

Supplementary Material for RoboPEPP: Vision-Based Robot Pose and Joint Angle Estimation through Embedding Predictive Pre-Training

Raktim Gautam Goswami^{1*} Prashanth Krishnamurthy¹ Yann LeCun^{2,3} Farshad Khorrami¹

¹New York University Tandon School of Engineering

²New York University Courant Institute of Mathematical Sciences ³Meta-FAIR

A1. Encoder and Predictor Architectures

As described in Sec. 3.1, we use Vision Transformer (ViT) [S4] architectures for both the encoder and predictor, similar to [S1]. 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 [S6], layer normalization [S2], 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 $j \in \{1, \dots, L\}$.

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

$$L = M = \frac{224}{16} \times \frac{224}{16} = 14 \times 14 = 196. \quad (1)$$

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 $i \in \{1, \dots, M\}$.

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 $i \in \{1, \dots, M\}$. 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 [S1], 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 [S9] 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 [S9] 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 [S9] is used with learning rates

*Corresponding author: rgg9769@nyu.edu. This paper is supported in part by the Army Research Office under grant number W911NF-21-1-0155 and by the New York University Abu Dhabi (NYUAD) Center for Artificial Intelligence and Robotics, funded by Tamkeen under the NYUAD Research Institute Award CG010.

