RoboPEPP: Vision-Based <u>Rob</u>ot <u>P</u>ose and Joint Angle Estimation through <u>E</u>mbedding <u>P</u>redictive <u>P</u>re-Training

Raktim Gautam Goswami[1*]   Prashanth Krishnamurthy[1]   Yann LeCun[1,2]   Farshad Khorrami[1]

[1]New York University   [2]Meta-FAIR

Abstract

Vision-based pose estimation of articulated robots with unknown joint angles has applications in collaborative robotics and human-robot interaction tasks. Current frameworks use neural network encoders to extract image features and downstream layers to predict joint angles and robot pose. While images of robots inherently contain rich information about the robot's physical structures, existing methods often fail to leverage it fully; therefore, limiting performance under occlusions and truncations. To address this, we introduce RoboPEPP, a method that fuses information about the robot's physical model into the encoder using a masking-based self-supervised embedding-predictive architecture. Specifically, we mask the robot's joints and pre-train an encoder-predictor model to infer the joints' embeddings from surrounding unmasked regions, enhancing the encoder's understanding of the robot's physical model. The pre-trained encoder-predictor pair, along with joint angle and keypoint prediction networks, is then fine-tuned for pose and joint angle estimation. Random masking of input during fine-tuning and keypoint filtering during evaluation further improves robustness. Our method, evaluated on several datasets, achieves the best results in robot pose and joint angle estimation while being the least sensitive to occlusions and requiring the lowest execution time.

1. Introduction

Estimating the pose and joint angles of an articulated robot in the coordinate frame of an external camera is valuable for facilitating collaborative applications wherein an agent (e.g., a human or another robot) operates in a shared space with the articulated robot [10, 20, 26, 29, 32, 36]. Traditional robot pose estimation methods assume known joint
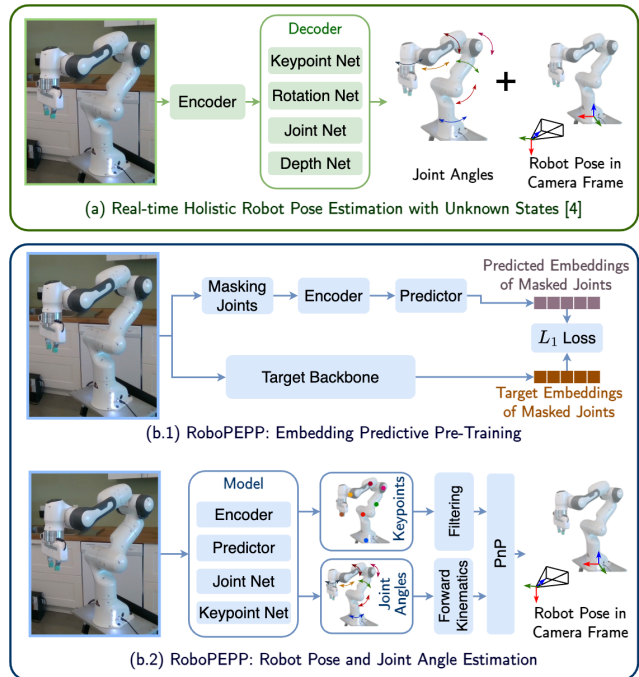
Figure 1. Comparison of an existing robot pose estimation method [5] with our RoboPEPP framework. RoboPEPP integrates joint masking-based pre-training (b.1) to enhance the encoder's grasp of the robot's physical model, combined with downstream networks, and keypoint filtering (b.2) to achieve high accuracy.

angles and capture multiple images with fiducial markers [12, 13, 25] attached to the robot's end-effector to establish 2D-3D correspondences between the image pixels and the robot's frame. Recent advancements in deep learning enable the prediction of 2D keypoints on robot joints from a single image [14, 18, 24, 34]. However, the assumption of known joint angles is not valid in many practical settings such as in collaborative robotics and human-robot interaction, where the joint angles may be unreliable or completely unknown. This challenge of simultaneously estimating joint angles and robot poses is particularly complex due to the high degrees of freedom in robotic systems and the infi-

nite space of potential robot poses and joint angle configurations. RoboPose [16] pioneered the field of robot pose estimation with unknown joints by using an iterative render-and-compare strategy. Later works [5, 35] enhanced the efficiency by employing neural networks that predict joint angles and robot poses in a single feed-forward pass. While input images provide rich information about the robot's physical structures and constraints, existing methods fail to leverage this fully, resulting in low performance in challenging scenarios like occlusions and truncations (i.e., instances where only part of the robot is visible).

Recently, self-supervised learning [3, 7] has shown that embedding predictive pre-training helps encoders develop a deeper semantic understanding of images. Inspired by such works, we propose RoboPEPP (Fig. 1), a robot pose estimation framework that integrates a joint-masking-based pre-training strategy to help the encoder better understand the robot's physical model. In this approach, the encoder extracts embeddings from the unmasked regions, which a predictor uses to estimate embeddings of the masked joints. In other words, the encoder-predictor network is trained to predict the embeddings of the joints using the context around them, thus improving the network's understanding of the robot's structure. While this pre-trained encoder supports various robotics tasks, we focus on robot pose estimation.

Following pre-training, the encoder and predictor are fine-tuned using downstream layers for joint angle prediction and 2D keypoint heatmap generation, allowing for end-to-end training. We further enhance the model's occlusion robustness by randomly masking the input while fine-tuning. During inference, the pixels with the highest values in the heatmaps are identified as 2D keypoints, while corresponding 3D keypoints in the robot's frame are computed using the forward kinematics and predicted joint angles. For cases where only part of the robot is visible in the image, we apply confidence-based keypoint filtering. Finally, we use the perspective-n-point (PnP) algorithm [19] on the filtered 2D-3D correspondences to estimate the robot's pose. In summary, our contributions are:

- A robot pose and joint angle estimation framework with embedding-predictive pre-training to enhance the network's understanding of the robot's physical model.
- An efficient network for robot pose and joint angle estimation using the pre-trained encoder-predictor alongside joint angle and keypoint estimators, trained using randomly masked inputs to enhance occlusion robustness.
- A confidence-based keypoint filtering method to handle cases where only part of the robot is visible in the image.
- Extensive experiments showing RoboPEPP's superior pose estimation, joint angle prediction, occlusion robustness, and computational efficiency.

2. Related Work

Classical methods for robot pose estimation typically involve attaching fiducial markers, such as ArUco [13], AprilTag [25], or ARTag [12], to the robot's end-effector to obtain easily detectable pixels in images. The corresponding 3D points in the robot's base coordinate frame are calculated using the robot's joint angles and forward kinematics. Using these correspondences and the camera intrinsics, an optimization problem is solved to find the robot-to-camera transformation [15, 27, 37], referred to here as the robot pose. This, however, requires multiple sets of correspondences from images taken at different robot configurations.

To streamline this process, DREAM [18] introduced a learning-based approach to detect multiple keypoint correspondences from a single image, estimating the pose using the PnP [19] algorithm. This method achieved performance comparable to classical approaches while requiring a single image. Building on this, PoseFusion [14] used multi-scale feature fusion to improve keypoint prediction accuracy. G-SAM [38] further improved robustness by adding a grouping and soft-argmax module, particularly useful when only part of the robot is visible. CTRNet [24] introduced a self-supervised sim-to-real approach, using differentiable PnP solver [9] and mesh renderer [28] to predict robot masks, which are compared against foreground segmentation for training. To incorporate information from prior frames, SGTAPose [33] employed a temporal attention framework.

These approaches, however, assume known joint angles, which is often impractical in real-world settings like collaborative robotics and human-robot interaction. To address this, RoboPose [16] estimated pose with unknown joint angles using an iterative render-and-compare approach, yielding strong results but at the cost of computational efficiency. An efficient framework [5] was later developed to predict joint angles and pose in a single, real-time feed-forward pass. RoboKeyGen [35] proposed another efficient method by lifting 2D keypoints to 3D using a stable diffusion model. While some frameworks [30, 31] used depth cameras for pose prediction, we restrict the discussion to methods using monocular RGB images, which are more widely accessible and avoid the need for specialized sensors.

