This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

CheckerPose: Progressive Dense Keypoint Localization for Object Pose Estimation with Graph Neural Network

Ruyi Lian Haibin Ling
Department of Computer Science, Stony Brook University, Stony Brook, NY 11794-2424, USA
{rulian,hling}@cs.stonybrook.edu

Abstract

Estimating the 6-DoF pose of a rigid object from a single RGB image is a crucial yet challenging task. Recent studies have shown the great potential of dense correspondence-based solutions, yet improvements are still needed to reach practical deployment. In this paper, we propose a novel pose estimation algorithm named CheckerPose*, which improves on three main aspects. Firstly, CheckerPose densely samples 3D keypoints from the surface of the 3D object and finds their 2D correspondences progressively in the 2D image. Compared to previous solutions that conduct dense sampling in the image space, our strategy enables the correspondence searching in a 2D grid (*i.e.*, pixel coordinate). Secondly, for our 3D-to-2D correspondence, we design a compact binary code representation for 2D image locations. This representation not only allows for progressive correspondence refinement but also converts the correspondence regression to a more efficient classification problem. Thirdly, we adopt a graph neural network to explicitly model the interactions among the sampled 3D keypoints, further boosting the reliability and accuracy of the correspondences. Together, these novel components make CheckerPose a strong pose estimation algorithm. When evaluated on the popular Linemod, Linemod-O, and YCB-V object pose estimation benchmarks, CheckerPose clearly boosts the accuracy of correspondence-based methods and achieves state-of-the-art performances. Code is available at* https://github.com/RuyiLian/CheckerPose.

1. Introduction

Object pose estimation from RGB images aims to estimate the rotation and translation of a given rigid object relative to the camera. It is crucial in various applications including robot grasping and manipulation [75, 61, 62], autonomous driving [36, 69, 29], augmented reality [37, 58], *etc*. Most existing methods [48, 59, 38, 18, 43, 72, 40, 33, 53] first estimate an intermediate geometric representation, *i.e.*, the correspondences between 3D object keypoints and 2D image locations, and then recover the object pose using the Perspective-n-Point (PnP) algorithm. Theoretically, for a rigid object, four pairs of 3D-2D correspondences can determine a unique pose [47, 10, 44]. In practice, however, sparse correspondences easily degrade due to occlusion, background clutter, lighting variation, *etc*.

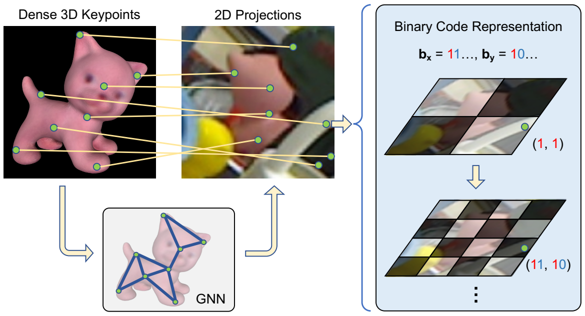

Dense 3D Keypoints
2D Projections
Binary Code Representation
$\mathbf{b}_x$ = 11..., $\mathbf{b}_y$ = 10...
(1, 1)
(11, 10)
⋮
GNN

Figure 1: **Illustration of CheckerPose.** We evenly sample dense keypoints from the object surface, and predict the 2D locations in the input image. We design a binary code representation to progressively localize each keypoint in the iteratively refined 2D grids. To improve the localization, we also use graph neural networks to explicitly model the interactions between 3D keypoints. Note: we plot 8 keypoints for better visualization, while use 512 keypoints in practice.

Increasing the number of 3D-2D correspondences is a feasible solution to enhance robustness, especially when combined with outlier removal mechanisms such as RANSAC. Recent methods [72, 40, 33, 15, 17, 66, 8] densely sample 2D image pixels and predict their 3D object coordinates. While these dense predictions improve the robustness of pose estimation, they have several drawbacks. Firstly, the predictions consider only visible pixels and ignore global relations between visible and occluded keypoints, making them unstable when the object is under severe occlusions. Secondly, estimating the corresponding

3D coordinates is nontrivial. Finally, the rich shape prior information is not effectively encoded.

To overcome the above issues, we propose a novel 6D pose estimation algorithm, named *CheckerPose*, which improves dense correspondence with three cooperative components: dense 3D sampling, progressive 2D localization through binary coding, and shape prior encoding with graph neural network, as illustrated in Figure 1.

For dense correspondence, CheckerPose samples 3D keypoints on the object surface and then finds their 2D pixel correspondences in the 3D-to-2D matching way. Compared to previous solutions that conduct dense sampling in the 2D image space, our strategy enables more efficient correspondence searching in a 2D grid (*i.e*., pixel coordinate) using 2D binary coding, as well as explicit shape prior modeling with graph representation.

Then, to facilitate the localization of dense keypoints, we propose a 2D hierarchical binary coding to represent a 2D image position. Specifically, we superpose a grid on the input image and predict which cells contain the desired keypoints. The precision of the 2D keypoint location is controlled by the resolution of the grid. This novel representation allows us to refine the correspondence progressively. We first localize the keypoints in the 2×2 grid, and then iteratively subdivide each cell and localize the keypoints in the refined grid. Inspired by ZebraPose [56], we use binary codes on the x and y directions to represent each cell, which makes the grids have a checkerboard pattern.

Furthermore, to capture the shape prior of the 3D object, we adopt a graph neural network to explicitly model the interactions among the sampled 3D keypoints and to guide the progressive correspondence estimation. In particular, we construct the k-nearest neighbor (k-NN) graph of the dense keypoints and utilize graph network layers to fuse information from a keypoint and its neighbors. By stacking multiple such layers, we can capture non-local interactions between invisible and visible keypoints, and thus significantly improve the prediction robustness of invisible keypoints.

To summarize, our main contributions are as follows:

- We propose to localize dense 3D keypoints in the input image, to establish dense correspondences for instance-level object pose estimation.
- We design a hierarchical binary coding strategy for 2D projections, which enables progressive localization of dense keypoints.
- We utilize graph neural networks to explicitly model the interactions between 3D keypoints and improve the predictions of invisible keypoints.

Together, these novel contributions make our CheckerPose a strong pose estimation algorithm. We conduct extensive experiments on the popular benchmarks including Linemod [14], Linemod-Occlusion [2], and YCB-V [70], and CheckerPose consistently achieves state-of-the-art performances.