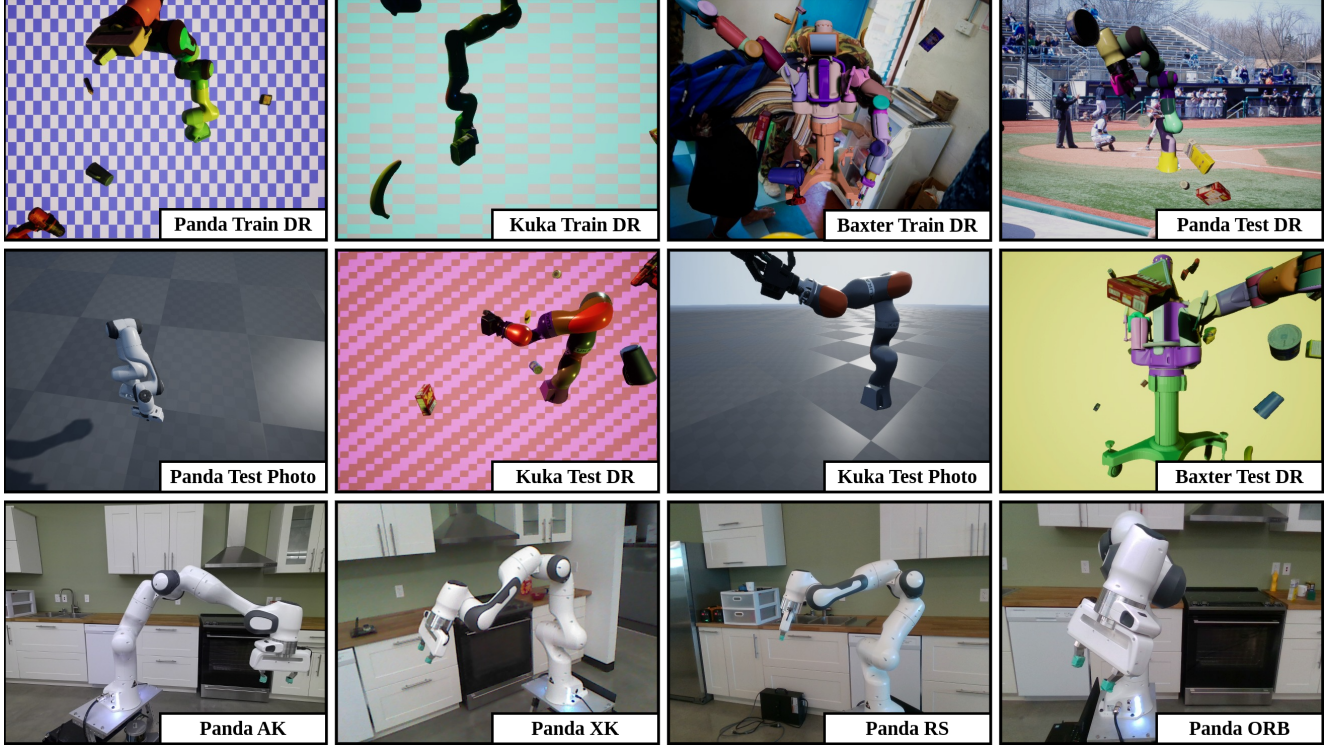


Figure A1. Example images from each of the training and test sequences from the DREAM dataset [S7].

of 10^{-7} for the encoder and predictor and 10^{-5} for the Joint Network. The learning rate for the Keypoint Network is set close to zero to prevent model collapse. We observed that prioritizing Joint Network updates over the Keypoint Network yielded the best results. On the Panda RS dataset, for example, using a learning rate of 10^{-10} for the Keypoint Network and 10^{-5} for the Joint Network improved ADD AUC from 70.4 to 80.5. In contrast, reversing or equalizing the rates resulted in lower performance (71.9 and 76.4, respectively), highlighting the robustness of the keypoints for guiding the Joint Network. Further, gradients originating from the Keypoint Network’s output are scaled by 10^{-10} to prevent conflicting updates to the encoder-predictor. These learning rates are all 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 [S8] 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 [S7], which includes sequences from three robots: Franka Emika Panda (Panda), Kuka iiwa7 (Kuka), and Rethink Robotics Baxter (Baxter). The dataset provides training and testing sequences in both synthetic and real-world settings, as detailed in Table A2. The synthetic data, created in Unreal Engine 4, 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 col-

	Known Joint Angles	Known Bounding Box	Real-World Sequences				Average
			Panda AK	Panda XK	Panda RS	Panda ORB	
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 [S3] 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.

	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.

lected 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 [S7], RoboPose [S5], HPE [S3], 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 [S7], which detects 2D keypoints and employs them in a PnP 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.

A5.3. Ablation: Joint Net and Keypoint Net

Fig. A3 and Table A3 analyze the Joint Net’s G value (number of refinement steps) and Keypoint Net’s hidden channel dimension, respectively, with $G=4$ and 256 channels yielding the best overall results.

Channel Dim.	AUC
128	69.3
256	75.5
512	74.4

Table A3. Average AUC on the Panda dataset for different values of the Keypoint Net’s hidden channel dimension.

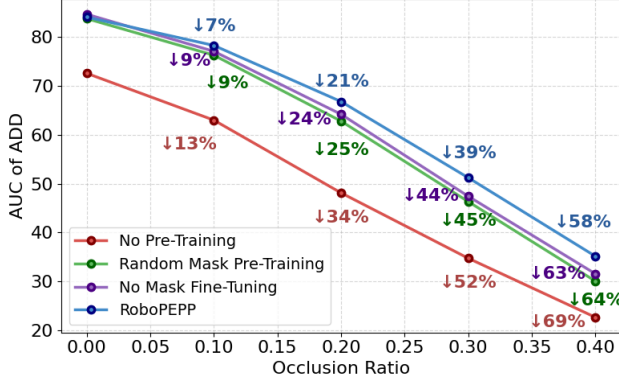


Figure A2. AUC comparison of the distance metric under varying occlusion levels, evaluated on the dataset in Sec. 4.2.3. Percentages next to the lines indicate the relative drop in each method’s performance compared to their performance without occlusions.

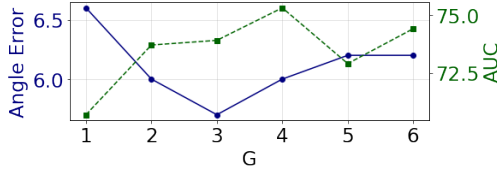


Figure A3. Average AUC and Joint Angle errors on the Panda dataset for different values of Joint Net’s G.

