RoPE-Flow: Monocular Vision-based Robot Pose Estimation via 2D Graph-Conditioned 3D Flow Matching

Anonymous authors
Paper under double-blind review

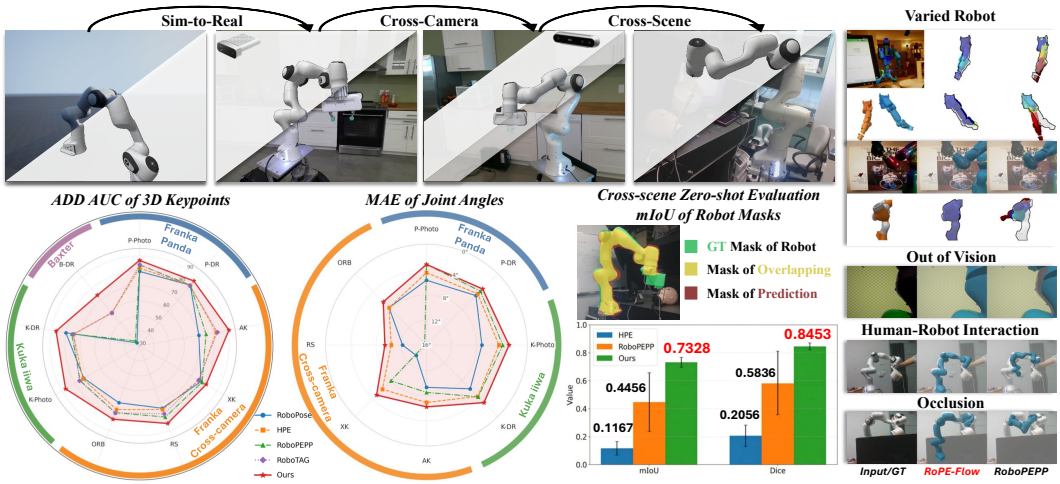

Figure 1: Overview of our work. RoPE-Flow can reconstruct **varied robot** pose **in diverse scenes** with **camera-frame 3D keypoints** and **joint angles** estimation from **monocular observation of diverse cameras** even with **human-robot interaction**, **occlusion** and **out-of-vision body**.

Abstract

Camera-to-robot pose estimation from a monocular image is significant for vision-based robot navigation and control. However, recovering 3D structural features from 2D observations remains a challenging task. Existing camera-to-robot pose estimation methods either rely on time-consuming iterative rendering or involve sophisticated 2D–3D registration which suffers from projection ambiguity, accumulated errors for distal joints and imbalanced learning for different joints. In this work, we propose **RoPE-Flow**, an end-to-end flow matching-based framework for camera-to-robot pose estimation. From the predicted 2D heatmaps, we propose a Keypoint-centered Feature Extractor to obtain positional features along with contextual representations from keypoints-centered multi-scale patches. Moreover, we introduce Graph-guided Condition Generator, which embeds a learnable adjacency map to generate condition for flow matching based on correlation among keypoints. Besides, we introduce Bone Length Loss and Ray-Depth Loss for topological consistency and scale stability, with joint-wise weighting to mitigate varied prediction difficulty across joints. The proposed method outperforms state-of-the-art methods for pose estimation across different types of robots with mutiple cameras. Besides, we establish a novel cross-scene benchmark to demonstrate the outstanding performance of RoPE-Flow on zero-shot cross-scene robot pose estimation. Meanwhile, animated visualizations are shown on the website: https://ropeflow.netlify.app/.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Accurate pose estimation of articulated robots from monocular RGB is a core capability for vision-based navigation (Bultmann et al., 2023), human–robot collaboration (Zhang et al., 2025), closed-loop manipulation (Liu et al., 2026b) and multi-robot collaboration (Yun et al., 2025). Unlike rigid object pose estimation, robot pose requires recovering the configuration of a kinematic chain whose links are self-occluding and appearance-ambiguous under changing viewpoints and lighting.

Conventional approaches perform camera-to-robot calibration using externally mounted fiducial markers such as ArUco Markers (Garrido-Jurado et al., 2014) and AprilTags (Olson, 2011). The marker's 2D image coordinates are extracted, while its corresponding 3D position on the robot is computed via forward kinematics. With a collection of across multiple robot configurations, the camera-to-robot transformation can be recovered by Perspective-n-Point (PnP) algorithms. After calibration, both the robot base and camera are assumed to remain stationary. Therefore, calibration can drift from minor impacts or repeated operation, alongside time-varying environmental conditions such as vibration.