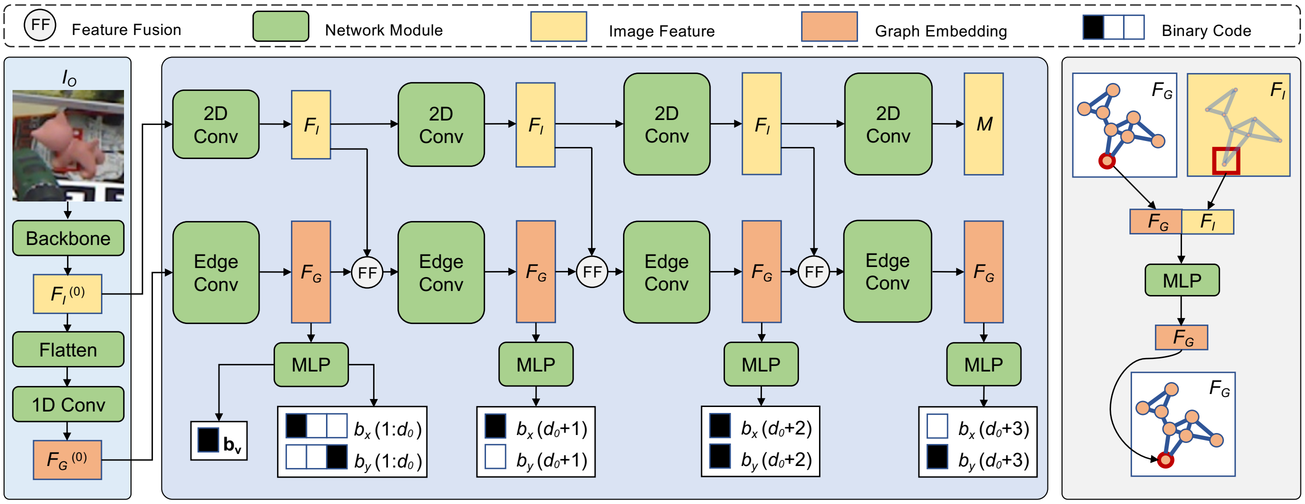
2. Related Work

In this section we review previous studies that are closely related to our work, mainly including different types of pose estimators and graph neural networks.

Direct Methods. Given an input RGB image, direct methods estimate the 6D pose of the object in the image without intermediate geometric representations, *e.g*., 3D-2D correspondences. Traditional direct methods mainly adopt template matching techniques with hand-crafted features [20, 11, 13], and thus can not handle textureless objects well. Recent deep learning based methods utilize features learned by CNNs to directly regress 6D pose [70] or formulate the rotation estimation as a classification task by discretizing the rotation space $SO(3)$ [63, 55, 23, 57].

Correspondence Guided Methods. Instead of direct estimation, correspondence guided methods [42, 48, 59, 38, 18, 43, 17, 19, 72, 40, 33, 66, 8, 56] follow a two-stage framework: they first predict a set of correspondences between 3D object frame coordinates and 2D image plane coordinates, and then recover the pose from the 3D-2D correspondences with a PnP algorithm [27, 25, 9, 64, 4]. RANSAC can be used to remove the outliers in the correspondences. Keypoint-localization based methods [42, 48, 59, 38, 18, 43, 17, 19] estimate the 2D coordinates for a sparse set of predefined 3D keypoints, while dense methods [72, 40, 33, 66, 8, 56] predict the 3D object frame coordinate of each 2D image pixel. Compared with sparse correspondences, dense correspondences contain richer context information of the scene and is more robust to occlusion.

Graph Neural Networks for 3D Vision. In 3D vision tasks, point clouds and meshes are important input data formats since they can efficiently represent complex shapes. Compared with convolutional neural networks (CNNs), graph neural networks (GNNs) [54] can handle inputs with irregular structures and effectively model the long-range dependencies, and thus are widely used for processing point clouds and meshes. While meshes can be naturally treated as graphs, a common practice of constructing graphs from point clouds is to treat each 3D point as graph nodes and connect each node to its k nearest neighbors [68, 52, 6]. GNN-based methods have been proposed for representation learning [52, 68, 35, 65], detection [51, 6], segmentation [45, 28], data generation [46, 34], camera pose inference [30, 31], *etc*. Graph techniques have also been used for learning dense correspondences between 3D shapes [50] using both local and global information. For object pose estimation, GNNs are mainly used for RGB-D inputs [7, 74] to enhance the feature extraction from different modalities. Another recent application is to learn geometric structures of the sparse keypoints for domain adaptation [73].



(a) Initialization (b) Progressive Prediction (c) Feature Fusion

Figure 2: **Framework of our progressive dense keypoint localization with graph neural network, *i.e.*, CheckerPose.** Given an RGB image and object detection results, we progressively generate the binary codes representing the 2D locations of N 3D keypoints. **(a)** Initial graph embedding generation: we use a CNN backbone network to extract feature $F_I^{(0)}$ from the zoomed-in RoI I_O, and then transform $F_I^{(0)}$ to the initial keypoint embeddings $F_G^{(0)}$ in the k-NN graph $\mathcal{G}$. **(b)** Progressive prediction: we use a graph neural network to generate the binary code representation in a coarse-to-fine manner. We adopt an additional CNN decoder network to generate image features with increased resolutions from $F_I^{(0)}$, and fuse the features in the graph neural network based on current predictions. Object segmentation masks M are predicted as an auxiliary learning task. **(c)** Feature fusion: to fuse the image feature F_I into the graph embeddings F_G, for each keypoint, we crop a feature patch from F_I based on the current localization result, and concatenate the flattened feature with keypoint embedding. We then use a shared MLP to fuse the concatenation and the result is the updated keypoint embedding.

Our work follows the two-stage framework and combines the strengths of both keypoint-based methods and dense methods, by localizing a dense set of predefined 3D keypoints to establish dense correspondences. Moreover, it utilizes GNNs to efficiently model the interactions among dense 3D keypoints and thus improve the localization in the input RGB image for monocular object pose estimation.

3. Method

3.1. Problem Formulation and Method Overview

Given an RGB image I and a rigid object O, our goal is to estimate rotation $\mathbf{R} \in SO(3)$ and translation $\mathbf{t} \in \mathbb{R}^3$ of O relative to the calibrated camera. We assume the 3D geometry information, *e.g.*, the 3D CAD model, is available, thus we can obtain $N(N \gg 8)$ keypoints $\mathcal{P} \subset \mathbb{R}^3$ from the object surface using farthest point sampling (FPS).