A6. Additional Qualitative Comparison

In this section, we provide additional examples of qualitative comparisons. Fig. A4 presents examples from the occlusion dataset discussed in Sec. 4.2.3. Fig. A5 shows comparisons on the Franka Photo dataset, while Fig. A6 highlights results on the real-world datasets Franka RS and AK. Lastly, Fig. A7 focuses on real-world images of the Franka robot collected in the lab under highly cluttered and occluded conditions. For all examples, comparisons are made against RoboPose [S5] and HPE [S3]. Rectangles are used to emphasize areas where these methods perform poorly, while RoboPEPP shows higher accuracy.

References

- [S1] 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, Vancouver, Canada, 2023. 1
- [S2] Jimmy Lei Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 1
- [S3] Shikun Ban, Juling Fan, Xiaoxuan Ma, Wentao Zhu, Yu Qiao, and Yizhou Wang. Real-time holistic robot pose es-

- timization with unknown states. In *European Conference on Computer Vision*, pages 1–17, Malmö, Sweden, 2024. 3, 4
- [S4] 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. 1
- [S5] 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. 3, 4
- [S6] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *Proceedings of the International Conference on Learning Representations*, 2022. 1
- [S7] 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, Paris, France, 2020. 2, 3
- [S8] 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. 2
- [S9] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 1