To enable flexible camera-robot calibration for robot interaction in unstructured environments, render-based pipelines (Labbé et al., 2021; Zhu et al., 2025) perform iterative rendering supervised by segmentation masks to directly reconstruct the robot body. However, these methods rely on iterative refinement and require proper initialization for the optimization loop to converge (Li et al., 2018). To boost efficiency, end-to-end models (Lu et al., 2023; 2025; Goswami et al., 2025) detect 2D keypoints and recover 3D structure via Perspective-n-Point (PnP), combined with robot forward kinematics (FK) and known camera intrinsics. Still, these approaches derive 3D keypoints from 2D keypoints and joint angles using PnP, with limited robustness due to high sensitivity to correspondence noise. Some recent works (Liu et al., 2026a; Ban et al., 2024) adopt 3D models for 2D–3D registration to jointly optimize 2D and 3D representations. Even so, identical 2D projections can correspond to multiple depth configurations, introducing optimization and inference ambiguity. Generative methods offer a promising alternative by modeling the conditional distribution of 3D keypoints rather than predicting single deterministic keypoints estimate(Tian et al., 2024; Zhu et al., 2025). Moreover, most existing methods lack zero-shot cross-scene generalization evaluation on real robots under varied backgrounds, they only conduct self-supervised sim-to-real training and tests on single-background real data captured by different cameras from DREAM (Lee et al., 2020).

We propose RoPE-Flow, the first conditioned flow matching-based model for camera-to-robot pose estimation. Method based on flow matching can face two major challenges. **First**, directly predicting 3D keypoints without PnP and forward kinematics (FK) fails to enforce rigid structural constraints, kinematic priors, and inter-joint relationships. **Second**, the prediction difficulty varies across different robot joints. To tackle these issues, we make the following contributions:

1. We propose a **Keypoints-centered Feature Extractor** to learn multi-scale contextual representations at keypoint locations together with positional embeddings.
2. We propose **Graph-guided Condition Generator** , which provides the condition for 3D flow matching based on keypoint-centric features and the relationships among keypoints.
3. We introduce **Bone Length Loss** and **Ray-Depth Loss** during training to guarantee rigid structural consistency and scale stability, respectively. Meanwhile, **Joint-wise Weighting** is adopted in loss computation to balance the varied prediction difficulty over joints.
4. The proposed model outperforms state-of-the-art models across different types of robot with various backgrounds from DREAM dataset. We also construct a novel realistic benchmark for cross-scene robot 3D pose estimation, LabRoPE, to demonstrate outstanding **zero-shot generalization** of RoPE-Flow.

2 RELATED WORK

2.1 MARKER-BASED CAMERA-TO-ROBOT POSE ESTIMATION

The goal of Camera-to-Robot Pose Estimation is to estimate the transformation between a robot's end-effector and its vision system. Camera-to-Robot Pose Estimation describes the setup where the camera is stationary and positioned away from the robot (Mahler et al., 2019; Park et al., 2020), in

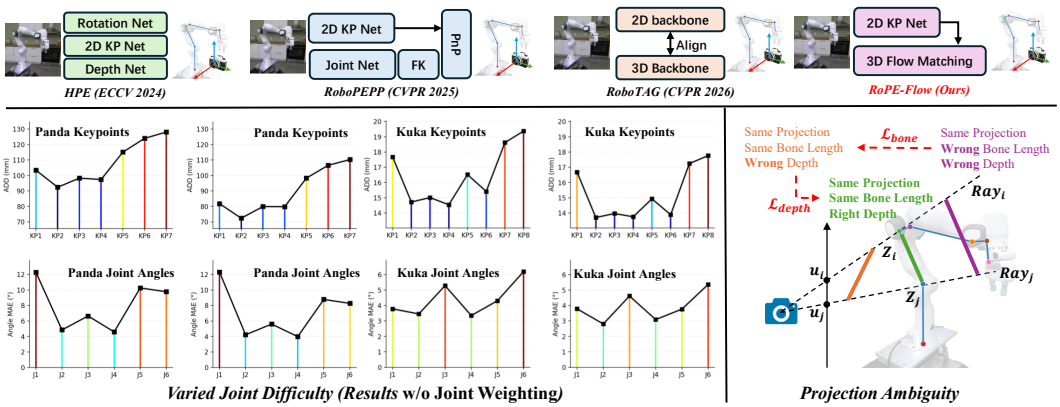

Figure 2: Comparison with previous methods for camera-to-robot estimation (Top Row). Evidence for *Varied Joint Difficulty* is shown based on distribution of prediction errors along the joints and keypoints. Problem for end-to-end 3D keypoints generation: *Projection Ambiguity*, as well as potential solutions in this work is also visualized.