We adopt a two-stage pipeline for object pose estimation: we first predict 2D projection $\boldsymbol{\rho} \in \mathbb{R}^2$ for each keypoint $P \in \mathcal{P}$, and then regress the rotation and translation from the 3D-2D correspondences via a PnP solver. For the input RGB image, we use an off-the-shelf object detector [49, 60] to detect the object bounding box and extract the zoomed-in Region of Interest (RoI) I_O, following the common practice in instance-level object pose estimation [33, 66, 8, 56]. Figure 2 illustrates our proposed pipeline. We first process the input RoI I_O by a backbone network to obtain backbone feature $F_I^{(0)}$ and keypoint embedding $F_G^{(0)}$ in the k-NN graph $\mathcal{G}$. Then we use graph network layers (*i.e.*, EdgeConv [68]) to progressively localize the keypoints, which are represented as binary codes $\mathbf{b_v}, \mathbf{b_x}$, and $\mathbf{b_y}$. We also use a standard CNN decoder to transform $F_I^{(0)}$ to a series of image feature maps, and fuse the features in the graph neural network based on the current predicted locations. The CNN decoder also outputs object segmentation masks M as an auxiliary learning task. Finally, we convert the binary codes to 2D coordinates and use a PnP solver to recover the pose from the established correspondences. We describe our method, named *CheckerPose* due to the checkerboard-like binary pattern, in details as follows.

3.2. Hierarchical Representation of 2D Keypoints

Establishing 3D-2D correspondences provides an intermediate representation for object pose estimation. In this work, we focus on localizing a dense set of predefined 3D keypoints $\mathcal{P}$ in the 2D image plane. For $N(N \gg 8)$ 3D keypoints $\mathcal{P}$, we first predict whether their 2D projections

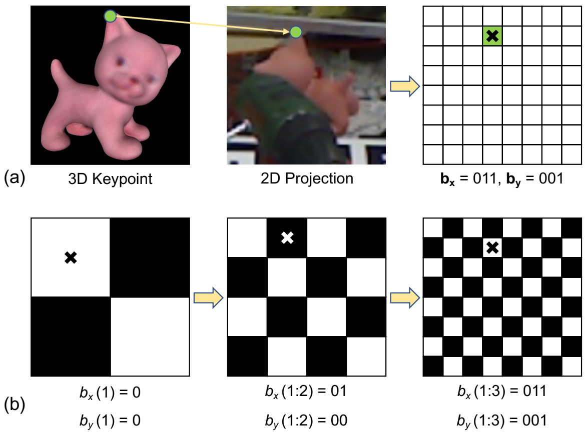


Figure 3: **Keypoint location representation.** **(a)** We represent the 2D projection coordinate as the center of the cell containing the 2D projection. **(b)** We iteratively refine the grid and represent the cell as binary codes $\mathbf{b_x}, \mathbf{b_y}$.

appear in the RoI I_O, and then localize the keypoints inside I_O, denoted as $\mathcal{P}_I$. In contrast to directly regressing the precise coordinates, we superpose a $2^d \times 2^d$ grid S on the RoI I_O and predict which cell $s \in S$ contains the 2D projection $\boldsymbol{\rho}$ (Figure 3 (a)). Then we can use the coordinate of the cell center to approximate $\boldsymbol{\rho}$, and only need to predict the discrete index $(i_x, i_y)(0 \leq i_x, i_y \leq 2^d - 1)$ of the cell s, which is much easier than precise regression. The localization precision is controlled by the resolution of the grid S, and approaches the actual 2D projection as $d \to \infty$.

Based on the approximate representation, we can further localize the keypoint $P \in \mathcal{P}_I$ in a coarse-to-fine manner. As shown in Figure 3 (b), at the beginning, we superpose a 2×2 grid $S^{(1)}$ on the RoI I_O and predict the index of the cell $s_P^{(1)}$. Then at iteration j $(2 \leq j \leq d)$, we increase the grid resolution from $2^{j-1} \times 2^{j-1}$ to $2^j \times 2^j$ by evenly splitting each cell $s^{(j-1)} \in S^{(j-1)}$ into halves on both x and y directions. With the prediction of $s_P^{(j-1)}$ in iteration $j-1$, we only need to search the corresponding 2×2 sub-cells to find $s_P^{(j)}$ in the refined grid $S^{(j)}$.

Inspired by ZebraPose [56], we use binary codes to concisely represent the hierarchical localization. For the cell s_P in the final $2^d \times 2^d$ grid S, we use a d-bit binary code $\mathbf{b_x}$ to represent the index i_x as

$$i_x = \sum_{k=1}^{d} b_x(k) \times 2^{d-k}, \tag{1}$$

where $b_x(k)$ is the k-th bit of $\mathbf{b_x}$. We use another d-bit binary code $\mathbf{b_y}$ to represent the index i_y in the same way. The first $j(1 \leq j \leq d)$ bits of $\mathbf{b_x}$ and $\mathbf{b_y}$ also represent the cell $s_P^{(j)} \in S^{(j)}$. We use an additional 1-bit binary code $\mathbf{b_v}$ to indicate the existence of the projection $\boldsymbol{\rho}$ in the RoI I_O, where $\mathbf{b_v} = 1$ means $\boldsymbol{\rho} \in I_O$ while $\mathbf{b_v} = 0$ means $\boldsymbol{\rho} \notin I_O$.

Compared with dense representations (*e.g.*, heatmaps [42, 38] and vector-fields [43, 18]), our representation needs only $2d+1$ binary bits for each keypoint, thus greatly reduces the memory usage for dense keypoint localization. In addition, during inference, we can efficiently convert the binary codes to the 2D coordinates. Furthermore, our representation can be naturally predicted in a progressive way, which allows to gradually improve the localization via iterative refinements.

3.3. Dense Keypoint Localization via Graph Neural Network

Modeling the interactions among the keypoints $\mathcal{P}$ is crucial for predicting their 2D locations. For the keypoints that are invisible due to occlusions or self-occlusions, the features of the visible ones provide additional clues to infer the 2D locations. However, previous keypoint-based methods mainly use convolutional neural networks (CNNs), which can not handle inputs with irregular structure and thus fail to explicitly capture the interactions among $\mathcal{P}$.

We instead utilize graph neural networks (GNNs) to process the features $F = \{f_1, \cdots, f_N\}$ of N keypoints $\mathcal{P}$. To construct a graph $\mathcal{G}$ from $\mathcal{P}$, we treat each keypoint $P_i \in \mathcal{P}(1 \leq i \leq N)$ as a graph node, and connect P_i to its k nearest neighbors in 3D Euclidean space to generate edges $\mathcal{E}$. We adopt the EdgeConv operation [68] as our graph network layer, which directly models local interactions between P_i and its neighbors. For edge $(i, j) \in \mathcal{E}$, we compute the feature e_{ij} as

$$e_{ijm} = \mathrm{ReLU}(\theta_m \cdot (f_j - f_i) + \phi_m \cdot f_i), \tag{2}$$

where e_{ijm} is the m-th channel of e_{ij}, and θ_m, ϕ_m are the weights of the filters. The feature of P_i is updated by aggregating the edge features as

$$f'_{im} = \max_{j:(i,j) \in \mathcal{E}} e_{ijm}, \tag{3}$$

where f'_{im} is the m-th channel of updated feature f'_i. By stacking multiple EdgeConv operations, our network can gradually learn the non-local interactions in a computationally efficient way for dense keypoints $\mathcal{P}$.

