B. Deformable Multi-resolution Attention

The attention mechanism offers a simple yet effective mechanism to model long-range dependencies in input tokens. Despite the advantages the attention mechanism offers for computer vision tasks, the computational cost of MHA, especially for high-resolution images, remains high. To address this issue, Zhu *et al.* [24] introduced *deformable multi-head attention* (DMHA). In Fig. 1, we depict the DMHA model for pose estimation. For a query token z, instead of computing attention over all the key tokens Ω_x, DMHA computes attention over a small set of 2D reference points $p \in \Omega_p$. The reference points are allowed to deform and the deformation is learned from the input tokens. Formally, MHA, introduced in Eq. (1), is extended as,

$$\text{DMHA}(z_q, p_q, x) = \sum_{m=1}^{M} W_m \left[\sum_{k=1}^{K} A_{mqk} \cdot W_m^{'} x(p_q + \Delta p_{mqk}) \right], \quad (3)$$

where k represents the sampled key tokens and Δp_{mqk} denotes deformation offset. Bilinear interpolation enables fractional

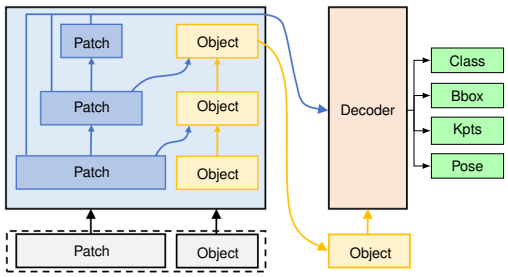

Fig. 4. Early fusion of object queries. The patch embeddings and the object queries interact within the encoder module at all layers. Thus, not only the high-level features interact with the object queries but the features of all resolutions.

offsets. Furthermore, we incorporate multi-resolution feature processing (MR-DMHA) into the deformable attention:

$$\text{MR-DMHA}(z_q, p_q, \{x\}_{l=1}^{L}) = \sum_{m=1}^{M} W_m \left[\sum_{l=1}^{L} \sum_{k=1}^{K} A_{mqk} \cdot W_m^{'} x^l(\phi(\hat{p}_q) + \Delta p_{mqk}) \right], \quad (4)$$

where l represents the feature level, and the function $\phi(\hat{p}_q)$ re-scales the pixel coordinates corresponding to the feature level. We refer to our multi-object pose estimation model based on MR-DMHA as Model-B.

C. Early Fusion of Object Queries

While the MR-DMHA enables efficient attention with linear complexity, the encoder module interacts with the object queries only at the final encoder layer. This results in the encoder module learning generic features in the earlier layers and only the final layer learning features relevant to object prediction. Enabling encoder-object query interaction at the early layers helps the encoder to focus on features more relevant to the object predictions. Following Song *et al.* [28], we introduce cross-attention between object queries and patch embeddings early in the encoder module. In each encoder layer, self-attention is performed between the patch embeddings and additionally, cross-attention is performed between object queries and patch embeddings of the previous layer, as shown in Fig. 4. In the decoder module, cross-attention is performed between the aggregated patch embeddings from the encoder and the object query to generate object embeddings. The object embeddings are processed by the FFNs to generate object predictions. We call this variant Model-C.

D. Query Aggregation

The learned object queries play a crucial role in DETR-like architectures. Despite their importance, the object queries are not fully understood. One of the major reasons behind this lack of understanding is the fact that neither the architecture itself nor the loss function contains any mechanism to bind the object queries specifically to object classes or locations. Zhang *et al.* [29] observed that using more object queries improves model accuracy. In our models, the number of object