Current robot pose estimation methods [5, 16, 35] do not fully utilize the rich features of the robot's physical model available in images. Meanwhile, self-supervised learning frameworks [1–3, 6–8] have advanced encoder training to extract robust image features with Joint-Embedding Predictive Architecture (JEPA) [3] using a masking-based pre-training strategy to enhance the encoder's semantic understanding of the image. Inspired by JEPA, we pre-train an encoder-predictor pair by masking regions around the robot's joints and predicting embeddings of the masked regions based on the surrounding context, thus enhancing the encoder's understanding of the robot's physical model. The
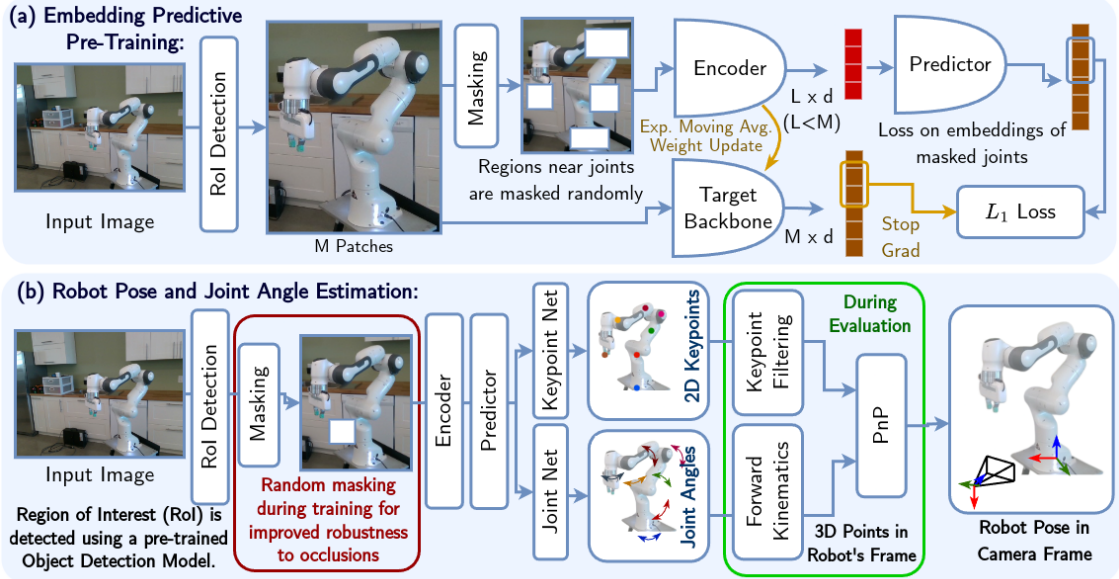
Figure 2. Overview of the RoboPEPP framework for robot pose and joint angle estimation. (a) Joint regions are masked to pre-train an encoder-predictor pair using an embedding predictive architecture. (b) The pre-trained encoder-predictor network is fine-tuned for robot pose estimation with Joint and Keypoint Prediction networks, using random masking during training to enhance occlusion robustness. During evaluation, keypoints are filtered, and a PnP algorithm estimates the robot's pose from the filtered 2D-3D correspondences.

pre-trained encoder-predictor pair is then fine-tuned along with joint and keypoint prediction networks, applying random masking during fine-tuning and confidence-based keypoint filtering at evaluation for improved robustness.

3. Methodology

Problem Description: Given a color image capturing an articulated robot with n joints, our objective is to estimate the joint angles $\Phi \in \mathbb{R}^n$ and the robot-to-camera rigid transformation matrix $T_R^C \in SE(3)$, with the robot frame being defined at its base. The robot's forward kinematics and the camera's intrinsic parameters are assumed to be known.

Method Overview: Our proposed framework (Fig. 2) comprises two stages: self-supervised pre-training of an encoder-predictor network (Sec. 3.1); and fine-tuning of the pre-trained encoder-predictor alongside 2D keypoint detection and joint angle estimation networks (Sec. 3.2). Predicted joint angles and forward kinematics yield 3D joint coordinates, which, combined with detected 2D keypoints, are used in a PnP solver to estimate pose (Sec. 3.3). During evaluation, confidence-based keypoint filtering and self-supervised fine-tuning on real-world data enhance accuracy.

3.1. Embedding Predictive Pre-Training

Building on embedding predictive architectures [3, 7], we employ a masking-based pre-training strategy tailored for robotic applications like pose and joint estimation. Masks are selected to occlude the regions around four randomly selected robot joints, or a random area if a joint is outside

the camera's field of view. Each mask covers 15–20% of the image with an aspect ratio between 0.75 and 1.5.

The original image consists of M patches, each sized 16×16 pixels. Let u_i represent the i-th patch, where $i \in \{1, 2, \ldots, M\}$, and let $\mathcal{B}$ denote the set of indices for the unmasked patches, with $L = |\mathcal{B}| < M$. With patches u_j, for $j \in \mathcal{B}$, as the context, a Vision Transformer (VIT) [11] encoder produces context embeddings $w_j \in \mathbb{R}^d$ for $j \in \mathcal{B}$. These context embeddings are then passed to a VIT-based predictor, which infers embeddings for all M patches of the original image, denoted $v_i \in \mathbb{R}^d$ for $i \in \{1, 2, \ldots, M\}$.

Meanwhile, a target backbone with the same architecture as the encoder extracts embeddings $\bar{v}_i \in \mathbb{R}^d$ for $i \in \{1, 2, \ldots, M\}$ directly from the original image. The embeddings for the masked patches, corresponding to indices $i \in \bar{\mathcal{B}}$ (where $\bar{\mathcal{B}}$ denotes the set of masked patch indices), are used to compute the L_1 loss during training, given by:

$$\mathcal{L}_{\text{pre-train}} = \frac{1}{|\bar{\mathcal{B}}|} \sum_{i \in \bar{\mathcal{B}}} \|v_i - \bar{v}_i\|_1. \tag{1}$$

Backpropagating through the encoder-predictor network and target backbone simultaneously risks trivial solutions, like constant predictions across all networks. To avoid this, we follow [3], backpropagating only through the encoder-predictor branch and updating the target backbone with an exponential moving average of the encoder's weights.

Our approach differs from JEPA [3] by using context-informed masking at joint locations. While JEPA learns deeper semantic representations by randomly masking the

3

input for tasks like object detection, we focus on encoding the robot's physical properties by specifically masking joint regions. This trains the encoder to infer the robot's joint-related information based on the surroundings, emulating a predictive understanding similar to how humans or animals deduce missing information about physical structures.

3.2. Keypoint Detection and Joint Angle Estimation

The pre-trained encoder and predictor are then fine-tuned, where they extract embeddings $v_i \in \mathbb{R}^d$ for $i \in \{1, 2, \ldots, M\}$ from images, which are used by the Joint Net and Keypoint Net to predict joint angles and 2D keypoints, respectively. To further increase occlusion robustness, random masks covering up to 20% of the image are applied during training. Consistent with Sec. 3.1, the predictor outputs all patch embeddings, including masked ones. This framework is trained using the loss functions in Sec. 3.2.3.

3.2.1. Joint Net

Using the patch embeddings, v_i, as input, the Joint Net predicts the angles for each of the robot's n joints. A global average pooling layer aggregates the patch embeddings v_i (for $i \in \{1, 2, \ldots, M\}$) into a single embedding $v_g \in \mathbb{R}^d$ to generate a global representation of the image. An iterative MLP-based approach [5] is then used to refine the joint angle predictions. Starting with a zero vector as the initial estimate Φ_{init}, the joint angles are iteratively updated through the MLP over $G = 4$ refinement steps (Fig. 3). The same MLP layer is used across all iterations, progressively refining the predicted joint angles Φ_{pred} for improved accuracy.

3.2.2. Keypoint Net

The Keypoint Net uses the patch embeddings to predict heatmaps for each of the k keypoints. The matrix $V = [v_1, v_2, \ldots, v_M]^T \in \mathbb{R}^{M \times d}$, contianing the patch embeddings, is reshaped into $\hat{V} \in \mathbb{R}^{m \times m \times d}$, where $m = \sqrt{M}$. With input image of 224×224 pixels and a patch size of 16×16 pixels, $m = 14$. The Keypoint Net takes $\hat{V}$ as input and applies four upsampling layers with output dimensions shown in Table 1. Each upsampling layer includes a transpose convolutional layer with a kernel size of 4, stride of 2, and one-pixel wide zero padding, followed by batch normalization, ReLU activation, and dropout. The channel dimension is gradually reduced from $d = 768$ to 256 across these layers. The output is then passed through a linear layer that reduces the channel dimension to k, followed by a sigmoid activation to produce heatmaps $H_{pred} \in \mathbb{R}^{224 \times 224 \times k}$. Typically, each keypoint is defined at a joint of the robot, with an additional keypoint at the base, making $k = n + 1$.

3.2.3. Loss Functions

For joint angles, we employ a mean squared error loss:

$$\mathcal{L}_{joint} = \frac{1}{n} \|\Phi_{pred} - \Phi_{gt}\|_2^2 \quad (2)$$
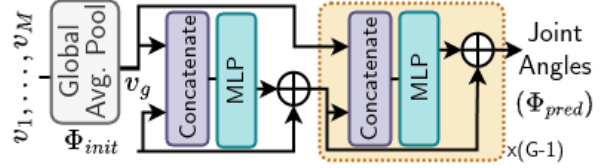

Figure 3. **Joint Net**: A global average pooling layer aggregates the patch embeddings, $v_1, \ldots, v_M$, into v_g, which is then iteratively refined using an MLP to estimate the joint angles.

Layer	Spatial Size	Channels
Input Size	14×14	768
Upsample 1	28×28	256
Upsample 2	56×56	256
Upsample 3	112×112	256
Upsample 4	224×224	256
Linear (Heatmaps)	224×224	k

Table 1. **Layer Output Sizes in Keypoint Net**: Patch embeddings are progressively upsampled through four layers and the channel dimension is reduced to k (the number of keypoint).

where Φ_{pred} and Φ_{gt} represent the predicted and ground truth joint angles, respectively. To enhance training convergence, mean-variance normalization is applied to Φ_{gt}. For keypoint detection, we utilize the focal loss [21]:

$$\mathcal{L}_{kp} = \text{FocalLoss}(H_{pred}, H_{gt}) \quad (3)$$

where H_{pred} and H_{gt} denote the predicted and ground truth heatmaps, respectively. H_{gt} is generated using unnormalized Gaussian probability density functions centered at each keypoint with a 2-pixel standard deviation.

The overall training loss is a weighted combination of the two losses: $\mathcal{L} = \mathcal{L}_{joint} + \alpha(t)\mathcal{L}_{kp}$., where $\alpha(t)$, dependent on epoch t, balances their relative importance. Since the joint angles are predicted in radians, $\mathcal{L}_{joint}$ tends to be much smaller than $\mathcal{L}_{kp}$, especially in early training. To address this, $\alpha(t)$ is initialized at 0.0001, increased to 0.01 after 5 epochs, 0.1 after 10 epochs, and finally to 1 after 40 epochs, ensuring a balanced curriculum for training.