As shown in Figure 2 (a), to obtain the initial keypoint embeddings $F_G^{(0)}$ in $\mathcal{G}$, we first use a backbone network to extract a $C_0 \times 2^{d_0} \times 2^{d_0}$ feature map $F_I^{(0)}$ from RoI I_O, where C_0 is the number of the feature channels, and $2^{d_0} \times 2^{d_0}$ is the spatial size. We then reshape $F_I^{(0)}$ to $C_0 \times 2^{2d_0}$ by flattening the spatial dimensions, and use a 1D convolutional network layer to obtain a $N \times 2^{2d_0}$ feature map, which is regarded as the initial 2^{2d_0}-dimensional embeddings $F_G^{(0)}$ for N keypoints.

After obtaining $F_G^{(0)}$, we use a graph neural network to predict the 1-bit indicator code $\mathbf{b_v}$, and progressively generate the d-bit index codes $\mathbf{b_x}, \mathbf{b_y}$. Specifically, at stage 0, we apply L_0 EdgeConv [68] operations to $F_G^{(0)}$ to get the updated embeddings $F_G^{(1)}$, and then use shared MLPs to generate $\mathbf{b_v}$ and the first d_0 bits of $\mathbf{b_x}, \mathbf{b_y}$, respectively. Then at stage $j (1 \leq j \leq d - d_0)$, we apply L_j EdgeConv operations to $F_G^{(j)}$ to obtain $F_G^{(j+1)}$, and use shared MLPs to generate new bits $b_x(d_0 + j), b_y(d_0 + j)$ for $\mathbf{b_x}, \mathbf{b_y}$, respectively. We regard stage $j(1 \leq j \leq d - d_0)$ as refinement stage, since it refines the localization from the low-resolution grid $S^{(d_0+j-1)}$ to the high-resolution one $S^{(d_0+j)}$.

Compared with generating all bits at the network output layer, our progressive prediction enables image feature fusion at each refinement stage. As shown in Figure 2 (b), starting with the image feature map $F_I^{(0)}$ with low spatial resolution $2^{d_0} \times 2^{d_0}$, we use an additional CNN-based decoder to progressively generate image feature maps $F_I^{(1)}, \cdots, F_I^{(d-d_0)}$ with increased spatial resolutions $2^{d_0+1} \times 2^{d_0+1}, \cdots, 2^d \times 2^d$, respectively. We also add skip connections between the backbone and the decoder to recover the high-resolution details lost in $F_I^{(0)}$. As shown in Figure 2 (c), at the beginning of the refinement stage j, for each keypoint P, we select local image feature from $F_I^{(j)}$ based on the localization result in the previous stage. We then concatenate $F_l^{(j)}$ with the keypoint embedding in the graph $\mathcal{G}$, and use a shared MLP to fuse the concatenation. The fused feature is used as the updated keypoint embedding. Since the initial keypoint embeddings $F_G^{(0)}$ are obtained from $F_I^{(0)}$, fusing the local image features in the refinement stages provides critical high-resolution details for fine-grained localization.

3.4. Training

For the 1-bit indicator code $\mathbf{b_v}$ of keypoint $P \in \mathcal{P}$, our network output $\hat{\mathbf{b}}_\mathbf{v}$ is the probability that $\mathbf{b_v} = 1$. We use binary cross-entropy loss for $\mathbf{b_v}$ as below:

$$\mathcal{L}_v = \frac{1}{N} \sum_{P \in \mathcal{P}} \mathbf{b_v} \log \hat{\mathbf{b}}_\mathbf{v} + (1 - \mathbf{b_v}) \log(1 - \hat{\mathbf{b}}_\mathbf{v}), \quad (4)$$

where N is the number of the keypoints. For d-bit index codes $\mathbf{b_x}, \mathbf{b_y}$, since we only localize the keypoints inside the RoI (*i.e.*, $\mathbf{b_v} = 1$), denoted as $\mathcal{P}_I$, we compute binary cross-entropy loss for each bit of $\mathbf{b_x}$ as

$$\mathcal{L}_x = \frac{1}{dN_I} \sum_{P \in \mathcal{P}_I} \sum_{k=1}^{d} b_x(k) \log(\hat{b}_x(k)) + (1 - b_x(k)) \log(1 - \hat{b}_x(k)), \quad (5)$$

where N_I is the number of keypoints inside the RoI, $\hat{b}_x(k)$ is the network prediction for k-th bit of $\mathbf{b_x}$. We compute the loss $\mathcal{L}_y$ for $\mathbf{b_y}$ in the same way as $\mathcal{L}_x$.

Besides predicting the 2D projections as binary codes, we also enforce the network to output object segmentation masks. To do this, we apply a single CNN layer to the final image feature map $F_I^{(d-d_0)}$ and obtain a $2 \times 2^d \times 2^d$ output, which serves as the full segmentation mask M_{full} and the visible one M_{vis}. We input the network predictions to the sigmoid function and apply L_1 loss as the mask loss $\mathcal{L}_{\text{mask}}$. Generating the masks can be regarded as an auxiliary task to facilitate the learning of image features.

The overall loss function $\mathcal{L}$ is a combination of $\mathcal{L}_v$, $\mathcal{L}_x$, $\mathcal{L}_y$, and $\mathcal{L}_{\text{mask}}$ as

$$\mathcal{L} = \mathcal{L}_v + \mathcal{L}_x + \mathcal{L}_y + \mathcal{L}_{\text{mask}}. \quad (6)$$

Before training the whole network, we pretrain the layers that generate $\mathbf{b_v}$ and the first d_0 bits of $\mathbf{b_x}, \mathbf{b_y}$. This encourages the backbone network to quickly adapt to the object keypoints with smaller GPU memory usage, and makes the initial localization to be good for local image feature fusion in the refinement stages.

3.5. Inference

During inference, we first discard the keypoints with $\mathbf{b_v} = 0$. Then we convert the binary codes to the corresponding cells in the final grid S (Eq. 1), and use the 2D coordinates of the cell centers as the keypoint projections. In this way, we establish dense 3D-2D correspondences from the network outputs without time-consuming computation operations, *e.g.*, voting for the vector-field representations [43]. Finally we use the RANSAC/PnP [27] or Progressive-X [1] solvers to obtain the object pose from the dense 3D-2D correspondences.