contrast to eye-in-hand setting where the camera is mounted on the robot end-effector (Morrison et al., 2018). Traditional camera-to-robot pose estimation approaches attach fiducial markers to the end-effector (Garrido-Jurado et al., 2014; Olson, 2011), move the marker to multiple known poses, detect marker locations in captured images, and then solve for the camera-to-robot transformation. While widely applicable, such classic pipeline requires a full offline recalibration whenever the camera-robot relative pose shifts, which becomes cumbersome for real-world deployment.

2.2 Markerless Camera-to-Robot Pose Estimation

To enable flexible online camera-to-robot pose estimation for real-world deployment, DREAM (Lee et al., 2020) pioneers online markerless hand-eye calibration with a large-scale dataset. RoboPose (Labbé et al., 2021) uses iterative rendering supervised by alignment between projected robot models and segmentation masks for pose estimation. Though accurate, its iterative optimization causes heavy computational overhead and slow inference. To resolve this bottleneck, keypoint-based methods are proposed for efficient end-to-end pose estimation. CtRNet (Lu et al., 2023) and CtRNet-X (Lu et al., 2025) predict 2D keypoints and recover robot poses via PnP given known joint angles. SGTAPose (Tian et al., 2023) exploits temporal cues through a sequential framework. RoboPEPP (Goswami et al., 2025) adopts embedding predictive pretraining on masked-joint images to mitigate occlusions, while HPE (Ban et al., 2024) integrates a depth branch to leverage 3D priors. However, all these approaches require a predefined robot ROI before inference. Moreover, pose reconstruction based on PnP and forward kinematics is still sensitive to noise which leads to limited robustness. RoboTAG (Liu et al., 2026a) incorporates 3D priors via multi-way 2D–3D alignment, yet it still suffers from occlusions and projection ambiguity. In contrast, our RoPE-Flow directly predicts camera-frame 3D pose from full image and enables uncertainty modeling for occluded joints, dealing with problems from projection ambiguity for direct 3D keypoints generation and varied difficulties for prediction of each joints shown in Fig. 2.

3 Proposed Method

The whole pipeline of RoPE-Flow, shown in Fig. 3, mainly consists of a pretrained image encoder with two groups of LoRA adapters $LoRA_{kp}$ and $LoRA_q$, 2D heatmap decoders $HmDec_{KP}$ and $HmDec_{joint}$, Keypoints-centered Feature Extractor (KFE), Graph-guided Condition Generator (GCG), a Joint Head and a 3D conditional flow matching (3D CFM).

3.1 2D Keypoints Prediction

A pretrained DINOv3 ViT-L/16 produces a spatial feature map $\mathbf{F} \in \mathbb{R}^{C\times h\times w}$ (C=1024, h=w=16). The backbone remains frozen, while $LoRA_{kp}$ of rank r=32 and scale α=32 is injected into projec-