3.3. Robot Pose Estimation

Keypoint Filtering: The final layer of the Keypoint Net contains a sigmoid nonlinearity, that produces heatmaps with pixel values between 0 and 1, representing keypoint confidence at each pixel. The pixel with the highest confidence indicates the keypoint location. However, when only a portion of the robot is visible, some keypoints may lie outside the image, leading to low confidence scores across the heatmap for these keypoints (Fig. 4). Selecting the pixel with the highest confidence in such cases can be misleading, as no pixel accurately represent the true keypoint. To address this, during evaluation, we apply a threshold, only considering keypoints with confidence above it. For use

4

(a) Example from panda-3cam_realsense (RS)

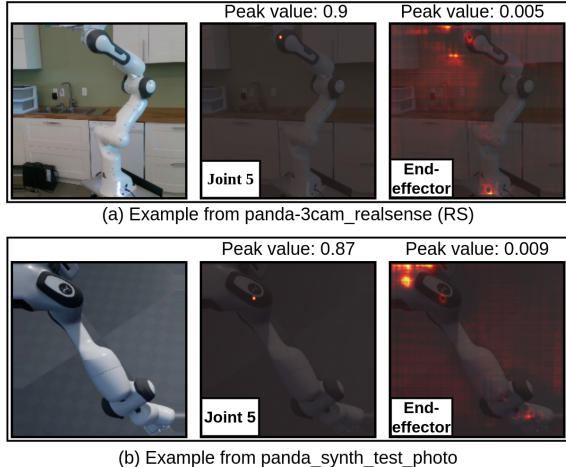(b) Example from panda_synth_test_photo

Figure 4. The examples show predicted heatmaps for Joint 5 and the End-Effector overlaid on the original image. The End-Effector, being positioned outside the field of view, produces noisy heatmaps with lower confidence (measured by peak values). Heatmap pixel values are normalized here for visual clarity.

with a PnP algorithm [19] for pose estimation, we require a minimum of four 2D-3D correspondences. If fewer than four keypoints remain after filtering, we iteratively reduce ϵ by 0.025 until at least four keypoints are retained.

Pose Estimation: The robot's pose is estimated using the EPnP algorithm [19] with the filtered 2D-3D correspondences and known camera intrinsics. As keypoints are defined on joints, we obtain the 3D points corresponding to the 2D keypoints using the robot's forward kinematics and predicted joint angles.

Sim-to-Real Self-Supervised Training: In addition to supervised training for pose estimation, our method supports self-supervised fine-tuning of the trained models on real-world data to bridge the sim-to-real gap. Specifically, we use a differentiable PnP algorithm [9, 24] and estimate the robot's pose and transform 3D joint locations from the robot to the camera frame. These transformed points are projected onto the image plane, yielding the projected keypoints, $P_{proj} \in \mathbb{R}^{k \times 2}$. We then minimize the mean squared error between P_{proj} and the predicted keypoints P_{pred}

$$\mathcal{L}_{\text{ssl}} = \frac{1}{k} \sum_{i=1}^{k} \|P_{pred}^i - P_{proj}^i\|_2^2 \qquad (4)$$

where P_{pred}^i and P_{proj}^i are the ith keypoint in P_{pred} and P_{proj}, respectively. During sim-to-real finetuning, we use a lower learning rate than in the prior supervised training. Further, to prevent model collapse, the Keypoint Net's learning rate is set close to zero.

Region of Interest (RoI) Detection: During evaluation, we utilize the GroundingDINO [22] object detection model to automatically locate the region of interest around the robot. The detected region is cropped and resized to 224×224

pixels. During training, we use ground truth bounding box information. However, to ensure our model's robustness to region-of-interest detection, we employ a training curriculum: we use the ground truth bounding boxes and expand them by adding random offsets sampled from the uniform distribution $\mathcal{U}(0, \lambda)$ to their edges, with λ progressing from 0 (first 30 epochs) to 30 pixels at epoch 30, 50 pixels at epoch 50, 80 pixels at epoch 70, 100 pixels at epoch 90, and 120 pixels at epoch 110. This approach encourages the model to generalize effectively, even with noisy region-of-interest detection during inference.

4. Experiments

4.1. Dataset and Implementation Details

We evaluate our framework on the DREAM dataset [18] that includes three robots (Franka Emika Panda, Kuka iiwa7, Rethink Baxter) and contains the following for each robot: Panda – synthetic domain-randomized (DR) training, DR test (Panda DR), photo-realistic test (Panda Photo), four real-world test (Panda AK, XK, RS, ORB) sequences; Kuka – synthetic DR training, DR test (Kuka DR), photo-realistic test (Kuka Photo) sequences; Baxter – synthetic DR training and test (Baxter DR) sequences.

The encoder is pre-trained using our self-supervised embedding predictive strategy (Sec. 3.1) for 200 epochs on the DR training sequences of all robots, using AdamW [23] optimizer with an initial learning rate of 10^{-3}. For end-to-end fine-tuning (Sec. 3.2), models are trained separately for each robot for 200 epochs with AdamW optimizer (learning rate 10^{-4}). Sim-to-real fine-tuning is performed for 10 epochs. More details are in the supplementary material.

4.2. Results

4.2.1. Robot Pose Prediction

We evaluate RoboPEPP by computing the average distance [5, 16, 18], $ADD = \frac{1}{n} \sum_0^n \|\hat{T}_R^C \hat{p}_i^{\{R\}} - p_i^{\{C\}}\|_2$ between predicted and ground truth joint positions in the camera frame for each image, where $\hat{T}_R^C$ is the predicted robot pose in the camera frame, $\hat{p}_i^{\{R\}}$ is the estimated position of joint i in the robot's frame (computed from predicted joints and forward kinematics), and $p_i^{\{C\}}$ is the ground truth joint position in the camera frame ($i = 0$ signifies the robot base).

In Table 2, we report the area-under-the-curve (AUC) of the average distance (ADD) across various thresholds, where higher AUC values indicate greater accuracy. We compare RoboPEPP with Real-Time Holistic Robot Pose Estimation with Unknown States (referred to as HPE in this manuscript) [5], RoboPose [16], and three variants of DREAM [18], though the latter assume known joint angles. To the best of our knowledge, RoboPose, HPE, and RoboKeyGen [35] are the only approaches besides RoboPEPP that predict robot pose with unknown joint

Method	Known Joint Angles	Known Bounding Box	Panda						Kuka		Baxter
			Synthetic		Real				Synthetic		Synthetic
			Photo	DR	AK	XK	RS	ORB	Photo	DR	DR
DREAM-F	Yes	No	79.5	81.3	68.9	24.4	76.1	61.9	-	-	-
DREAM-Q	Yes	No	74.3	77.8	52.4	37.5	78.0	57.1	-	-	75.5
DREAM-H	Yes	No	81.1	82.9	60.5	64.0	78.8	69.1	72.1	73.3	-
HPE	No	Yes	82.0	82.7	82.2	76.0	75.2	75.2	73.9	75.1	58.8
RoboPose	No	No	79.7	82.9	70.4	77.6	74.3	70.4	73.2	**80.2**	32.7
HPE*	No	No	40.7	41.4	66.7	-	49.1	51.6	56.7	56.2	9.8
RoboPEPP (Ours)	No	No	**84.1**	**83.0**	**75.3**	**78.5**	**80.5**	**77.5**	**76.1**	76.2	**34.4**

Table 2. Comparison of robot pose estimation using AUC on the ADD metric. Best values among methods using unknown joint angles and bounding boxes during evaluation are bolded. HPE* denotes HPE [5] evaluated with the same off-the-shelf bounding box detector as RoboPEPP. HPE* was not evaluated on Panda XK since corresponding model weights were unavailable.

angles. However, RoboKeyGen evaluates on a different dataset, and its code is unavailable, preventing direct comparison. Nonetheless, its reported AUC on similar datasets is lower than ours. Moreover, HPE [5] assumes known bounding boxes during evaluation, a condition often unrealistic in practice. Therefore, we also evaluate HPE with our bounding box detection strategy (denoted HPE*) in Table 2.

RoboPEPP yields the highest scores across all sequences (except for Kuka DR where it remains competitive) among methods with unknown joint angles and bounding boxes. HPE [5], on the other hand, shows sensitivity to bounding box selection, with its performance dropping when ground truth bounding boxes are unavailable. Our analysis has shown that using bounding boxes just 5 pixels wider than ground truth reduces HPE's accuracy by up to 25% on the Panda Photo test set and by around 50% with 10-pixel wider boxes. While both RoboPEPP and HPE are trained using ground truth bounding boxes, RoboPEPP's training strategy (Sec. 3.3) reduces its dependency on them. Notably, RoboPEPP outperforms HPE on most sequences even when HPE has acces to known bounding boxes during evaluation.

A qualitative comparison of RoboPEPP with Robo-Pose [16] and HPE [5] on Panda Photo test dataset (example 1) and the occlusion dataset of Sec. 4.2.3 (examples 2 and 3) is presented in Fig. 5. Each method uses the input image to predict pose and joint angles, rendering a robot mesh that is projected and overlaid onto the original image, where closer alignment indicates higher prediction accuracy. Example 1 depicts a case where only part of the robot is visible and examples 2 and 3 show cases of occlusions (detailed in Sec. 4.2.3). In these challenging scenarios, RoboPEPP achieves highly accurate overlays while other methods are less precise, as highlighted by the red rectangles. In Fig. 5, HPE is used with ground truth bounding boxes.