We empirically find that for textureless objects with severe self-occlusions, discarding the correspondences outside M_{vis} can improve the pose estimation results. To quantify the self-occlusions of a given object O, we uniformly sample 2,562 camera viewpoints on a sphere, and use the Hidden Point Removal (HPR) operator [22] to estimate the visibility of point $P \in O$ from each viewpoint. We then calculate the proportion of the viewpoints for which P is visible, denoted as $V(P)$. If $0.2 \leq V(P) < 0.4$, then P is considered to be easily self-occluded. Note we ignore the points with $V(P) < 0.2$, to make our estimation robust to the classification error of the HPR operator. The overall self-occlusion of the object O can be computed by

$$r_{\text{so}}(O) = \frac{1}{|O|} \sum_{P \in O} \mathbb{1}(0.2 \leq V(P) < 0.4), \quad (7)$$

where $|O|$ is the number of vertices of the object CAD model, and $\mathbb{1}(\cdot)$ is the indicator function. If $r_{\text{so}}(O) \geq 0.5$, *i.e.*, over half part of O is easily to be self-occluded, then we regard O as severely self-occluded.

4. Experiments

4.1. Experimental Setup

Implementation Details. Our method is implemented using PyTorch [41] and trained using the Adam optimizer [24] with a batch size of 32. We pretrain our network for $50,000$ steps with learning rate of 2e-4. We use $N = 512$ keypoints, and utilize $k = 20$ nearest neighbors to construct the k-NN graph $\mathcal{G}$. For the binary code representation, we set $d = 6$ and $d_0 = 3$. We resize the input RoIs to 256×256, and use HRNet [67] as our image feature backbone to extract $1024 \times 8 \times 8$ feature map $F_I^{(0)}$. Then we apply $L_0 = 2$ EdgeConv operations to get $\mathbf{b_v}$ and the first $d_0 = 3$ bits of $\mathbf{b_x}, \mathbf{b_y}$, and obtain the full binary codes after 3 refinement stages with $L_j = 3$ $(j = 1, 2, 3)$ EdgeConv operations.

Datasets. We conduct our experiments on three commonly-used datasets for object pose estimation: Linemod (LM) [14], Linemod-Occlusion (LM-O) [2], and YCB-V [70]. LM consists of 13 sequences of real images with ground truth poses for a single object with background clutter and mild occlusion. Each sequence contains around $1,200$ images. Following [3], we utilize about 15% images for training while keeping the rest for testing. We additionally use $1,000$ synthetic RGB images for each object during training following [33, 66, 8]. LM-O consists of $1,214$ images from a sequence of LM [14], where ground truth poses of eight objects with partial occlusion are annotated for testing. YCB-V is composed of more than $110,000$ real images of 21 objects with severe occlusion and clutter. Apart from the real training images, we also utilize the physically-based rendered data following [16] for training on LM-O and YCB-V.

Evaluation Metrics. We employ the common evaluation metric ADD(-S) for object pose estimation. ADD(-S) measures whether the average distance between the model points transformed by the predicted pose and the ground truth is less than 10% of the object's diameter (0.1d). For symmetric objects, ADD(-S) metric computes the deviation to the closest model point. On YCB-V, we also compute the AUC (area under curve) of ADD-S and ADD(-S) with a maximum threshold of 10 cm [70]. On LM, we also report the $n°, n$ cm metric, measuring the percentage of predicted 6D poses with rotation error below $n°$ and translation error below n cm. For symmetric objects $n°, n$ cm computes the smallest error for all possible ground truth poses [32, 66].

4.2. Ablation Study on LINEMOD Dataset

We present ablation experiments on LM [14] in Table 1 to verify the effectiveness of each module. We also study the number of keypoint N and the size of neighborhood k in Supplementary. We train a single pose estimator for all objects for 120k steps, with a fixed learning rate of 1e-4 for the first 100k steps and a smaller learning rate of 5e-5 for the remaining steps. During inference, we utilize the detection results from Faster-RCNN [49] by [33]. We do not use any segmentation masks to filter the correspondences for fair comparison. Without specification, we use Progressive-X [1] to compute pose from the dense correspondences.

Method	ADD(-S)			2°2cm	5°5cm
	0.02d	0.05d	0.1d		
GDR-Net [66]	35.5	76.3	93.7	62.1	N/A
SO-Pose [8]	**45.9**	83.1	96.0	76.9	98.5
EPro-PnP [5]	44.8	82.0	95.8	**81.0**	98.5
Ours (w/o GNN)	26.4	77.8	95.2	67.7	97.9
Ours (w/o Prog.)	14.1	56.9	85.8	42.3	94.1
Ours (w/o M_{full})	30.2	82.8	96.7	79.3	**98.9**
Ours (w/o M_{vis})	34.1	82.8	96.6	79.1	**98.9**
Ours (ResNet34)	31.3	80.2	95.6	74.2	98.6
Ours (RANSAC/PnP)	31.1	81.4	96.6	78.4	**98.9**
CheckerPose (Ours)	35.7	**84.5**	**97.1**	79.7	**98.9**

Table 1: **Ablation Study on the LM Dataset.**

Comparison with State of the Art. As shown in Table 1, our method outperforms the state-of-the-art methods [66, 8, 5] w.r.t. ADD(-S) 0.05d, ADD(-S) 0.1d, and 5°5cm, and achieves comparable results w.r.t. ADD(-S) 0.02d and 2°2cm. The improvement of ADD(-S) 0.1d indicates that our method can facilitate the estimation of hard cases and serve as a good initialization for refinement methods [32, 21, 71]. Since the 2D coordinates of our estimated correspondences are approximated by the cell centers (Sec. 3.2), our pose estimation results in terms of ADD(-S) 0.02d may be further improved by increasing the grid resolution.

Effectiveness of Graph Neural Networks. Our network utilizes GNN layers, *e.g.*, EdgeConv [68], to explicitly model the interactions between different keypoints. We also report the result of removing all GNN layers in Table 1. Without GNN layers, the keypoints still interact indirectly via local image feature fusion modules, since the keypoints with close 2D locations share the similar local image features. However, the performance of pose estimation degrades significantly, demonstrating that it is important to directly model the keypoint interactions with GNN layers.