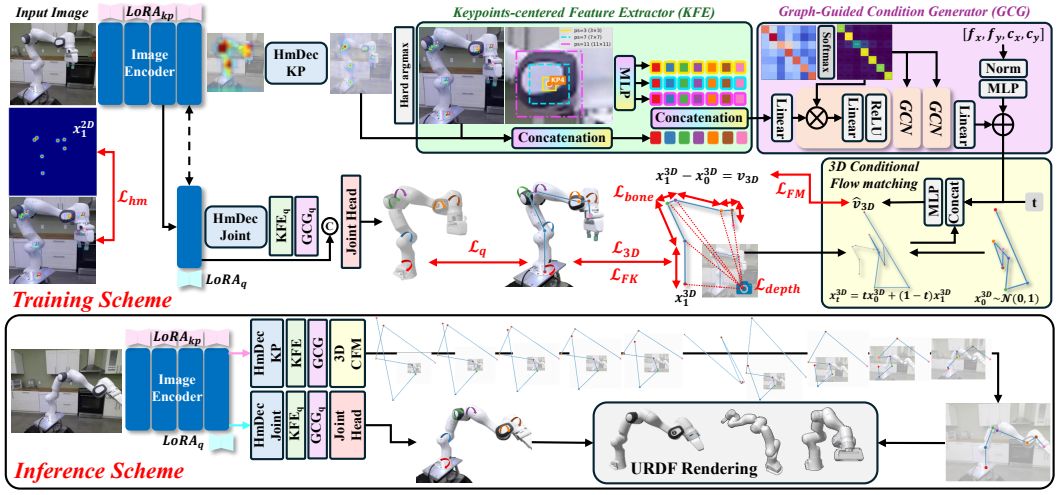

Figure 3: Pipeline of the proposed method.

tion layers of multi-head self-attention and MLPs of feed forward network. Spatial average pooling of $\mathbf{F}_{img}$ yields a global context $\mathbf{c}_{\text{img}} \in \mathbb{R}^C$ that conditions heatmap flow matching during training. Four transposed convolutions followed by a 1×1 convolution produce per-keypoint belief maps $\mathbf{H} = \text{HmDec}(\mathbf{F}_{img}) \in [0,1]^{K\times H\times W}$. Hard argmax on channel k gives a peak coordinate to denote a position of each keypoint $\mathbf{u}_k = (u_k, v_k) = \arg\max_{(u,v)} \mathbf{H}_k(u,v)$, while confidence of each keypoint also comes from the predicted heatmap $s_k = \mathbf{H}_k(u_k, v_k)$.

3.2 Keypoints-centered Feature Extractor

Around peak $\mathbf{u}_k$, we bilinearly sample patches $\mathbf{P}_{k,s}$ of sizes $S = \{3, 7, 11\}$ from $\mathbf{H}_k$. Each flattened patch is mapped by a two-layer MLP ϕ_s, then three embeddings are concatenated and fused by an MLP ψ:

$$\mathbf{p}_k = \psi\Big(\big[\phi_3(\mathbf{P}_{k,3});\ \phi_7(\mathbf{P}_{k,7});\ \phi_{11}(\mathbf{P}_{k,11})\big]\Big). \tag{1}$$

The final keypoint representation is $\mathbf{f}_k = [\mathbf{u}_k; s_k; \mathbf{p}_k]$, linearly embedded to a latent code $\mathbf{F}_k \in \mathbb{R}^{d_k}$.

3.3 Graph-guided Condition Generator

A shared adjacency $\mathbf{A} \in \mathbb{R}^{K\times K}$, initialized as $\mathbf{I}$, is processed by softmax along the row dimension. Given input feature map $\mathbf{F}_{in}$, one graph convolution provides the processed feature map $\mathbf{F}_{out}$ based on connections among keypoints $\mathbf{F}_{out} = \text{softmax}(\mathbf{A})\,\mathbf{F}_{in}\mathbf{W}$.

Gathering keypoints feature $\mathbf{F} \in \mathbb{R}^{K\times d_k}$ will be processed by three graph convolution layers and flattened, linearly projected to keypoint-based condition $\mathbf{c}_{kp} \in \mathbb{R}^{512}$. Meanwhile, normalized intrinsics $\boldsymbol{\kappa} = (f_x/W,\ f_y/H,\ c_x/W,\ c_y/H)$ are mapped by a two-layer MLP whose last layer is zero-initialized, then added residually to form the final condition code $\mathbf{c}$ for flow matching, as Eq. 2 shows.

$$\mathbf{c} = \mathbf{c}_{\text{gcn}} + \text{CamEnc}(\boldsymbol{\kappa}) \in \mathbb{R}^{512}. \tag{2}$$

3.4 3D Conditioned Flow Matching

Camera-frame 3D keypoints are generated under the graph condition $\mathbf{c}$ via conditional flow matching. Let $\mathbf{x}_1^{3D} = \mathbf{X}^{\text{gt}} \in \mathbb{R}^{K\times3}$ and $\mathbf{x}_0^{3D} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{K\times3})$, the interpolant $\mathbf{x}_t^{3D}$ and target velocity $\mathbf{x}_1^{3D} - \mathbf{x}_0^{3D}$ follow Eq. 3. A four-layer velocity MLP v_{3D} is utilized to predict velocity from concatenation of intepolant distribution, time token and condition $\big[\mathbf{x}_t^{3D};\ t;\ \mathbf{c}\big]$.