4.2.2. Joint Prediction

In Table 3, we report the mean absolute error (in degrees) for joint angle prediction on the Panda Photo, Panda DR, Kuka Photo, and Kuka DR test datasets. The keypoints corresponding to the end-effector and the final joint (i.e., the

		Method	J1	J2	J3	J4	J5	J6	Avg.
Panda	Photo	RoboPose	7.7	3.5	4.3	3.4	7.3	8.1	5.7
		HPE (Known BBox)	6.1	2.2	3.6	2.0	6.2	6.6	4.5
		RoboPEPP	**4.4**	**1.8**	**2.2**	**1.8**	**4.4**	4.8	**3.2**
	DR	RoboPose	6.1	2.7	3.6	2.5	6.3	8.1	4.9
		HPE (Known BBox)	6.2	**2.2**	3.9	**1.9**	5.9	6.6	4.4
		RoboPEPP	**4.9**	2.3	**2.7**	2.2	**4.9**	**5.4**	**3.8**
Kuka	Photo	RoboPose	4.9	5.1	6.7	6.0	10.8	9.6	7.2
		HPE (Known BBox)	4.8	3.8	5.0	**2.8**	4.9	5.9	4.5
		RoboPEPP	**3.8**	**2.8**	**4.6**	3.1	**3.8**	**5.4**	**3.9**
	DR	RoboPose	4.4	**2.8**	5.4	3.4	12.5	8.5	6.2
		HPE (Known BBox)	4.6	3.6	**4.9**	**2.8**	5.2	**6.1**	4.5
		RoboPEPP	**3.7**	3.5	5.1	3.5	**4.1**	6.2	**4.3**
Avg.		RoboPose	5.8	3.5	5.0	3.8	9.2	8.6	6.0
		HPE (Known BBox)	5.4	3.0	4.4	**2.4**	5.6	6.3	4.5
		RoboPEPP	**4.2**	**2.6**	**3.7**	2.7	**4.3**	**5.5**	**3.8**

Table 3. Mean absolute error between the predicted and actual joint angles (in degrees) for the Panda and Kuka synthetic test sets.

joint nearest to the end-effector) lie along the axis of rotation of this joint, making their locations independent of this joint's angle. Consequently, we predict the angles of all joints except the last one, assigning a random angle to it during evaluation. Thus, Table 3 presents the mean absolute error for the first six joint angles. We compare RoboPEPP with HPE [5] and RoboPose [16]. RoboPEPP demonstrates the lowest average joint prediction error on all datasets. Although HPE utilizes known bounding boxes for region-of-interest detection, RoboPEPP still outperforms HPE by over 15% on average across all datasets in Table 3. When HPE is tested without ground truth bounding boxes, its performance drops to an average error of 7.2 degrees.

4.2.3. Robustness to Occlusions

In addition to achieving high performance across various metrics and datasets, RoboPEPP demonstrates robustness to occlusions. To evaluate this, we compared the performance of RoboPEPP with other methods [5, 16] on a custom dataset, created by adding synthetic occlusions to Panda Photo. Specifically, we overlaid black rectangular or circular masks at random positions on the robot, ensuring that the masks covered at least some part of the robot (and not just the background). We generated four test se-
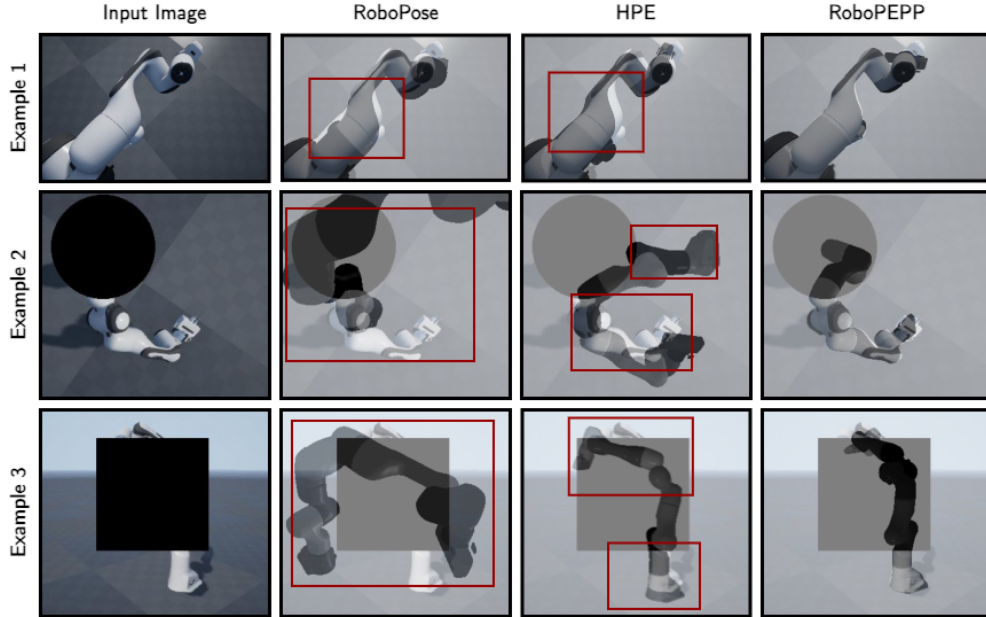
Figure 5. **Qualitative Comparison on Panda Photo (Example 1) and Occlusion (Example 2 and 3) datasets**: Predicted poses and joint angles are used to generate a mesh overlaid on the original image, where closer alignment indicates greater accuracy. Highlighted rectangles indicate regions where other methods' meshes misalign, while RoboPEPP achieves high precision.
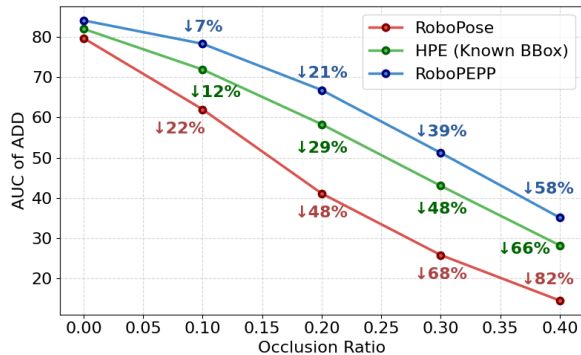

Figure 6. AUC comparison of the distance metric under varying occlusion levels, evaluated on the dataset in Sec. 4.2.3. Percentages next to the lines indicate the relative drop in each method's performance compared to their performance without occlusions.
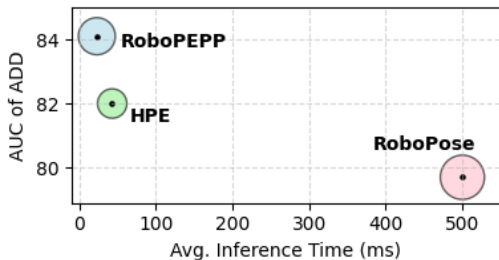

Figure 7. Execution time and computation analysis on the Panda Photo test dataset with RoboPEPP showing best performance and accuracy. The circle sizes in the plot represent model FLOPs.

quences with occlusion ratios of 0.1, 0.2, 0.3, and 0.4 on the RoI area in each image, respectively. This approach differs from the masking used during our training, where the model is informed about the number of masked patches. Here, the model processes occluded images as it would any other input image, without any knowledge of the occlusion.

In Fig. 6, we plot the AUC of the ADD against the occlusion ratio. The plot also includes the percentage decrease in AUC relative to the respective model's performance without occlusion. RoboPEPP demonstrates superior robustness to occlusion, achieving an AUC score of 35.1 even when 40% of the RoI is occluded, compared to 28.2 for HPE and 14.5 for RoboPose. Examples 2 and 3 in Fig. 5 provide qualitative comparisons of RoboPEPP, HPE [5], and Robo-Pose [16] on the occlusion dataset. RoboPEPP demonstrates superior performance, even in challenging cases like example 3, where most of the robot is occluded. In both examples 2 and 3, RoboPose produces inaccurate results, while HPE shows partial overlap but still exhibits notable inaccuracies, highlighted by the red rectangles.

4.2.4. Percentage of Correct Keypoints

The accuracy of 2D keypoint detection affects the overall performance of RoboPEPP. Therefore, in Table 4, we report the percentage of correct keypoints (PCK) within thresholds of 2.5, 5, and 10 pixels on the Panda Photo dataset. Since HPE [5] and RoboPose [16] do not rely on 2D keypoint detection for pose estimation, we include only DREAM [18] as a baseline for comparison. RoboPEPP achieves high average PCK scores, with 0.43 @ 2.5 pixels, 0.73 @ 5 pix-
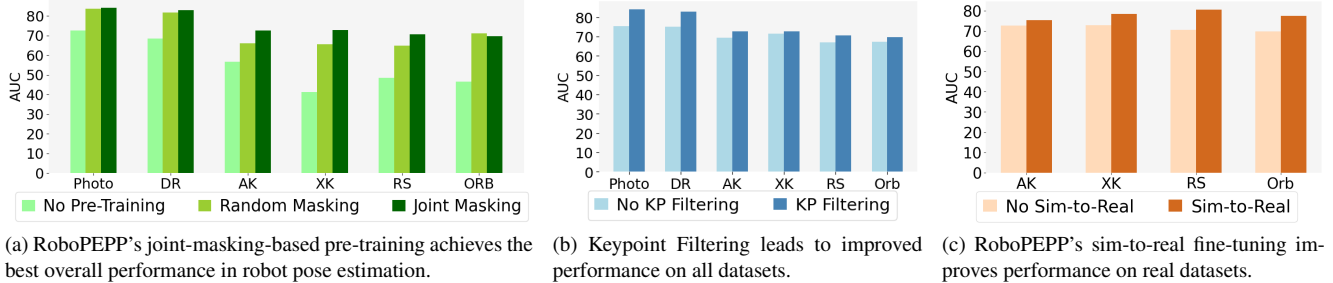
(a) RoboPEPP's joint-masking-based pre-training achieves the best overall performance in robot pose estimation.