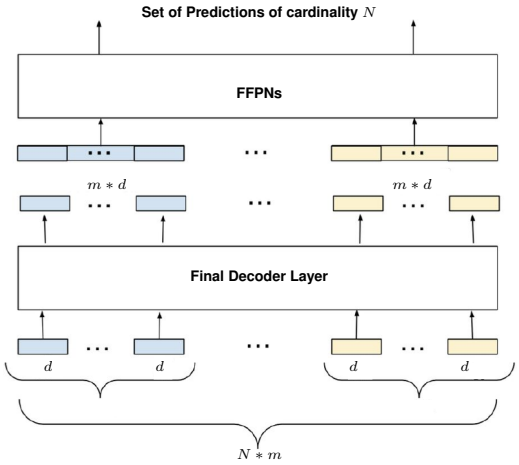

Fig. 5. Query Aggregation. In contrast to the standard approach of using the same number of object queries as the cardinality of the set N, in the query aggregation approach, we set the number of object queries to $N \times m$. The query aggregation factor m is a hyperparameter. $N \times m$ decoder output embeddings of dimension d are concatenated to form N object embeddings of dimension $d \times m$ that are processed by the FFNs to generate N object predictions.

queries equals the cardinality of the set we predict. Thus, increasing the number of object queries also increases the computation cost of bipartite matching and thus, the overall training time. We propose a novel approach for increasing the number of object queries while keeping the computational cost low. We call this method *query aggregation* (shown in Fig. 5). To generate a set of predictions of cardinality N, the standard approach uses N object queries of dimension d. In contrast to the standard approach, in the query aggregation approach, we set the number of object queries to $N \times m$, where the aggregation factor m is a hyperparameter. After the last decoder layer, we concatenate each set of m embeddings to generate one object embedding. Thus, from $N \times m$ output embeddings we generate N object embeddings. The FFNs process the N object embeddings to generate a set of object predictions. By decoupling the number of decoder output embeddings and the number of object embeddings, we enable a larger number of embeddings in the decoder layer without increasing the cardinality of the predicted set.

E. Refinement of the Object Predictions

The decoder module in the standard architecture consists of six decoder layers. The output embeddings of each of the decoder layer serves an input for the subsequent decoder layer. The output embeddings of the final decoder layer are processed by the FFNs to generate object predictions. Instead of generating object predictions directly once, generating an initial prediction and refining it to generate the final object predictions allows the model to iteratively improve the initial predictions as well as the overall accuracy. The design of decoder layers naturally suits refinement. In Fig. 6, we show the refinement of the object predictions in the decoder module. Each decoder layer contains its own independent set of FFNs. The first decoder layer performs cross-attention between the

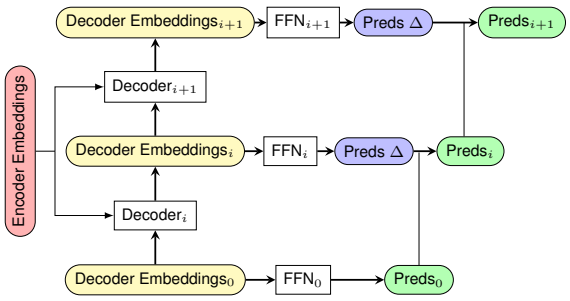

Fig. 6. Prediction refinement. At each decoder layer, only a small Δ is predicted on top of the previous predictions enabling the refinement of predictions through the decoder layers.

encoder embeddings and the object queries to generate decoder embeddings that are processed by a corresponding set of FFNs to generate object predictions. The FFNs of subsequent decoder layers generate only a small Δ to refine the predictions made by the previous layer. This allows for the model to iteratively refine the initial predictions and generate more accurate final predictions.

F. Loss Function

Our model is trained to minimize the Hungarian loss between the predicted and the ground-truth sets. The Hungarian loss is the dissimilarity measure between the matched predicted and the ground-truth pairs. To establish the matching pairs, we employ the differentiable bipartite matching algorithm [5], [30].

The Hungarian loss we use consists of four components: bounding box, class probability, translation, and key points. However, the matching cost we employ comprises only the bounding box and the class probability components.