Effectiveness of Progressive Prediction. Progressively generating the binary codes enforces our network to gradually refine the localization in the iteratively subdivided grids. It also enables image feature fusion based on the intermediate estimations, which can provide crucial high-resolution details for fine-grained localization. As shown in Table 1, the accuracy decreases significantly without

Method	PVNet [43]	S. Stage [17]	Hybrid [53]	RePose [21]	GDR-Net [66]	SO-Pose [8]	Zebra [56]	Ours
ape	15.8	19.2	20.9	31.1	46.8	48.4	57.9	58.3
can	63.3	65.1	75.3	80.0	90.8	85.8	95.0	95.7
cat	16.7	18.9	24.9	25.6	40.5	32.7	60.6	62.3
driller	65.7	69.0	70.2	73.1	82.6	77.4	94.8	93.7
duck	25.2	25.3	27.9	43.0	46.9	48.9	64.5	69.9
eggbox*	50.2	52.0	52.4	51.7	54.2	52.4	70.9	70.0
glue*	49.6	51.4	53.8	54.3	75.8	78.3	88.7	86.4
holep.	36.1	45.6	54.2	53.6	60.1	75.3	83.0	83.8
mean	40.8	43.3	47.5	51.6	62.2	62.3	76.9	77.5

Table 2: **Comparison with State-of-the-art Methods on the LM-O Dataset.** We report the Average Recall (%) of ADD(-S). (*) denotes symmetric objects. We highlight the best result in red color, and the second best result in blue color.

progressively generating the binary codes, which clearly demonstrates the importance of progressive prediction.

Effectiveness of Object Segmentation Masks. Our network outputs the full segmentation mask M_{full} and the visible one M_{vis} as auxiliary tasks. As shown in Table 1, the performance degrades without either M_{full} or M_{vis}. The ADD(-S) 0.02d metric drops significantly without M_{full}, indicating that predicting M_{full} facilitates image feature extraction for keypoint localization, since all the keypoints should be located within M_{full}. The degraded performance without M_{vis} also implies that predicting M_{vis} provides important context information including occlusions.

Impact of Backbone Networks. We report the results of our method with different backbone networks in Table 1. After replacing HRNet [67] by ResNet34 [12], our method still achieves comparable results with state of the art, which demonstrates the efficacy of our method regardless of the backbone networks.

Influence of PnP Solvers. We show the results with different PnP solvers during inference in Table 1. Since our correspondences are established from the binary codes, a small perturbation of our network prediction can result in flipped bit values, which may correspond to dramatically different locations in the input RoI. Compared with RANSAC/PnP [27], Progressive-X [1] contains a spatial coherence filter to efficiently remove such outliers, and thus achieves better performance w.r.t. to all the metrics, especially ADD(-S) 0.02d.

4.3. Comparison to State of the Art

In this section we present the quantitative results of our method on LM-O and YCB-V datasets. We train a single CheckerPose for each object for 380,000 steps with a fixed learning rate of 1e-4. During inference, we utilize the detections from FCOS [60] provided by CDPNv2 [33].

Experiments on the LM-O dataset. We report the recall of ADD(-S) metric for the LM-O dataset in Table 2. Based on the criterion discussed in Sec. 3.5, we filter out the correspondences outside the visible segmentation masks M_{vis} for textureless objects with severe self-occlusions, including can, cat, driller, and eggbox. Without the filtering operation, the average recall of ADD(-S) of our method is 77.1, which surpasses previous methods. The detailed results of each object without filtering are provided in supplementary material. The additional filtering operation further improves the performance of our method. The intuition is that it is infrequent to observe an easily self-occluded keypoint P in the training images. Besides, due to the lack of texture, it is also hard to infer the location of P from other keypoints with distinguishable features. Such objects may require much more training steps to achieve stable estimations for easily self-occluded keypoints. Simply discarding correspondences outside M_{vis} reduces unstable localization results when our network is trained for limited steps, and enhances the robustness of pose estimation.

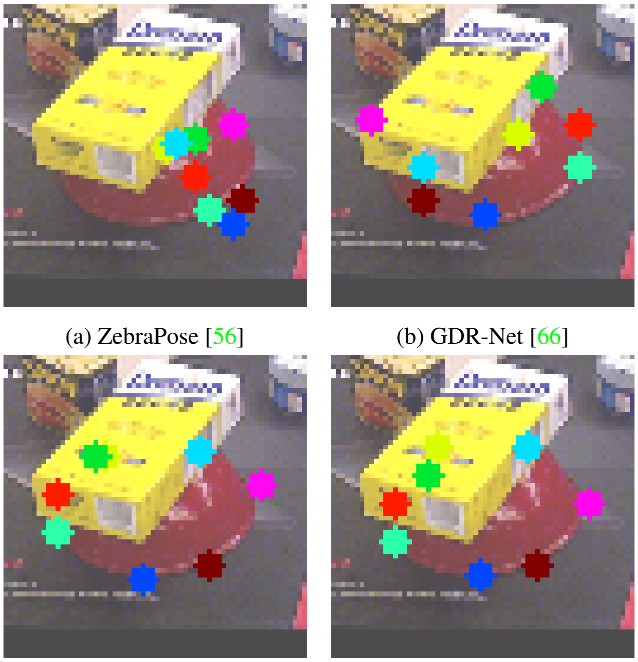
Experiments on the YCB-Video dataset. We report the averaged metrics of 21 objects in Table 3, and provide detailed results in the suppl.. Based on the criterion discussed in Sec. 3.5, we use visible segmentation masks to filter correspondences for foam_brick. We also apply the filtering operation to pudding_box because it is severely occluded by gelatin_box, which is a distraction object with similar texture. As shown in Table 3, CheckerPose achieves the best performance w.r.t. ADD(-S) and AUC of ADD(-S), and is comparable with state of the art w.r.t. AUC of ADD-S.

4.4. Qualitative Results

In Figure 4, we provide localization results of eight keypoints for the occluded and flipped bowl. While our network directly outputs the 2D locations, the results of other dense methods [56, 66] are computed by projecting the keypoints using the estimated poses. Figure 4 (a) visualizes the reprojections of ZebraPose [56], where the keypoints

Method	ADD(-S)	AUC ADD-S	AUC ADD(-S)
SegDriven [18]	39.0	–	–
SingleStage [17]	53.9	–	–
CosyPose [26]	–	89.8	84.5
RePose [21]	62.1	88.5	82.0
GDR-Net [66]	60.1	91.6	84.4
SO-Pose [8]	56.8	90.9	83.9
ZebraPose [56]	80.5	90.1	85.3
DProST [39]	65.1	–	77.4
CheckerPose (Ours)	81.4	91.3	86.4

Table 3: **Comparison on the YCB-Video Dataset.** We report the ADD(-S), and AUC of ADD-S and ADD(-S). Following [70], the symmetric metric is used for all objects in ADD-S while only for symmetric objects in ADD(-S). We highlight the best result in red color, and the second best result in blue color. "–" denotes unavailable results.