(b) Keypoint Filtering leads to improved performance on all datasets.

(c) RoboPEPP's sim-to-real fine-tuning improves performance on real datasets.

Figure 8. Ablation studies on (a) Pre-Training, (b) Keypoint Filtering, and (c) Sim-to-Real fine-tuning on the Panda test datasets.

Dataset	Method	PCK @ (pixel)		
		2.5	5	10
DR	DREAM	0.79	0.88	0.9
	RoboPEPP	**0.84**	**0.91**	**0.93**
Photo	DREAM	0.77	0.87	0.9
	RoboPEPP	**0.87**	**0.92**	**0.94**
AK	DREAM	**0.36**	**0.65**	0.9
	RoboPEPP	0.16	0.62	**0.93**
XK	DREAM	**0.15**	**0.37**	0.59
	RoboPEPP	0.09	**0.37**	**0.96**
RS	DREAM	0.24	**0.83**	0.96
	RoboPEPP	**0.31**	0.82	**0.97**
ORB	DREAM	**0.28**	0.67	0.83
	RoboPEPP	**0.28**	**0.73**	**0.96**
Avg.	DREAM	**0.43**	0.71	0.85
	RoboPEPP	**0.43**	**0.73**	**0.95**

Table 4. Comparison of Percentage of Correct Keypoints (PCK) at different pixel thresholds across the Panda test datasets.

els, and 0.95 @ 10 pixels. While DREAM outperforms RoboPEPP on certain metrics, such as PCK@2.5 pixels on the Panda AK and Panda XK, RoboPEPP demonstrates superior accuracy across most other metrics and on average, highlighting its robust keypoint detection performance.

4.2.5. Execution Time

To demonstrate the practical effectiveness of the proposed RoboPEPP method, we compare execution times (in milliseconds) in Fig. 7. The circle sizes in the figure correspond to the relative number of floating-point operations (FLOPs) required by each model. All evaluations were conducted on a system equipped with an Nvidia RTX A4000 GPU, an Intel(R) i9 CPU, and 128 GB RAM, using the Panda Photo test dataset. Consistent with previous work [5], we report only model execution time, excluding pre-processing steps such as data loading and RoI detection. Despite having a slightly higher FLOPs count than HPE [5], RoboPEPP achieves the highest AUC ADD score and the fastest execution time, completing inference in just 23 milliseconds.

4.3. Ablation Studies

Embedding Predictive Pre-Training: To assess the impact of embedding predictive pre-training in RoboPEPP, we conducted an ablation study comparing three versions of the model: a version of RoboPEPP without pre-training,

and a version pre-trained with random masking instead of joint-specific masking, and standard RoboPEPP (i.e., pre-trained with joint masking). For all experiments, we utilized the same model architecture and training settings. The bar graphs in Fig. 8a illustrate that pre-training significantly improves performance. While the model trained with the default masking strategy demonstrated competitive results on synthetic test datasets, the model trained with joint-specific masking achieved better performance on real-world datasets in general. Note that the real-world results shown here do not include the sim-to-real fine-tuning of Sec. 3.3. Further, on the occlusion dataset (Sec. 4.2.3) with a 0.4 occlusion ratio, the model without pre-training achieves AUC of 22.6, the one with random masking reaches 30, and RoboPEPP achieves 35.1, highlighting the latter's occlusion robustness.
Keypoint Filtering: In Fig. 8b, we demonstrate that the integration of keypoint filtering (KP filtering) enhances performance across all datasets by helping in filtering out keypoints that fall outside the camera's field of view. Similar to Fig. 8a, the real-world results presented in Fig. 8b do not include any sim-to-real fine-tuning.
Sim-to-Real Fine-Tuning: In Fig. 8c, we show performance gains from sim-to-real self-supervised training, with the model's accuracy improving by an average of ∼6 points after fine-tuning. Our sim-to-real training requires only 10 epochs, with each epoch lasting around 2 minutes.

5. Conclusion

We introduced a novel framework RoboPEPP enhancing robot pose and joint angle estimation using an embedding predictive pre-training strategy. RoboPEPP uses a joint-masking-based method to pre-train an encoder-predictor pair to be integrated into downstream networks for joint and pose predictions. Experimental results show RoboPEPP's superior performance, particularly in handling occlusions due to the combined effects of pre-training and random masking during fine-tuning. RoboPEPP's training helps fuse knowledge of the robot's physical model within the encoder, making RoboPEPP effective for pose estimation and versatile for broader applications such as system dynamic prediction and imitation learning.

References

[1] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. *arXiv preprint arXiv:1911.05371*, 2019. 2

[2] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, Pascal Vincent, Armand Joulin, Mike Rabbat, and Nicolas Ballas. Masked siamese networks for label-efficient learning. In *Proceedings of the European Conference on Computer Vision*, pages 456–473. Springer, 2022.

[3] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15619–15629, 2023. 2, 3, 11

[4] Jimmy Lei Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 11

[5] Shikun Ban, Juling Fan, Xiaoxuan Ma, Wentao Zhu, Yu Qiao, and Yizhou Wang. Real-time holistic robot pose estimation with unknown states. *arXiv preprint arXiv:2402.05655*, 2024. 1, 2, 4, 5, 6, 7, 8, 12, 13

[6] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021. 2

[7] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. V-JEPA: Latent video prediction for visual representation learning, 2024. 2, 3

[8] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020. 2

[9] 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*, pages 8100–8109, 2020. 2, 5

[10] Sammy Christen, Wei Yang, Claudia Pérez-D'Arpino, Otmar Hilliges, Dieter Fox, and Yu-Wei Chao. Learning human-to-robot handovers from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9654–9664, 2023. 1

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations*, 2020. 3, 11

[12] Mark Fiala. Artag, a fiducial marker system using digital techniques. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 590–596. IEEE, 2005. 1, 2

[13] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014. 1, 2

[14] Xujun Han, Shaochen Wang, Xiucai Huang, and Zhen Kan. Posefusion: Multi-scale keypoint correspondence for monocular camera-to-robot pose estimation in robotic manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 795–801, 2024. 1, 2

[15] Jarmo Ilonen and Ville Kyrki. Robust robot-camera calibration. In *Proceedings of the International Conference on Advanced Robotics*, pages 67–74. IEEE, 2011. 2

[16] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Single-view robot pose and joint angle estimation via render & compare. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1654–1663, 2021. 2, 5, 6, 7, 12, 13

[17] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *Proceedings of the International Conference on Learning Representations*, 2022. 11

[18] Timothy E Lee, Jonathan Tremblay, Thang To, Jia Cheng, Terry Mosier, Oliver Kroemer, Dieter Fox, and Stan Birchfield. Camera-to-robot pose estimation from a single image. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 9426–9432. IEEE, 2020. 1, 2, 5, 7, 11, 12

[19] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPnP: An accurate O(n) solution to the p n p problem. *International journal of computer vision*, 81:155–166, 2009. 2, 5

[20] Shushuai Li, Christophe De Wagter, and Guido CHE De Croon. Self-supervised monocular multi-robot relative localization with efficient deep neural networks. In *Proceedings of the International Conference on Robotics and Automation*, pages 9689–9695. IEEE, 2022. 1

[21] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017. 4

[22] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding Dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. 5, 11

[23] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5, 11

[24] Jingpei Lu, Florian Richter, and Michael C. Yip. Markerless camera-to-robot pose estimation via self-supervised sim-to-real transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21296–21306, 2023. 1, 2, 5

[25] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3400–3407. IEEE, 2011. 1, 2

[26] Andreas Papadimitriou, Sina Sharif Mansouri, and George Nikolakopoulos. Range-aided ego-centric collaborative pose estimation for multiple robots. *Expert Systems with Applications*, 202:117052, 2022. 1

[27] Frank C Park and Bryan J Martin. Robot sensor calibration: solving AX= XB on the euclidean group. *IEEE Transactions on Robotics and Automation*, 10(5):717–721, 1994. 2

[28] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020. 2

[29] Yara Rizk, Mariette Awad, and Edward W Tunstel. Cooperative heterogeneous multi-robot systems: A survey. *ACM Computing Surveys (CSUR)*, 52(2):1–31, 2019. 1

[30] Alessandro Simoni, Stefano Pini, Guido Borghi, and Roberto Vezzani. Semi-perspective decoupled heatmaps for 3D robot pose estimation from depth maps. *IEEE Robotics and Automation Letters*, 7(4):11569–11576, 2022. 2

[31] Alessandro Simoni, Francesco Marchetti, Guido Borghi, Federico Becattini, Lorenzo Seidenari, Roberto Vezzani, and Alberto Del Bimbo. Robot pose nowcasting: Forecast the future to improve the present. *arXiv preprint arXiv:2308.12914*, 2023. 2

[32] Mikael Svenstrup, Soren Tranberg, Hans Jorgen Andersen, and Thomas Bak. Pose estimation and adaptive robot behaviour for human-robot interaction. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3571–3576. IEEE, 2009. 1

[33] Yang Tian, Jiyao Zhang, Zekai Yin, and Hao Dong. Robot structure prior guided temporal attention for camera-to-robot pose estimation from image sequence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8917–8926, 2023. 2

[34] Yang Tian, Jiyao Zhang, Zekai Yin, and Hao Dong. Robot structure prior guided temporal attention for camera-to-robot pose estimation from image sequence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8917–8926, 2023. 1