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

$$\mathcal{L}_{Hungarian}(\mathcal{Y}, \hat{\mathcal{Y}}_\sigma) = \sum_i^N \mathcal{L}_{class} + \mathbb{1}_{c_i \neq \emptyset} \Big[\mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}(i)}) + \lambda_{kp} \mathcal{L}_{kp}(k_i, \hat{k}_{\hat{\sigma}(i)}) + \lambda_{pose} \mathcal{L}_{pose}(R_i, t_i, \hat{R}_{\hat{\sigma}(i)}, \hat{t}_{\hat{\sigma}(i)}) \Big]. \quad (5)$$

The individual components of the Hungarian loss are the following.

1) Class Probability Loss:

$$\mathcal{L}_{class}(c_i, \hat{p}_{\sigma(i)}) = -\log \hat{p}_{\hat{\sigma}(i)}(c_i), \quad (6)$$

The standard negative log-likelihood is used to measure the dissimilarity of the predicted class probability scores $\hat{p}_\sigma$ and the ground-truth class labels c, represented as one-hot encoding. The images in the YCB-Video dataset [6] comprise between three and eleven objects per frame. In our model, the cardinality of the set N is set to 20. Thus, padding the ground-truth set with Ø class creates a heavy class imbalance between the Ø class and the rest of the classes. To counter the class imbalance, we weigh the loss for the Ø class with a factor of 0.1.

2) Bounding Box Loss: A weighted combination of the Generalized IoU (GIoU) [31] and ℓ_1-loss is employed as the measure of dissimilarity between the predicted and the ground-truth bounding boxes, b_i and $b_{\sigma(i)}$, respectively:

$$\mathcal{L}_{box}(b_i, \hat{b}_{\sigma(i)}) = \alpha \mathcal{L}_{iou}(b_i, \hat{b}_{\sigma(i)}) + \beta ||b_i - \hat{b}_{\sigma(i)}||, \quad (7)$$

where α and β are hyperparameters.

3) Keypoint Loss: The keypoint loss is a combination of ℓ_1 loss and cross-ratio consistency loss:

$$\mathcal{L}_{kp}(K_i, \hat{K}_{\hat{\sigma}(i)}) = \gamma ||K_i - \hat{K}_{\hat{\sigma}(i)}||_1 + \delta \mathcal{L}_{\mathcal{CR}}, \quad (8)$$

where K_i and $\hat{K}_{\hat{\sigma}(i)}$ are ground truth and the predicted keypoint coordinates, respectively, $\mathcal{L}_{\mathcal{CR}}$ is the Smoothℓ_1 loss between the ground truth and predicted cross-ratio, and γ & δ are weighting factors.

4) Pose Loss: We compute the dissimilarity between the ground-truth pose parameters R_i and $\mathbf{t}_i$, and the predicted pose parameters $\hat{R}_{\sigma(i)}$ and $\hat{\mathbf{t}}_{\sigma(i)}$ using the disentangled pose loss:

$$\mathcal{L}_{pose}(R_i, \mathbf{t}_i, \hat{R}_{\sigma(i)}, \hat{\mathbf{t}}_{\sigma(i)}) = \mathcal{L}_{rot}(R_i, \hat{R}_{\sigma(i)}) + ||\mathbf{t}_i - \hat{\mathbf{t}}_{\sigma(i)}||_1. \quad (9)$$

We employ PLoss and SLoss [6] for the rotation component, and ℓ_1 loss for the translation component:

$$\mathcal{L}_{rot} = \begin{cases} \frac{1}{|\mathcal{M}_i|} \sum_{\mathrm{x}_1 \in \mathcal{M}_i} \min_{\mathrm{x}_2 \in \mathcal{M}_i} ||R_i \mathrm{x}_1 - \hat{R}_{\sigma(i)} \mathrm{x}_2||_1 \text{ if symmetric,} \\ \frac{1}{|\mathcal{M}_i|} \sum_{\mathrm{x} \in \mathcal{M}_i} ||R_i \mathrm{x} - \hat{R}_{\sigma(i)} \mathrm{x}||_1 \text{ otherwise,} \end{cases} \quad (10)$$

where $\mathcal{M}_i$ indicates the subsampled 3D model points.