$$\mathbf{x}_t^{3D} = (1-t)\mathbf{x}_0^{3D} + t\mathbf{x}_1^{3D} \tag{3}$$

Sampling starts from standard Gaussian noise $\mathbf{x}_0^{3D} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{K\times 3})$ and applies N Euler steps to generate final predicted 3D Keypoints $\hat{\mathbf{X}} = \mathbf{x}_1^{3D}$, shown as Eq. 4

$$\mathbf{x}_{s+\Delta s}^{3D} = \mathbf{x}_s^{3D} + \Delta s \cdot v_{3D}(\mathbf{y}_s, s, \mathbf{c}), \quad \Delta s = 1/N \tag{4}$$

3.5 Joint Angle Prediction

A second adapter set LoRA_q is placed on the last 6 ViT blocks, together with a dedicated 2D keypoint decoder $HmDec_q$ and keypoint-based condition generator $\mathcal{F}_{cg}^q$. Different from 3D keypoints estimation, the condition for joint angle prediction is concatenation of pooled image feature $\mathbf{f}_{\mathrm{img}}^{(q)}$ and keypoints-aware condition $\mathbf{z} = \left[\mathbf{f}_{\mathrm{img}}^{(q)};\ \mathcal{F}_{cg}^q(HmDec_q(\mathbf{f}_{\mathrm{img}}^{(q)}))\right]$. Following RoboPEPP (Goswami et al., 2025), we use an iterative residual MLP g_ϕ as Joint Head to perform 4-step iteration of Eq. 5, starting from $\bar{\mathbf{q}}^{(0)} = \mathbf{0}$:

$$\bar{\mathbf{q}}^{(i+1)} = \bar{\mathbf{q}}^{(i)} + g_\phi\big([\mathbf{z};\ \bar{\mathbf{q}}^{(i)}]\big). \tag{5}$$

3.6 Training Losses

3.6.1 Training on Synthetic Data

Synthetic supervision has three training stages for 2D keypoints prediction, 3D keypoints prediction and joint angle prediction, respectively.

Stage I: 2D keypoint prediction. In stage I, only encoder-decoder architecture for 2D keypoints prediction is trained by $\mathcal{L}_{\mathrm{hm}}$. Ground-truth locations of 2D keypoints $\{u_k^{gt}, v_k^{gt}\}$ are rendered as isotropic Gaussians $\{\mathbf{H}_k^{\mathrm{gt}}\}$. Following Goswami et al. (2025), we further use CornerNet-style modified focal loss (Law & Deng, 2018) to supervise 2D keypoints prediction.

Stage II: 3D keypoints prediction. In Stage II, we keep training Encoder and heatmap decoder with small learning rate, while modules in keypoint-centered feature extractor, graph-guided condition generator and 3D flow matching are mainly optimized. We utilize **weighted joint-wise velocity loss** of Eq. 6 to train 3D flow matching.

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{t,\mathbf{y}_0} \sum_{k=1}^{K} w_k \big\| v_{\theta_2}(\mathbf{y}_t, t, \mathbf{c})_k - (\mathbf{y}_1 - \mathbf{y}_0)_k \big\|_2^2. \tag{6}$$

Geometric loss on the predicted 3D keypoints $\hat{\mathbf{X}}$ consists of a **weighted joint-wise 3D L1** (Eq. 7), **Bone-Length Loss** (Eq. 8), and FK alignment (Eq. 9).