[35] Yang Tian, Jiyao Zhang, Guowei Huang, Bin Wang, Ping Wang, Jiangmiao Pang, and Hao Dong. Robokeygen: Robot pose and joint angles estimation via diffusion-based 3D keypoint generation. *arXiv preprint arXiv:2403.18259*, 2024. 2, 5

[36] Chengjun Xu, Xinyi Yu, Zhengan Wang, and Linlin Ou. Multi-view human pose estimation in human-robot interaction. In *Proceedings of the Annual Conference of the IEEE Industrial Electronics Society*, pages 4769–4775. IEEE, 2020. 1

[37] Dekun Yang and John Illingworth. Calibrating a robot camera. In *Proceedings of the British Machine Vision Conference*, pages 1–10, 1994. 2

[38] Xiaopin Zhong, Wenxuan Zhu, Weixiang Liu, Jianye Yi, Chengxiang Liu, and Zongze Wu. G-SAM: A robust one-shot keypoint detection framework for PnP based robot pose estimation. *Journal of Intelligent & Robotic Systems*, 109(2): 28, 2023. 2

Supplementary Material

A1. Encoder and Predictor Architectures

As described in Sec. 3.1, we use Vision Transformer (ViT) [11] architectures for both the encoder and predictor, similar to [3]. The input image, originally sized at 640×480 pixels, is cropped based on the region of interest, resized to obtain 224 pixels along its longer side, and padded to yield a 224×224 resolution. A convolutional layer with a kernel size of 16 and a stride of 16 serves as the patch embedding layer, converting the image into L patches of size 16×16 each with a channel dimension of $d = 768$. These patches are flattened, and learnable positional embeddings, initialized as 2D sinusoidal functions, are added to the patches. The combined representations are then passed through 12 transformer blocks. Each block contains multi-headed self-attention with 12 heads, drop-path regularization [17], layer normalization [4], and a multi-layer perceptron (MLP). The output of the final transformer block undergoes another layer normalization step, resulting in the encoder output $w_j \in \mathbb{R}^{768}$ for $j \in \{1, \ldots, L\}$.

During evaluation, for the image of size 224×224 and a patch size of 16×16, the number of patches is computed as

$$L = M = \frac{224}{16} \times \frac{224}{16} = 14 \times 14 = 196. \qquad (5)$$

However, during training, only the unmasked patches are considered, so $L < M$, i.e., $L < 196$.

The predictor takes the encoder output and reduces the embedding dimension of the patches from 768 to 384 using a linear layer. It also adds positional embeddings, similar to the encoder. During training, the $L(< M)$ embeddings corresponding to the unmasked patches and $(M - L)$ learnable mask tokens are concatenated to represent all patches of the original image, including the masked ones. These embeddings are then processed through 12 transformer blocks. The final output's dimension is increased to 768 to match the encoder's output dimension, resulting in the predictor output v_i for $i \in \{1, \ldots, M\}$.

The target backbone uses the same architecture as the encoder but directly operates on all $M = 196$ patches during training. It produces outputs $\bar{v}_i$ for $i \in \{1, \ldots, M\}$. As outlined in the manuscript, during embedding predictive pre-training, an L_1 loss between v_i and $\bar{v}_i$ is used to update the weights of the encoder and predictor. Following [3], the target backbone is updated using an exponential moving average of the encoder's weights.

A2. Training Settings

Embedding Predictive Pre-Training: The AdamW optimizer [23] with an initial learning rate of 10^{-4} is used for embedding predictive pre-training. The learning rate is linearly increased to 10^{-3} over the first 10 epochs and subsequently decreased to 10^{-6} using a cosine annealing scheduler. The network is pre-trained for a total of 200 epochs with a batch size of 320. Weight decay is linearly increased from 0.04 to 0.4 during pre-training. For the exponential moving average (EMA) update of the target backbone's weights, a momentum value of 0.996 is used, which is linearly increased to 1.0 over the training process.

Keypoint Detection and Joint Angle Estimation: As detailed in the manuscript, the pre-trained encoder-predictor pair is fine-tuned along with the Keypoint Net and Joint Net. An AdamW optimizer [23] is used with an initial learning rate of 10^{-4}, which is decreased to 10^{-8} using a cosine annealing scheduler. The network is trained for a total of 200 epochs with a batch size of 140.

Sim-to-Real Self-Supervised Training: To bridge the sim-to-real gap, the trained networks are fine-tuned on real datasets with self-supervised training, as described in Sec. 3.3. An AdamW optimizer [23] is used with learning rates of 10^{-7} for the encoder and predictor, 10^{-5} for the joint network. The learning rate for the keypoint network is set close to zero to prevent model collapse. These learning rates are decreased by a factor of 10^8 over the training process. Models are fine-tuned separately for each real-world dataset for 10 epochs with a batch size of 64.

A3. Region of Interest Detection

We utilize the pre-trained GroundingDINO [22] object detection model to identify the region of interest, as described in Sec. 3.3. GroundingDINO is a highly accurate open-set object detector that accepts an (image, text) pair as input and outputs bounding boxes corresponding to regions of the image described by the text query. For all real and photo-realistic test datasets, we use the text query "robotic arm." However, for the Panda, Kuka, and Baxter domain-randomized datasets, we use the query "robot" because these images often contain multiple objects, some of which resemble arms and can confuse the detection model. All other parameters of GroundingDINO are left at their default values. To address scenarios where only a portion of the robot is detected, we expand all the detected bounding boxes, especially for real datasets. Increasing all the bounding box sizes by 100 pixels on all sides generally yields robust robot pose estimation results. However, some fine-tuning of this parameter may be necessary for optimal performance depending on the specific dataset. Nonetheless, high performance is obtained even without fine-tuning.

A4. Dataset Details

We evaluate our method on the DREAM dataset [18], which includes sequences from three robots: Franka Emika Panda (Panda), Kuka iiwa7 (Kuka), and Rethink Robotics Bax-
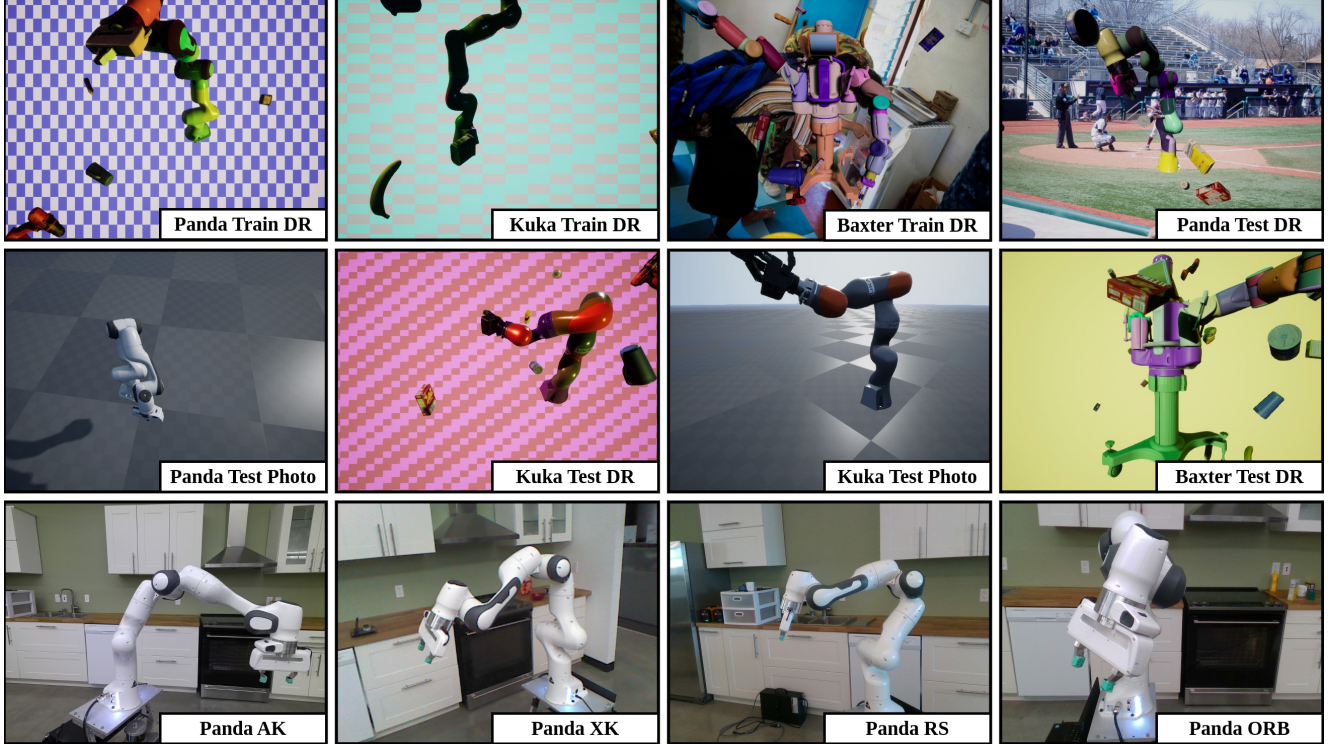
Figure A1. Example images from each of the training and test sequences from the DREAM dataset [18].