IV. Experiments

A. Dataset

We evaluate the proposed method on the challenging YCB-Video dataset [6]. The dataset comprises 92 video sequences of the cluttered tabletop scenes. Each video sequence consists of a randomly selected subset of objects from a total of 21 objects placed in a random configuration. The 6D pose annotations and the bounding box annotations were generated using a semi-automatic procedure in which the first frame of each video sequence is manually annotated and extrapolated for the rest of the frames employing visual odometry techniques. Overall, the dataset consists of 133,827 images. Out of the 92 video sequences, twelve are used for testing and the remaining ones are used for training and validation. We also use the COCO dataset for pre-training our model, which consists of 123,287 images and 886,284 bounding box annotations belonging to 80 categories.

B. Metrics

We report the standard area under the curve (AUC) ADD and ADD-S metrics [6], computed using a varying threshold between 10 cm to 1 cm. Given the ground-truth 6D pose

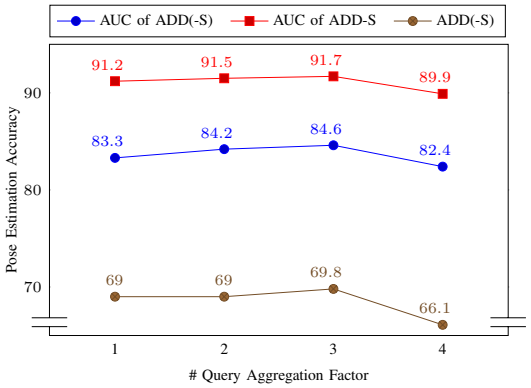

Fig. 7. Results from query aggregation experiment.

TABLE I
EFFECT OF PRETRAINING ON THE COCO DATASET FOR OBJECT DETECTION.

Method	AUC of ADD-S	AUC of ADD(-S)	ADD(-S)
Without pretraining	71.0	82.7	42.5
With pretraining	**81.4**	**90.1**	**61.1**

annotation with rotation and translation components R and $\mathbf{t}$, and the predicted rotation and translation components $\hat{R}$ and $\hat{\mathbf{t}}$, the ADD metric is the average ℓ_2 distance between the subsampled mesh points $\mathcal{M}$ in the ground truth and the predicted pose, whereas the symmetry-aware ADD-S metric is the average ℓ_2 distance between the closest subsampled mesh points $\mathcal{M}$ in the ground-truth and predicted pose.

The ADD(-S) metric corresponds to ADD-S for symmetric objects and ADD for non-symmetric objects.

C. Implementation Details

We implement our models using the PyTorch [32] library. Our models are trained using NVIDIA A100 GPUs with 40 GBs of memory. The size of the input images is 640×480. The models are trained for 200 epochs with a batch size of 32 using AdamW optimizer with an initial learning rate of 10^{-4}. During training, we supplement the training images with the synthetic images provided with the dataset. The hyperparameters α, β, γ, and δ in Eqs. (7) and (8) are set to 2, 5, 10, and 1, respectively.

D. Results

In Fig. 8, we show pose estimation for exemplar scenes; and in Table II, we report the quantitative comparison of our model with the state-of-the-art RGB pose estimation models. The best-performing variant of our model is based on deformable multi-resolution attention (Model-B) discussed in Section III-B utilizing 16 deformable points. Our model achieves an impressive AUC of ADD-S score of 92.0 and AUC of ADD(-S) score of 84.7. Among the object categories, our model performs worse for *extra-large clamp* and *scissors*– both exhibit challenging geometry. In Fig. 9, we show typical

Fig. 8. Qualitative results on YCB-Video dataset [6]. Ground truth and predicted object poses are visualized as object contours in green and blue colors, respectively.