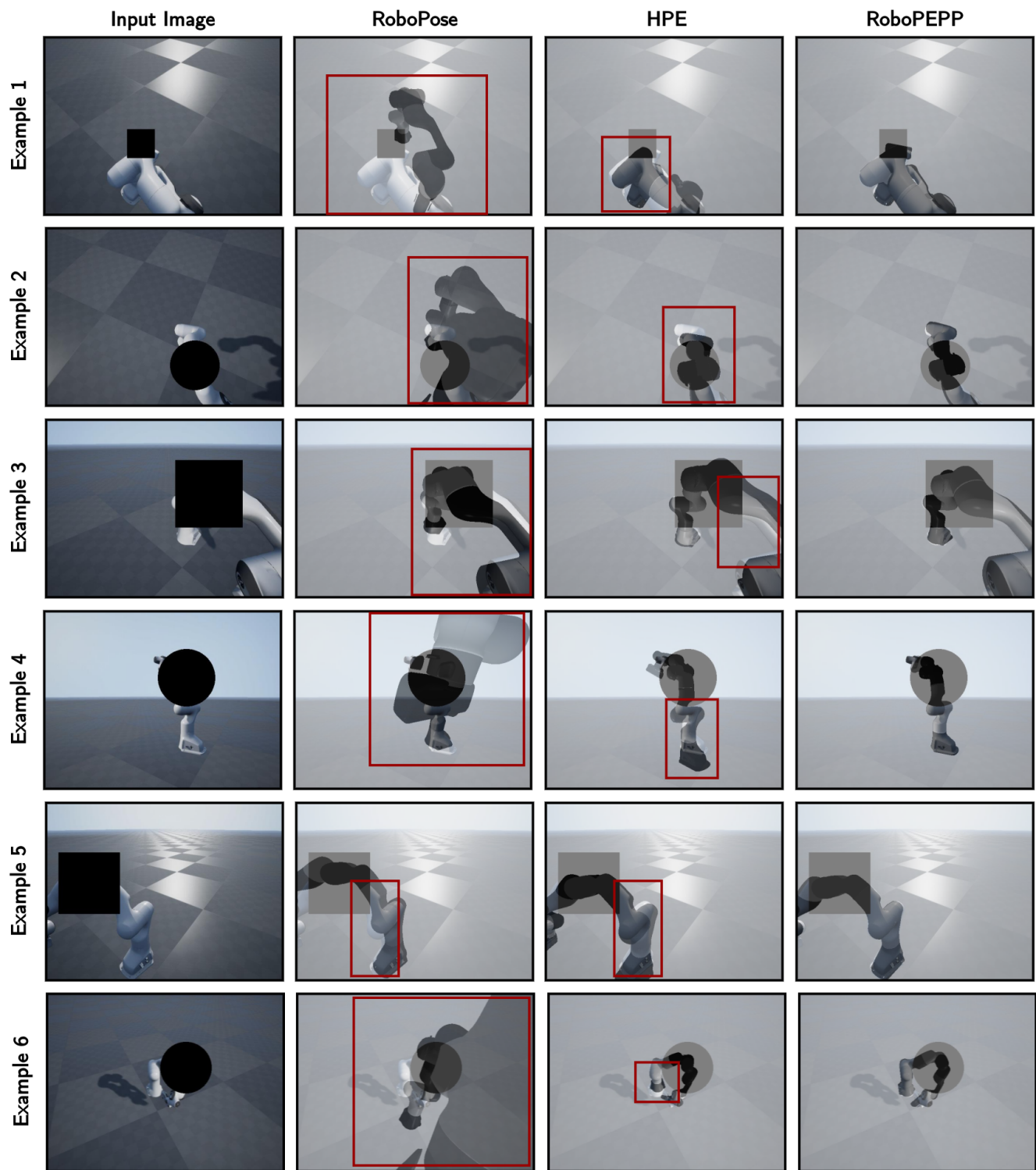


Figure A4. **Qualitative Comparison on Occlusion dataset:** Predicted poses and joint angles are used to generate a mesh overlaid on the original image, where closer alignment indicates greater accuracy. Highlighted rectangles indicate regions where other methods' meshes misalign, while RoboPEPP achieves high precision.