	Known Joint Angles	Known Bounding Box	Real-World Sequences				
			Panda AK	Panda XK	Panda RS	Panda ORB	Average
DREAM-F	Yes	No	11413	491911	2077	95319	150180
DREAM-Q	Yes	No	78089	54178	27	64248	49136
DREAM-H	Yes	No	57	7382	24	25685	8287
HPE	No	Yes	19	24	25	25	23
RoboPose	No	No	34	**22**	26	30	28
HPE*	No	No	46	-	61	52	53
RoboPEPP (Ours)	No	No	**29**	**22**	**23**	**27**	**26**

Table A1. Comparison of robot pose estimation using mean ADD (in millimeters), with lower a value signifying better performance. The best values among methods that use unknown joint angles and unknown bounding boxes during evaluation are bolded. HPE* denotes HPE [5] evaluated with the same off-the-shelf bounding box detector as RoboPEPP. HPE* was not evaluated on Panda XK since corresponding model weights were unavailable.

ter (Baxter). The dataset provides training and testing sequences in both synthetic and real-world settings, as detailed in Table A2. The synthetic data comprises domain-randomized (DR) and photo-realistic (Photo) sequences. For real-world data, sequences of the Panda robot were captured using Microsoft Azure Kinect (AK), Xbox 360 Kinect (XK), and Intel RealSense D415 (RS) cameras, with the cameras positioned at fixed locations. Additionally, the Panda ORB dataset was collected using a RealSense camera but with varying camera placements. Example images from each dataset sequence are illustrated in Fig. A1.

A5. Additional Results

A5.1. Mean ADD

In Table A1, we present the mean ADD (Average Distance) values (ADD defined in Sec. 4.2.1) on the Panda real-world datasets. Consistent with Table 2, we compare our method, RoboPEPP, against DREAM [18], RoboPose [16], HPE [5], and HPE* (HPE using our bounding box detection strategy). RoboPEPP achieves the lowest mean ADD across all real-world data sequences among methods that operate with unknown joint angles and bounding boxes. DREAM [18], which detects 2D keypoints and employs them in a PnP

	Dataset	Real	# Images
Training	Panda Train DR	×	104972
	Kuka Train DR	×	104977
	Baxter Train DR	×	104982
Testing	Panda Photo	×	5997
	Panda DR	×	5998
	Panda AK	✓	6369
	Panda XK	✓	4966
	Panda RS	✓	5944
	Panda ORB	✓	32315
	Kuka Photo	×	5999
	Kuka DR	×	5997
	Baxter DR	×	5982

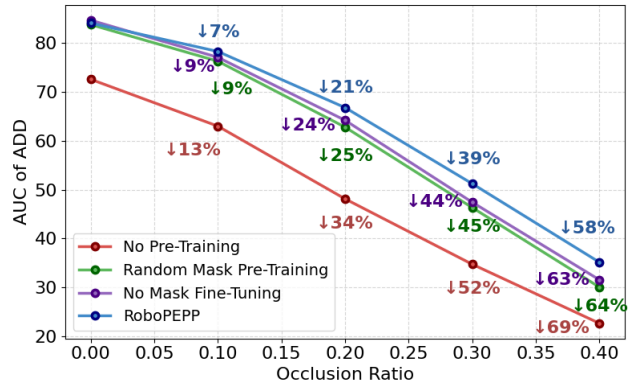Table A2. Number of images in each sequence of the dataset.

Figure A2. AUC comparison of the distance metric under varying occlusion levels, evaluated on the dataset in Sec. 4.2.3. Percentages next to the lines indicate the relative drop in each method's performance compared to their performance without occlusions.

solver to estimate the robot pose, is highly sensitive to keypoint detection errors. Even a single incorrectly detected keypoint can cause DREAM to fail in pose estimation, leading to high ADD.

A5.2. Ablation: Occlusion Robustness

In this section, we evaluate the methods from the *Embedding Predictive Pre-Training* ablation studies (Sec. 4.3) on the occlusion dataset described in Sec. 4.2.3. Specifically, we compare the following models: (1) a version of RoboPEPP without pre-training, (2) a version pre-trained with random masking instead of joint-specific masking, (3) the standard RoboPEPP (pre-trained with joint masking), and (4) a model pre-trained with joint masking but fine-tuned without masking during the encoder-predictor fine-tuning phase. As shown in Fig. A2, and similar to Fig. 6, we plot the AUC of the ADD metric against the occlusion ratio. Additionally, the percentage decrease in AUC relative to the performance without occlusion is annotated on the plot. Among the methods, RoboPEPP achieves the best performance across all occlusion ratios. While the framework with random-masking-based pre-training and the one fine-tuned without masking achieve performance comparable to RoboPEPP under zero occlusion, their performances degrade more rapidly as the occlusion ratio increases.

A6. Additional Qualitative Comparison

In this section, we provide additional examples of qualitative comparisons. Fig. A3 presents examples from the occlusion dataset discussed in Sec. 4.2.3. Fig. A4 shows comparisons on the Franka Photo dataset, while Fig. A5 highlights results on the real-world datasets Franka RS and AK. Lastly, Fig. A6 focuses on real-world images of the Franka robot collected in the lab under highly cluttered and oc-