(a) ZebraPose [56]

(b) GDR-Net [66]

(c) CheckerPose (Ours)

(d) Ground Truth

Figure 4: **Keypoint localization.** **(a)** Keypoint locations based on the predicted pose of ZebraPose [56]. **(b)** Keypoint locations based on the pose estimated by GDR-Net [66]. **(c)** Keypoint locations output by our network. **(d)** The ground truth keypoint locations. Considering the symmetry of the bowl, we use the equivalent rotations closest to our prediction to project the keypoints in (a), (b), and (d).

concentrate on the visible pixels. Since ZebraPose generates pixel-wise 3D coordinates from the visible regions, it predicts a drastically wrong pose for the severely occluded bowl. As shown in Figure 4 (b), the reprojections of GDR-Net [66] cover the region similar to the ground truth (Figure 4 (d)). However, the order of the blue keypoint and the red one changes from clockwise to counterclockwise, indicating the bowl is actually faced up. Since GDR-Net is an end-to-end method, it may memorize poses that frequently appear in the training samples. As shown in Figure 4 (c), our network is capable of localizing the keypoints for the upside-down object with severe occlusion. More qualitative results can be found in the Supplementary Material.

4.5. Runtime Analysis

We test the running speed on the LM-O dataset. Given a 640 × 480 RGB image, we evaluate the speed on a desktop with an Intel 3.30GHz CPU and an NVIDIA GeForce GTX 1080 GPU (8G), which is reasonable in real-world application. The FCOS detector [60] takes 87 ms for each image. The runtime of establishing the dense 3D-2D correspondences by our network is 78 ms. RANSAC/PnP [27] takes only 1 ms to recover pose from the correspondences, while Progressive-X [1] takes 32 ms. Under the same testing environment, ZebraPose [56] requires 10ms for generating 3D-2D correspondences by CNN and around 350ms to estimate pose using Progressive-X. The overall running time of our method is greatly reduced, because we establish at most 512 candidate 3D-2D correspondences while ZebraPose outputs 128^2 candidates in the worst case.

5. Conclusion

In this work, we propose a novel way to establish dense correspondences for object pose estimation, by progressively localizing dense 3D keypoints in the input image. With dense keypoints including occluded and self-occluded ones, we comprehensively explore the available geometry information and enhance the robustness of pose estimation under severe occlusion. We adopt graph neural networks to explicitly model the keypoint interactions, and design a hierarchical binary code representation for the 2D locations. The experiments on LM, LM-O and YCB-V datasets demonstrate that our method achieves state-of-the-art performance of instance-level object pose estimation.

Acknowledgement. We thank all reviewers for valuable comments and suggestions. The work was supported in part by US National Science Foundation Grants 2006665 and 2128350, and by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR0011-22-9-0077. This work is also supported in part by the SBU/BNL Seed Grant Award. Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of funding agencies.

References

[1] Daniel Barath and Jiri Matas. Progressive-x: Efficient, anytime, multi-model fitting algorithm. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3780–3788, 2019. 5, 6, 7, 8

[2] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 536–551. Springer, 2014. 2, 6

[3] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, et al. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3364–3372, 2016. 6

[4] Bo Chen, Alvaro Parra, Jiewei Cao, Nan Li, and Tat-Jun Chin. End-to-end learnable geometric vision by backpropagating pnp optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8100–8109, 2020. 2

[5] Hansheng Chen, Pichao Wang, Fan Wang, Wei Tian, Lu Xiong, and Hao Li. EPro-PnP: generalized end-to-end probabilistic perspective-n-points for monocular object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2781–2790, 2022. 6

[6] Jintai Chen, Biwen Lei, Qingyu Song, Haochao Ying, Danny Z Chen, and Jian Wu. A hierarchical graph network for 3d object detection on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 392–401, 2020. 2

[7] Wei Chen, Xi Jia, Hyung Jin Chang, Jinming Duan, Linlin Shen, and Ales Leonardis. FS-Net: fast shape-based network for category-level 6d object pose estimation with decoupled rotation mechanism. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1581–1590, 2021. 2

[8] Yan Di, Fabian Manhardt, Gu Wang, Xiangyang Ji, Nassir Navab, and Federico Tombari. SO-Pose: exploiting self-occlusion for direct 6d pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12396–12405, 2021. 1, 2, 3, 6, 7, 8

[9] Luis Ferraz, Xavier Binefa, and Francesc Moreno-Noguer. Very fast solution to the pnp problem with algebraic outlier rejection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 501–508, 2014. 2

[10] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 25(8):930–943, 2003. 1

[11] Chunhui Gu and Xiaofeng Ren. Discriminative mixture-of-templates for viewpoint classification. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 408–421. Springer, 2010. 2

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 7

[13] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE transactions on pattern analysis and machine intelligence*, 34(5):876–888, 2011. 2

[14] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian Conference on Computer Vision (ACCV)*, pages 548–562. Springer, 2012. 2, 6

[15] Tomas Hodan, Daniel Barath, and Jiri Matas. EPOS: estimating 6d pose of objects with symmetries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11703–11712, 2020. 1

[16] Tomáš Hodaň, Martin Sundermeyer, Bertram Drost, Yann Labbé, Eric Brachmann, Frank Michel, Carsten Rother, and Jiří Matas. BOP challenge 2020 on 6D object localization. *European Conference on Computer Vision Workshops (ECCVW)*, 2020. 6

[17] Yinlin Hu, Pascal Fua, Wei Wang, and Mathieu Salzmann. Single-stage 6d object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2930–2939, 2020. 1, 2, 7, 8

[18] Yinlin Hu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. Segmentation-driven 6d object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3385–3394, 2019. 1, 2, 4, 8

[19] Yinlin Hu, Sebastien Speierer, Wenzel Jakob, Pascal Fua, and Mathieu Salzmann. Wide-depth-range 6d object pose estimation in space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15870–15879, 2021. 2

[20] Daniel P Huttenlocher, Gregory A. Klanderman, and William J Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, 15(9):850–863, 1993. 2

[21] Shun Iwase, Xingyu Liu, Rawal Khirodkar, Rio Yokota, and Kris M Kitani. RePOSE: fast 6d object pose refinement via deep texture rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3303–3312, 2021. 6, 7, 8

[22] Sagi Katz, Ayellet Tal, and Ronen Basri. Direct visibility of point sets. *ACM Transactions On Graphics (TOG)*, 26(3):24, 2007. 5

[23] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. SSD-6D: making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1521–1529, 2017. 2

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 6

[25] Laurent Kneip, Hongdong Li, and Yongduek Seo. Upnp: An optimal o (n) solution to the absolute pose problem with universal applicability. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 127–142. Springer, 2014. 2

[26] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. CosyPose: consistent multi-view multi-object 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 574–591. Springer, 2020. 8

[27] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009. 2, 5, 7, 8

[28] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. DeepGCNs: can GCNs go as deep as CNNs? In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9267–9276, 2019. 2

[29] Shichao Li, Zengqiang Yan, Hongyang Li, and Kwang-Ting Cheng. Exploring intermediate representation for monocular vehicle pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1873–1883, 2021. 1

[30] Xinyi Li and Haibin Ling. PoGO-Net: Pose graph optimization with graph neural networks. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5875–5885, 2021. 2