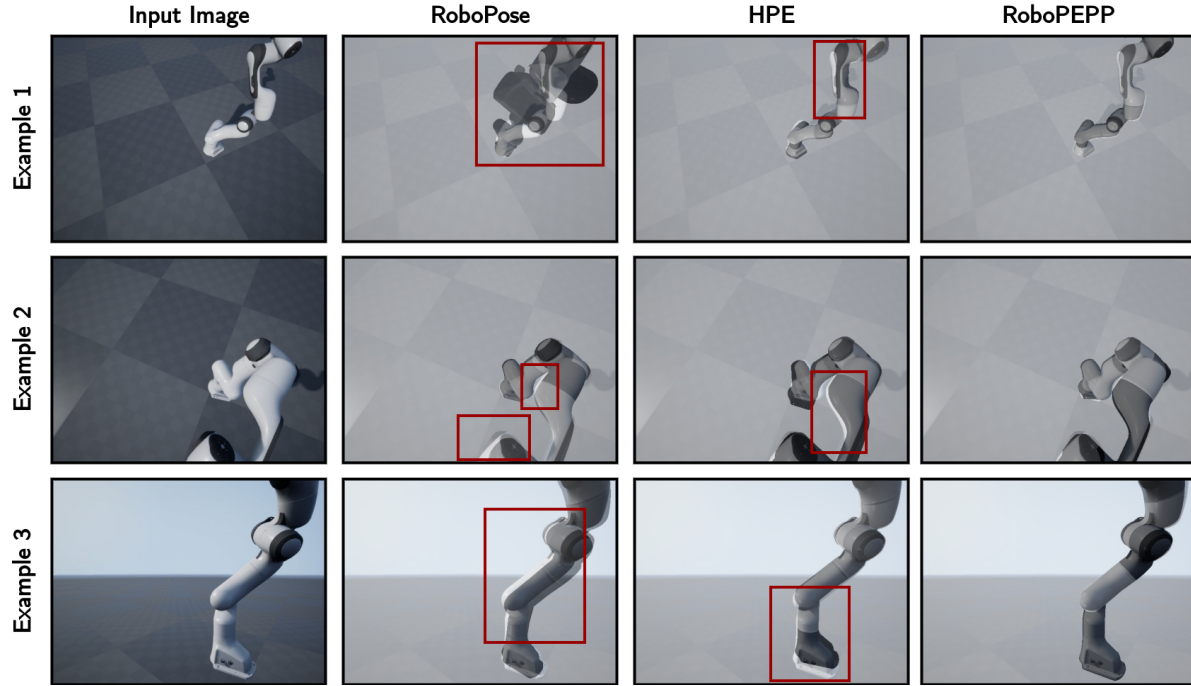


Figure A5. **Qualitative Comparison on Panda Photo dataset:** Predicted poses and joint angles are used to generate a mesh overlaid on the original image, where closer alignment indicates greater accuracy. Highlighted rectangles indicate regions where other methods' meshes misalign, while RoboPEPP achieves high precision.

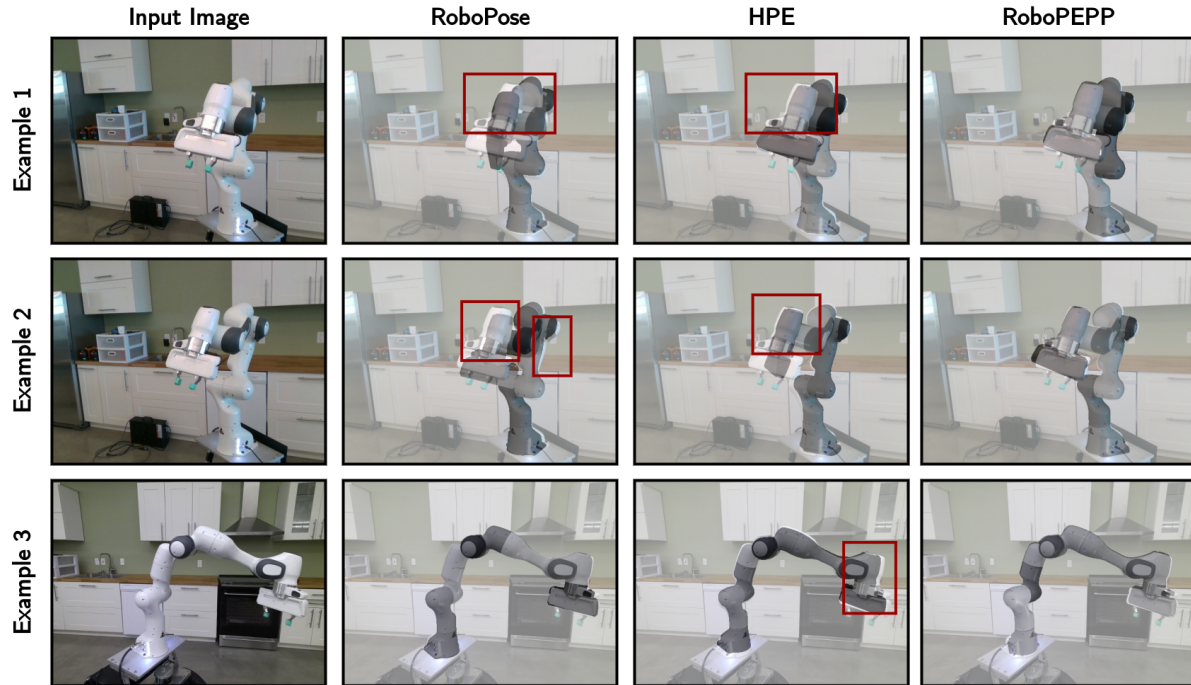


Figure A6. **Qualitative Comparison on Panda RS (Example 1 and 2) and Panda AK (Example 3) datasets:** Predicted poses and joint angles are used to generate a mesh overlaid on the original image, where closer alignment indicates greater accuracy. Highlighted rectangles indicate regions where other methods' meshes misalign, while RoboPEPP achieves high precision.