$$\mathcal{L}_{\mathrm{3D}} = \sum_{k=1}^{K} w_k \big\| \hat{\mathbf{X}}_k - \mathbf{X}_k^{\mathrm{gt}} \big\|_1 \tag{7}$$

$$\mathcal{L}_{\mathrm{bone}} = \frac{1}{|\mathcal{E}|} \sum_{(i,j)\in\mathcal{E}} \Big| \|\hat{\mathbf{X}}_i - \hat{\mathbf{X}}_j\|_2 - \|\mathbf{X}_i^{\mathrm{gt}} - \mathbf{X}_j^{\mathrm{gt}}\|_2 \Big| \tag{8}$$

$$\mathcal{L}_{\mathrm{FK}} = \big\| \hat{\mathbf{X}}_r - \mathrm{FK}(\mathbf{q}^{\mathrm{gt}}) \big\|_1 \tag{9}$$

where $\hat{\mathbf{X}}_r = (\mathbf{R}^{\mathrm{gt}})^\top \big(\hat{\mathbf{X}} - \mathbf{t}^{\mathrm{gt}}\big)$, $\mathrm{FK}(\mathbf{q}^{\mathrm{gt}})$ denotes URDF positions of the seven supervised links and $\mathbf{T}_{c\leftarrow r}^{\mathrm{gt}}$ is the ground-truth extrinsics. The overall training objective for Stage II is formulated as:

$$\mathcal{L}_{\mathrm{II}} = \lambda'_{\mathrm{hm}}\,\mathcal{L}_{\mathrm{hm}} + \lambda_{\mathrm{FM}}\,\mathcal{L}_{\mathrm{FM}} + \lambda_{\mathrm{3D}}\,\mathcal{L}_{\mathrm{3D}} + \lambda_{\mathrm{bone}}\,\mathcal{L}_{\mathrm{bone}} + \lambda_{\mathrm{FK}}\,\mathcal{L}_{\mathrm{FK}}, \tag{10}$$

with $(\lambda'_{\mathrm{hm}}, \lambda_{\mathrm{FM}}, \lambda_{\mathrm{3D}}, \lambda_{\mathrm{bone}}, \lambda_{\mathrm{FK}}) = (0.2,\ 1,\ 2,\ 1,\ 1)$.

Stage III: joint-angle prediction head. The supervised loss of MSE in normalized joint space Eq. 11 is utilized to train the pathway for joint angles prediction, with frozen pathway for 3D keypoints prediction.

$$\mathcal{L}_{\mathrm{joint}} = \frac{1}{6} \sum_{j=1}^{6} \big(\bar{q}_j - \bar{q}_j^{\mathrm{gt}}\big)^2. \tag{11}$$

3.6.2 SIM-TO-REAL FINE-TUNING

We firstly finetune the pathway for 3D keypoints prediction except keypoints decoder on realistic data. 3D keypoints $\hat{\mathbf{X}}_{b,k}$ obtained from the model trained on synthetic data are mapped back to original-image pixels by $\hat{u}_{b,k} = \pi(\hat{\mathbf{X}}_{b,k}; \mathbf{K})$ with predicted 3D keypoints $\hat{\mathbf{X}}$. The reprojection residual of the sample b is $r_b = \frac{1}{K}\sum_{k=1}^{K} \left\|\hat{\mathbf{u}}_{b,k} - \mathbf{u}_{b,k}\right\|_1$, where $\mathbf{u}$ denotes heatmap peaks predicted by synthetic-domain model. Samples with $r_b < \tau_{\text{px}}$ (τ_{px}=100) form the inlier set $\mathcal{V}$, otherwise the lowest ρ=50% of finite residuals are kept. The Smooth-L1 reprojection loss is defined as:

$$\mathcal{L}_{\text{reproj}} = \frac{1}{|\mathcal{V}|}\sum_{b\in\mathcal{V}} \text{SmoothL1}\big(\pi(\hat{\mathbf{X}}_b; \mathbf{K}),\, \mathbf{u}_b\big). \tag{12}$$

We formulate bone length loss with adjacent supervised-link distances ℓ_{ij} at $\mathbf{q}=\mathbf{0}$ (pose-invariant):

$$\mathcal{L}^{\text{S2R}}_{\text{bone}} = \frac{1}{|\mathcal{E}|}\sum_{(i,j)\in\mathcal{E}} \Big|\|\hat{\mathbf{X}}_i - \hat{\mathbf{X}}_j\|_2 - \ell_{ij}\Big|. \tag{13}$$

For each keypoint we take the predicted depth $z_k = \hat{X}_{k,z}$ and lift the detached 2D peak $\mathbf{u}_k$ along the corresponding camera ray $\mathbf{d}_k = \mathbf{K}^{-1}[u_k, v_k, 1]^\top$, yielding a ray-consistent point $\tilde{\mathbf{X}}_k = z_k\mathbf{d}_k$. We then align prediction $\hat{\mathbf{X}}$ to this lift by Eq. 14 with $z_{\min}$=0.05 m.