[31] Xinyi Li and Haibin Ling. GTCaR: Graph transformer for camera re-localization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 13670 of *Lecture Notes in Computer Science*, pages 229–246. Springer, 2022. 2

[32] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. DeepIM: deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 683–698, 2018. 6

[33] Zhigang Li, Gu Wang, and Xiangyang Ji. CDPN: coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7678–7687, 2019. 1, 2, 3, 6, 7

[34] Kevin Lin, Lijuan Wang, and Zicheng Liu. Mesh graphormer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12939–12948, 2021. 2

[35] Zhi-Hao Lin, Sheng-Yu Huang, and Yu-Chiang Frank Wang. Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1809, 2020. 2

[36] Fabian Manhardt, Wadim Kehl, and Adrien Gaidon. ROI-10D: monocular lifting of 2d detection to 6d pose and metric shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2069–2078, 2019. 1

[37] Eric Marchand, Hideaki Uchiyama, and Fabien Spindler. Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics*, 22(12):2633–2651, 2015. 1

[38] Markus Oberweger, Mahdi Rad, and Vincent Lepetit. Making deep heatmaps robust to partial occlusions for 3d object pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 119–134, 2018. 1, 2, 4

[39] Jaewoo Park and Nam Ik Cho. DProST: 6-dof object pose estimation using space carving and dynamic projective spatial transformer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 8

[40] Kiru Park, Timothy Patten, and Markus Vincze. Pix2Pose: pixel-wise coordinate regression of objects for 6d pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7668–7677, 2019. 1, 2

[41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019. 6

[42] Georgios Pavlakos, Xiaowei Zhou, Aaron Chan, Konstantinos G Derpanis, and Kostas Daniilidis. 6-DoF object pose from semantic keypoints. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2011–2018. IEEE, 2017. 2, 4

[43] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. PVNet: pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4561–4570, 2019. 1, 2, 4, 5, 7

[44] Mikael Persson and Klas Nordberg. Lambda twist: An accurate fast robust perspective three point (p3p) solver. In *Proceedings of the European conference on computer vision (ECCV)*, pages 318–332, 2018. 1

[45] Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. 3D graph neural networks for rgbd semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5199–5208, 2017. 2

[46] Yue Qian, Junhui Hou, Sam Kwong, and Ying He. PUGeo-Net: a geometry-centric network for 3d point cloud upsampling. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 752–769. Springer, 2020. 2

[47] Long Quan and Zhongdan Lan. Linear n-point camera pose determination. *IEEE Transactions on pattern analysis and machine intelligence*, 21(8):774–780, 1999. 1

[48] Mahdi Rad and Vincent Lepetit. BB8: a scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3828–3836, 2017. 1, 2

[49] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015. 3, 6

[50] Mahdi Saleh, Shun-Cheng Wu, Luca Cosmo, Nassir Navab, Benjamin Busam, and Federico Tombari. Bending graphs: Hierarchical shape matching using gated optimal transport. In *Proceedings of the IEEE/CVF Conference on Com-*

puter Vision and Pattern Recognition (CVPR), pages 11757–11767, 2022. 2

[51] Weijing Shi and Raj Rajkumar. Point-GNN: graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1711–1719, 2020. 2

[52] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3693–3702, 2017. 2

[53] Chen Song, Jiaru Song, and Qixing Huang. HybridPose: 6d object pose estimation under hybrid representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 431–440, 2020. 1, 7

[54] Alessandro Sperduti and Antonina Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997. 2

[55] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for CNN: viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2686–2694, 2015. 2

[56] Yongzhi Su, Mahdi Saleh, Torben Fetzer, Jason Rambach, Nassir Navab, Benjamin Busam, Didier Stricker, and Federico Tombari. ZebraPose: coarse to fine surface encoding for 6dof object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6738–6748, 2022. 2, 3, 4, 7, 8

[57] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 699–715, 2018. 2

[58] Fulin Tang, Yihong Wu, Xiaohui Hou, and Haibin Ling. 3d mapping and 6d pose computation for real time augmented reality on cylindrical objects. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(9):2887–2899, 2019. 1

[59] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 292–301, 2018. 1, 2

[60] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9627–9636, 2019. 3, 7, 8

[61] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. In *Conference on Robot Learning*, pages 306–316. PMLR, 2018. 1

[62] Jonathan Tremblay, Stephen Tyree, Terry Mosier, and Stan Birchfield. Indirect object-to-robot pose estimation from an external monocular rgb camera. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4227–4234. IEEE, 2020. 1

[63] Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1510–1519, 2015. 2

[64] Steffen Urban, Jens Leitloff, and Stefan Hinz. Mlpnp - a real-time maximum likelihood solution to the perspective-n-point problem. *arXiv preprint arXiv:1607.08112*, 2016. 2

[65] Nitika Verma, Edmond Boyer, and Jakob Verbeek. FeaStNet: feature-steered graph convolutions for 3d shape analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2598–2606, 2018. 2

[66] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. GDR-Net: geometry-guided direct regression network for monocular 6d object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16611–16621, 2021. 1, 2, 3, 6, 7, 8

[67] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364, 2020. 6, 7

[68] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions On Graphics (TOG)*, 38(5):1–12, 2019. 2, 3, 4, 5, 6

[69] Di Wu, Zhaoyong Zhuang, Canqun Xiang, Wenbin Zou, and Xia Li. 6D-VNet: end-to-end 6-dof vehicle pose estimation from monocular RGB images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 1

[70] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: a convolutional neural network for 6d object pose estimation in cluttered scenes. In *Robotics: Science and Systems*, 2018. 2, 6, 8

[71] Yan Xu, Kwan-Yee Lin, Guofeng Zhang, Xiaogang Wang, and Hongsheng Li. RNNPose: recurrent 6-dof object pose refinement with robust correspondence field estimation and pose optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14880–14890, 2022. 6

[72] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. DPOD: 6d pose object detector and refiner. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1941–1950, 2019. 1, 2

[73] Shaobo Zhang, Wanqing Zhao, Ziyu Guan, Xianlin Peng, and Jinye Peng. Keypoint-graph-driven learning framework for object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1065–1073, 2021. 2

[74] Guangyuan Zhou, Huiqun Wang, Jiaxin Chen, and Di Huang. PR-GCN: a deep graph convolutional network with point refinement for 6d pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2793–2802, 2021. 2

[75] Menglong Zhu, Konstantinos G Derpanis, Yinfei Yang, Samarth Brahmbhatt, Mabel Zhang, Cody Phillips, Matthieu Lecce, and Kostas Daniilidis. Single image 3d object detection and pose estimation for grasping. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3936–3943. IEEE, 2014. 1