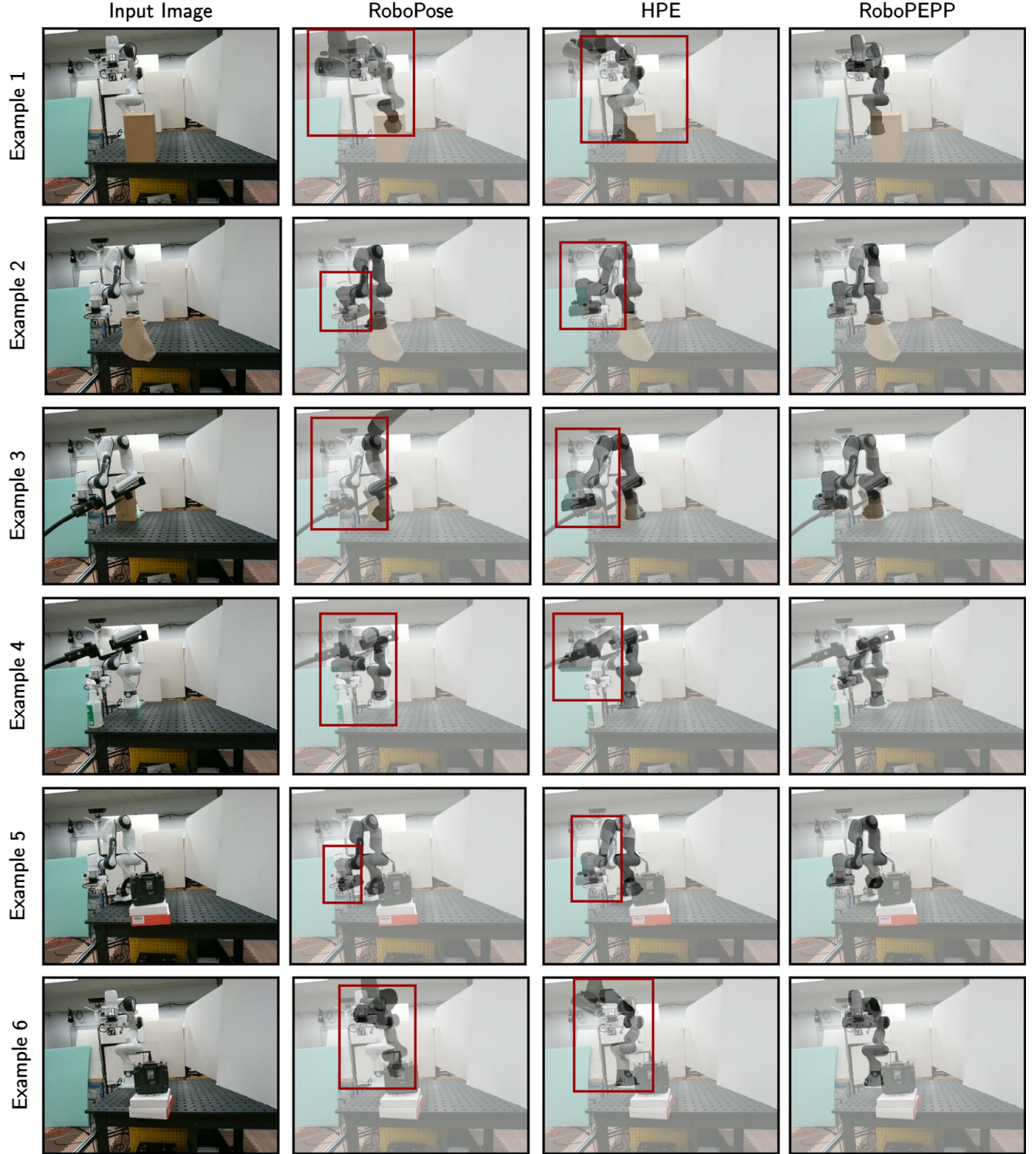


Figure A7. **Qualitative Comparison on Additional Real-World Images:** These images are collected in highly cluttered environments with robot occlusions. Predicted poses and joint angles generate a mesh overlaid on the original image, where closer alignment indicates greater accuracy. Highlighted rectangles indicate regions where other methods' meshes misalign, while RoboPEPP achieves high precision.