cluded conditions. For all examples, comparisons are made against RoboPose [16] and HPE [5]. Rectangles are used to emphasize areas where these methods perform poorly, while RoboPEPP shows higher accuracy.
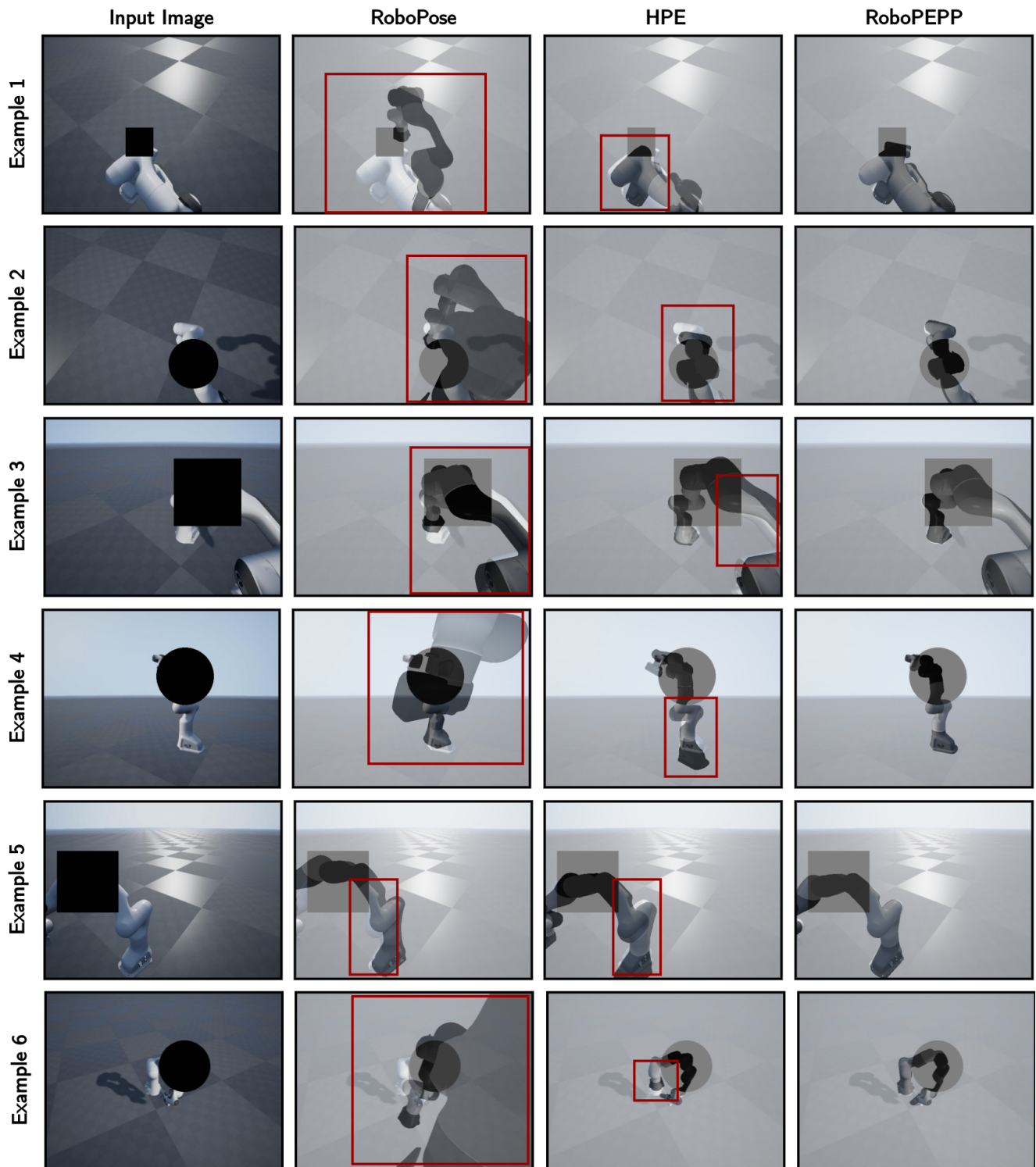
Figure A3. **Qualitative Comparison on Occlusion dataset**: Predicted poses and joint angles are used to generate a mesh overlaid on the original image, where closer alignment indicates greater accuracy. Highlighted rectangles indicate regions where other methods' meshes misalign, while RoboPEPP achieves high precision.
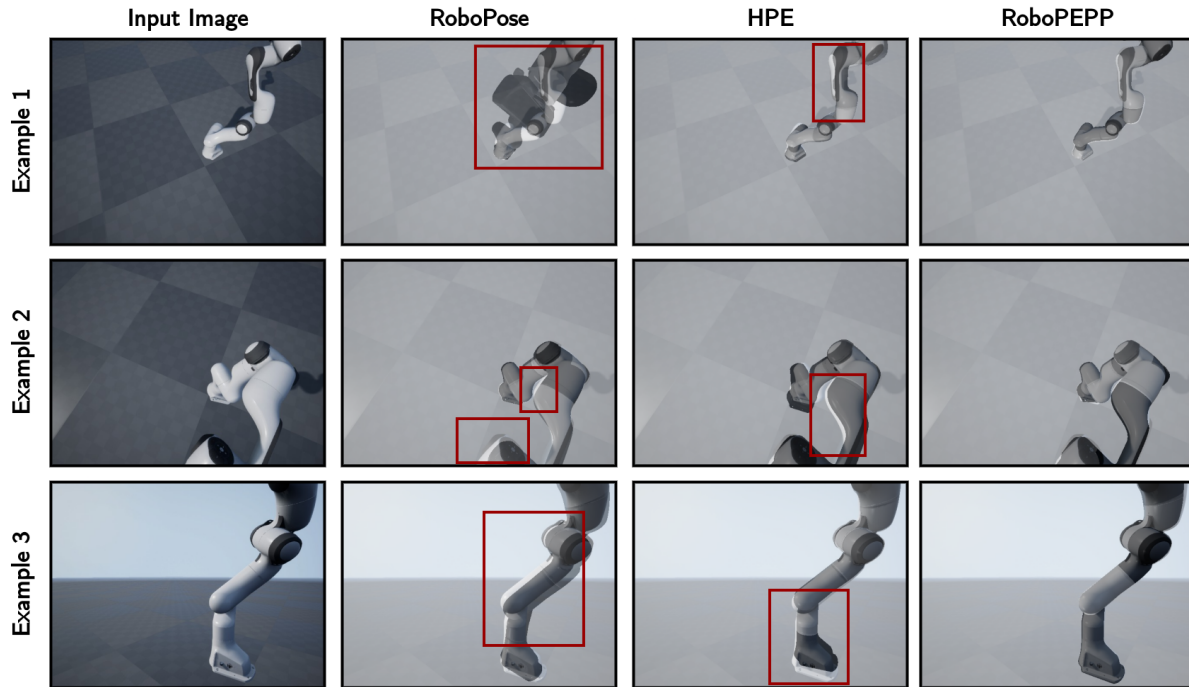
Figure A4. **Qualitative Comparison on Panda Photo dataset**: Predicted poses and joint angles are used to generate a mesh overlaid on the original image, where closer alignment indicates greater accuracy. Highlighted rectangles indicate regions where other methods' meshes misalign, while RoboPEPP achieves high precision.
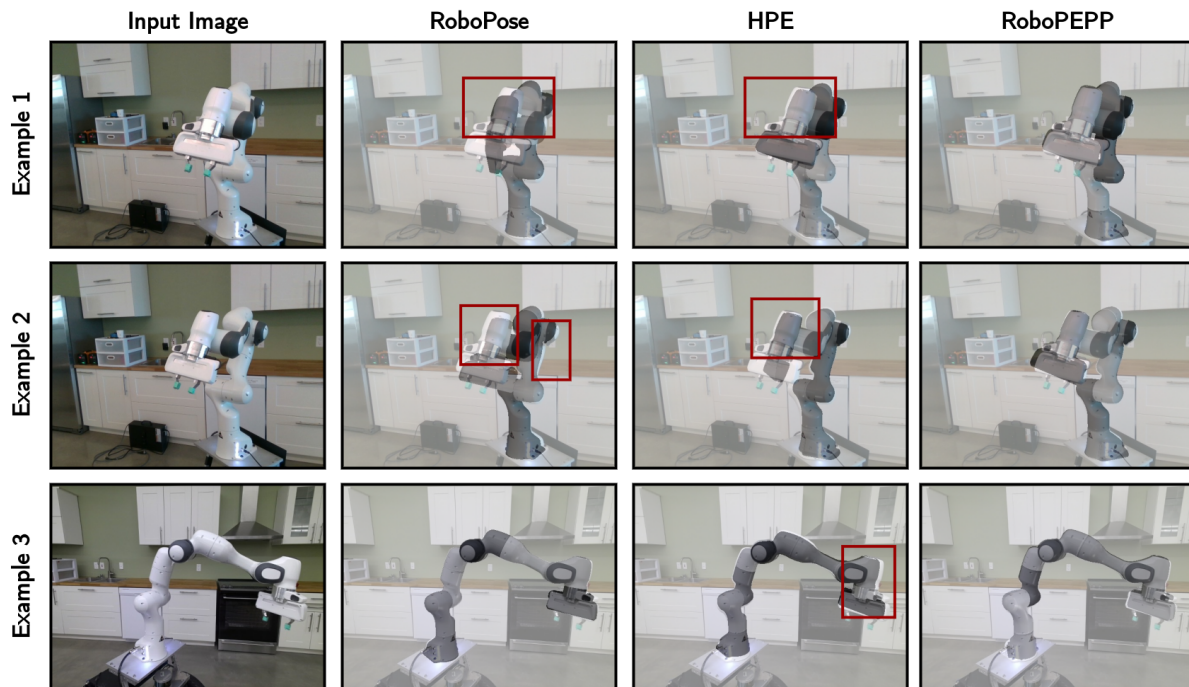

Figure A5. **Qualitative Comparison on Panda RS (Example 1 and 2) and Panda AK (Example 3) datasets**: Predicted poses and joint angles are used to generate a mesh overlaid on the original image, where closer alignment indicates greater accuracy. Highlighted rectangles indicate regions where other methods' meshes misalign, while RoboPEPP achieves high precision.
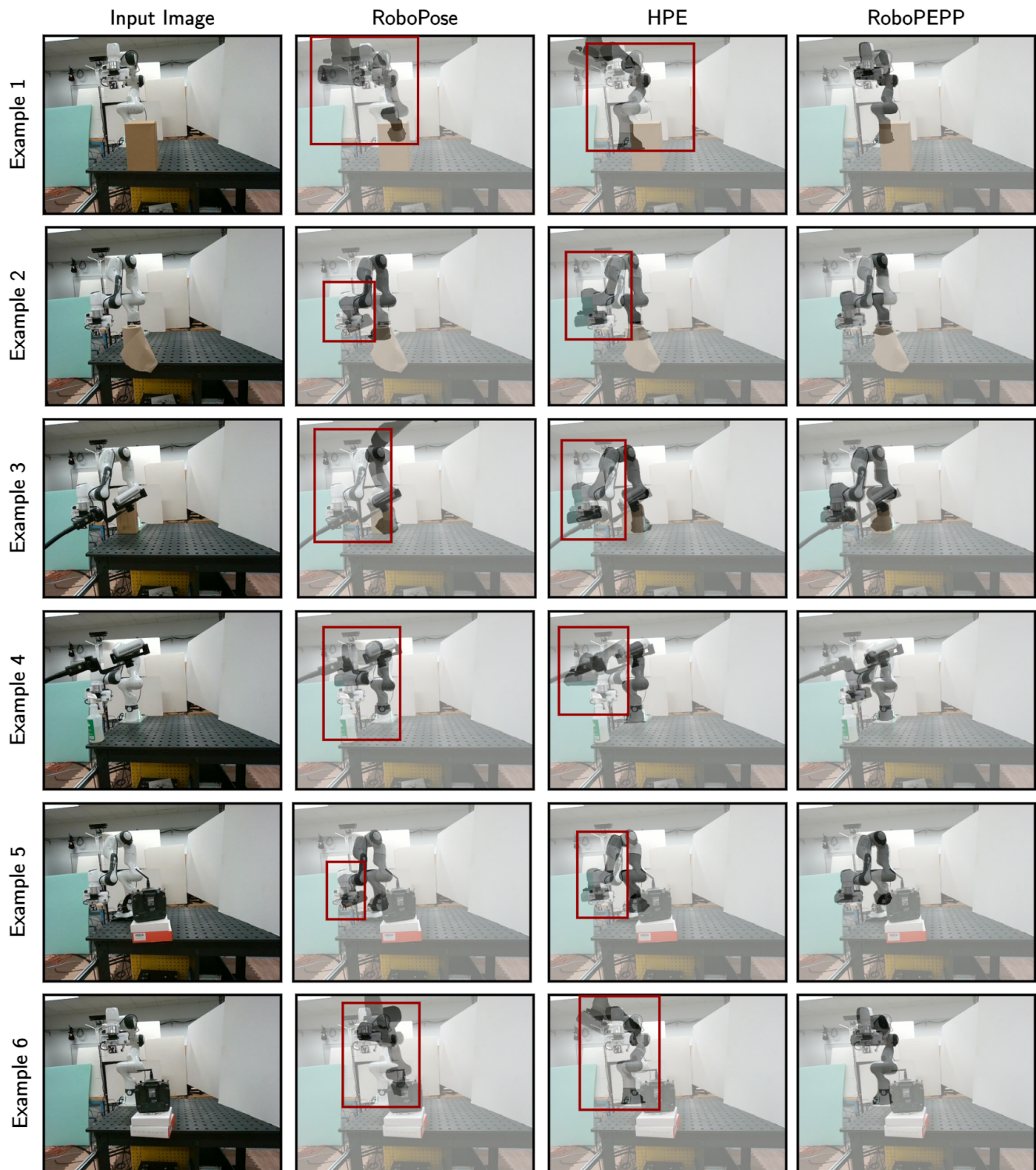
Figure A6. **Qualitative Comparison on Additional Real-World Images**: These images are collected in highly cluttered environments with robot occlusions. Predicted poses and joint angles generate a mesh overlaid on the original image, where closer alignment indicates greater accuracy. Highlighted rectangles indicate regions where other methods' meshes misalign, while RoboPEPP achieves high precision.