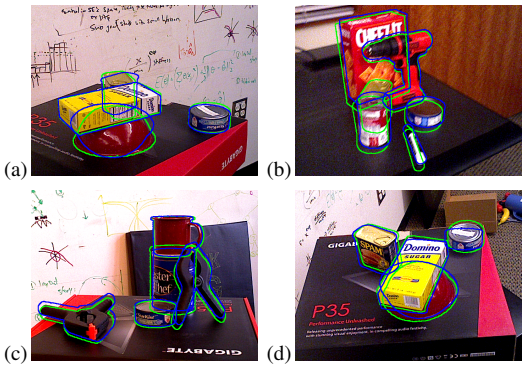

Fig. 9. Typical failure cases. Ground truth and predicted object poses are visualized as object contours in green and blue colors, respectively. Occlusion negatively influences the accuracy of our model.

failure cases of our model. *bowl*, for example, is often predicted upside down. Occlusion still remains a big challenge for our model. In Table III, we compare the accuracy of the different models we discussed in Section III.

In comparison to Model-B, Model-A performs a little worse. This demonstrates that despite the careful design employing shifted window attention, the model suffers from inefficiencies in global dependency modeling. Moreover, Zhou *et al.* [33] noted that the models based on local hierarchical shifting windows suffer from a lack of robustness. Although Model-C, based on the early fusion of object queries, performed better than Model-A, it did not match the performance of Model-B. However, in terms of the inference speed, Model-A performs better than other models—except for the baseline YOLOPose model. Model-B is the slowest. This is caused by accessing random memory locations in the deformable attention computation.

E. Ablation Study

Query Aggregation: In Fig. 7, we present the results of training our model with different query aggregation factors. The performance of the model increases with the query aggregation factor and reaches the highest accuracy for factor 3 but drops for factor 4.

Need for COCO Pre-training: Training the vision transformer models for set prediction is harder than training CNN

TABLE II
RESULTS ON THE YCB-VIDEO DATASET [6].

Metric	AUC of ADD-S					AUC of ADD(-S)				
Method	GDR-Net [8]	DeepIM [12]	YOLOPose [25]	YOLOPose V2 [26]	Ours	GDR-Net [8]	DeepIM [12]	YOLOPose [25]	YOLOPose V2 [26]	Ours
master_chef_can	**96.6**	93.1	91.3	91.7	90.3	71.1	71.2	64.0	**71.3**	66.7
cracker_box	84.9	91.0	86.8	92.0	**92.3**	63.5	83.6	77.9	83.3	**86.0**
sugar_box	**98.3**	96.2	92.6	91.5	94.4	93.2	**94.1**	87.3	83.6	89.1
tomato_soup_can	**96.1**	92.4	90.5	87.8	89.2	**88.9**	86.1	77.8	72.9	76.3
mustard_bottle	**99.5**	95.1	93.6	96.7	96.5	**93.8**	91.5	87.9	93.4	93.3
tuna_fish_can	95.1	**96.1**	94.3	94.9	94.5	85.1	**87.7**	74.4	70.5	67.4
pudding_box	94.8	90.7	92.3	92.6	**95.5**	86.5	82.7	87.9	87.0	**91.9**
gelatin_box	95.3	94.3	90.1	92.2	**95.4**	88.5	**91.9**	83.4	85.7	91.8
potted_meat_can	82.9	86.4	85.8	85.0	**88.9**	72.9	76.2	**76.7**	71.4	76.4
banana	**96.0**	91.3	90.0	95.8	95.4	85.2	81.2	88.2	90.0	**91.0**
pitcher_base	**98.8**	94.6	93.6	95.2	94.9	**94.3**	90.1	88.5	90.8	89.9
bleach_cleanser	**94.4**	90.3	85.3	83.1	87.3	80.5	**81.2**	73.0	70.8	73.9
bowl*	84.0	81.4	92.3	**93.4**	91.9	84.0	81.4	**92.3**	**93.4**	91.9
mug	**96.9**	91.3	84.9	95.5	95.5	87.6	81.4	69.6	**90.0**	89.3
power_drill	91.9	92.3	92.6	92.5	**94.6**	78.7	85.5	86.1	85.2	**88.9**
wood_block*	77.3	81.9	84.3	**93.0**	**93.0**	77.3	81.9	84.3	**93.0**	**93.0**
scissors	68.4	75.4	**93.3**	80.9	89.5	43.7	60.9	**87.0**	71.2	76.2
large_marker	**87.4**	86.2	84.9	85.2	84.5	76.2	75.6	76.6	77.0	**77.4**
large_clamp*	69.3	74.3	92.0	**94.7**	**94.2**	69.3	74.3	92.0	**94.7**	94.2
extra_large_clamp*	73.6	73.3	**88.9**	80.7	79.2	73.6	73.3	**88.9**	80.7	79.2
foam_brick*	90.4	81.9	90.7	93.8	**95.0**	90.4	81.9	90.7	93.8	**95.0**
Mean	89.1	88.1	90.1	91.2	**92.0**	80.2	81.9	82.6	83.3	**84.7**

The best results are shown in bold.

TABLE III
RESULTS FROM ABLATION STUDY.

Method	AUC of ADD-S	AUC of ADD(-S)	Params $\times 10^6$	fps
YOLOPose [25]	90.1	82.6	**48.6**	**41.8**
YOLOPose V2 [26]	91.2	83.3	53.2	39.1
CosyPose [13]	89.8	84.5	-	2.5
Model-A	90.1	81.4	57	39.5
Model-B 16 def pts	**92.0**	**84.7**	55.2	25.9
Model-B 6 def pts	81.9	89.3	51.7	28.1
Model-B 6 def pts + refine	90.2	83.0	87.1	25.6
Model-C	90.3	82.1	48.7	34.8

Model-A: Local hierarchical attention-based model discussed in Section III-A.
Model-B: Model based on multi-resolution deformable multi-head attention discussed in Section III-B.
Model-C: Model utilizing early fusion discussed in Section III-C.
def pts: Number of deformable points.

models to perform single object pose prediction due to the usage of bipartite matching to find the matching pairs between the predicted and the ground-truth sets, which results in slower convergence. Moreover, training data requirements are also much larger for the set predictions task, compared to single object prediction. We hypothesize that in the initial phase of the training, the model learns to detect multiple objects in the image and only in the later stages the model learns to predict keypoints and the pose parameters. Although the YCB-Video dataset [6] is considerably larger than the other pose annotation datasets, it is not big enough to train vision transformer models for multi-object pose estimation. To overcome this limitation, we train our model initially on the COCO dataset for the task of multi-object detection (class probability and bounding box prediction) and then train the model for multi-object pose estimation on the YCB-Video dataset. In Table I, we compare the pose estimation accuracy of models trained using only the YCB-Video dataset [6] and using COCO dataset for pretraining. The model pretrained using the COCO dataset outperforms the model trained using only the YCB-Video dataset, highlighting the importance of large-scale pretraining for training vision transformers to learn the task of multi-object prediction.

Prediction Refinement: In Table III, we present the results of the prediction refinement experiment. Refinement boosted the performance of Model-B constructed with six deformable points by 0.9 and 1.1 accuracy points in terms of the AUC of ADD-S and AUC of ADD-(S) metrics, respectively. However, the improvements come at a cost of an increased number of parameters: 87.1×10^6 compared to 51.7×10^6. Interestingly, for the Model-B constructed using 16 deformable points that achieves an impressive accuracy of 92 AUC of ADD-S and 84.7 AUC of ADD-(S), the boost in performance is negligible.

F. Limitations

While formulating multi-object pose estimation as a set prediction problem facilitates the prediction of a varying number of objects in the given image, training the model needs complete set annotations for all objects in the given image. Most of the commonly used pose estimation datasets like Linemod-Occluded [34] and Linemod [35] offer only partial annotations for training images. Thus, they are unsuitable for training our models. Acquiring complete pose annotations can be prohibitively expensive in real-world settings.

V. Conclusion

In this paper, we investigated various inductive biases in the design of multi-object pose estimation models, namely, local hierarchical shifting window attention, deformable multi-resolution attention, and early fusion of object queries. Moreover, we proposed a query aggregation mechanism to increase the number of object queries without increasing the computational complexity of our model. The best-performing model based on deformable multi-resolution attention achieves state-of-the-art results on the challenging YCB-Video dataset.

VI. 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".

References

[1] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, 255–258, 1995.

[2] S. Behnke, "Hierarchical neural networks for image interpretation," *LNCS*, vol. 2766, 2003.

[3] N. Cohen and A. Shashua, "Inductive bias of deep convolutional networks through pooling geometry," in *International Conference on Learning Representations (ICLR)*, 2017.

[4] H. Schulz and S. Behnke, "Deep learning: Layer-wise learning of feature hierarchies," *KI-Künstliche Intelligenz*, vol. 26, pp. 357–363, 2012.

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

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

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

[8] G. Wang, F. Manhardt, F. Tombari, and X. Ji, "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.

[9] M. Rad and V. Lepetit, "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)*, 2017, pp. 3828–3836.

[10] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "PVNet: Pixel-wise voting network for 6DOF pose estimation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[11] Y. Hu, P. Fua, W. Wang, and M. Salzmann, "Single-stage 6D object pose estimation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2930–2939.

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

[13] Y. Labbe, J. Carpentier, M. Aubry, and J. Sivic, "CosyPose: Consistent multi-view multi-object 6D pose estimation," in *European Conference on Computer Vision (ECCV)*, 2020.

[14] A. S. Periyasamy, M. Schwarz, and S. Behnke, "Refining 6D object pose predictions using abstract render-and-compare," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2019.

[15] C. Capellen, M. Schwarz, and S. Behnke, "ConvPoseCNN: Dense convolutional 6D object pose estimation," *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2020.

[16] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 2015.

[17] J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4507–4515.

[18] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7263–7271.

[19] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother, "DSAC-differentiable RANSAC for camera localization," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6684–6692.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Neural Information Processing Systems (NeurIPS)*, 2017.

[21] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *International Conference on Learning Representations (ICLR)*, 2021.

[22] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[23] J. Yang, C. Li, P. Zhang, X. Dai, B. Xiao, L. Yuan, and J. Gao, "Focal attention for long-range interactions in vision transformers," *Neural Information Processing Systems (NeurIPS)*, 2021.

[24] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable DETR: Deformable transformers for end-to-end object detection," in *International Conference on Learning Representations (ICLR)*, 2021.

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

[26] A. S. Periyasamy, A. Amini, V. Tsaturyan, and S. Behnke, "YOLO-Pose V2: Understanding and improving transformer-based 6D pose estimation," *Robotics and Autonomous Systems*, p. 104 490, 2023.

[27] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy, "Do vision transformers see like convolutional neural networks?" *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 12 116–12 128, 2021.

[28] H. Song, D. Sun, S. Chun, V. Jampani, D. Han, B. Heo, W. Kim, and M. Yang, "ViDT: An efficient and effective fully transformer-based object detector," in *10th International Conference on Learning Representations (ICLR)*, 2022.

[29] S. Zhang, X. Wang, J. Wang, J. Pang, C. Lyu, W. Zhang, P. Luo, and K. Chen, "Dense distinct query for end-to-end object detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 7329–7338.

[30] R. Stewart, M. Andriluka, and A. Y. Ng, "End-to-end people detection in crowded scenes," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2325–2333.

[31] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 658–666.

[32] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," *Neural Information Processing Systems (NeurIPS)*, 2021.

[33] D. Zhou, Z. Yu, E. Xie, C. Xiao, A. Anandkumar, J. Feng, and J. M. Alvarez, "Understanding the robustness in vision transformers," in *International Conference on Machine Learning (ICML)*, 2022, pp. 27 378–27 394.

[34] E. Brachmann, "6D object pose estimation using 3D object coordinates [data]," 2020. [Online]. Available: https://doi.org/10.11588/data/V4MUMX.

[35] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model-based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes," in *Asian Conference on Computer Vision (ACCV)*, Springer, 2013, pp. 548–562.