$$\mathcal{L}_{\text{depth}} = \text{SmoothL1}\big(\hat{\mathbf{X}},\, \tilde{\mathbf{X}}\big) + \frac{1}{|\mathcal{E}|}\sum_{(i,j)\in\mathcal{E}} \Big|\|\tilde{\mathbf{X}}_i - \tilde{\mathbf{X}}_j\|_2 - \ell_{ij}\Big| + \frac{1}{K}\sum_{k} \text{ReLU}(z_{\min} - z_k), \tag{14}$$

The final sim-to-real training loss is formulated as Eq. 15, where $\lambda^{\text{S2R}}_{\text{bone}} = 20, \lambda_{\text{depth}} = 40$

$$\mathcal{L}^{kp}_{\text{S2R}} = \mathcal{L}_{\text{reproj}} + \lambda^{\text{S2R}}_{\text{bone}}\,\mathcal{L}^{\text{S2R}}_{\text{bone}} + \lambda_{\text{depth}}\,\mathcal{L}_{\text{depth}} \tag{15}$$

Additionally, we adapt pathway for joint angle prediction following self-supervised sim-to-real fine-tuning in Goswami et al. (2025), based on 2D keypoints reprojected from 3D keypoints generated by finetuned 3D keypoints prediction pathway.

4 EXPERIMENTS AND RESULTS

4.1 IMPLEMENTATION DETAILS

Following (Goswami et al., 2025; Liu et al., 2026a), we train and evaluate our method on the DREAM dataset (Lee et al., 2020). The model is separately trained on the synthetic DR split of Franka, Kuka and Baxter. Evaluation is conducted on DR and Photos splits for Panda and Kuka, and the DR split for Baxter. For Franka, we perform sim-to-real finetuning on Azure (AK) data using DR pretrained weights, then train and test on Kinect360 (XK), RealSense (RS) and ORB data based on weights for AK.

For synthetic training, we apply random 256×256 cropping, color jitter, Gaussian blur and horizontal flipping. Staged training runs 100, 300 and 300 epochs with batch size 32 for Stage I–III under the OneCycle schedule. Joint-wise weights are [1.0, 1.0, 1.0, 1.0, 1.5, 2.0, 2.0], [1.0, 1.0, 1.0, 1.0, 1.5, 2.0, 2.0, 2.5] and [1.0, 1.0, 1.0, 1.0, 1.0, 1.2, 1.2, 1.5, 1.5, 2.0, 2.0, 2.5, 2.5, 3.0, 3.0, 3.5, 3.5] for Panda, Kuka and Baxter, respectively. Sim-to-real finetuning takes 20 epochs with batch size 16 and no augmentation. All experiments are conducted on one NVIDIA H100 GPU.

4.2 CAMERA-FRAME POSE ESTIMATION

We compare RoPE-Flow with prior 3D keypoints prediction methods on the DREAM dataset, using the Area Under Curve (AUC) of Average Distance Distribution (ADD) as in previous works (Goswami et al., 2025; Liu et al., 2026a). Table 1 shows our method surpasses SOTA approaches on all splits across different robots and cameras in DREAM. We also adopt Mean Average Distance between predicted and ground-truth camera-frame joint positions for evaluation (Table 2). RoPE-Flow yields the most accurate joint positions without joint priors and achieves competitive results against methods using known joint states. Qualitative comparisons of 3D skeletons reconstructed from predicted keypoints are provided in the *Appendix* D.2.

Methods	Panda						Kuka		Baxter	Avg
	Photo	DR	AK	XK	RS	ORB	Photo	DR	DR	
DREAM-F*	79.5	81.3	68.9	24.4	76.1	61.9	/	/	/	/
DREAM-Q*	74.3	77.8	52.4	37.5	78.0	57.0	/	/	75.5	/
DREAM-H*	81.1	82.9	60.5	64.0	78.8	69.1	72.1	73.3	/	/
RoboPose†	79.7	82.9	70.4	77.6	74.3	70.4	73.2	80.2	32.7	71.3
HPE‡	82.0	82.7	82.2	76.0	75.2	75.2	73.9	75.1	58.8	75.7
HPE†	40.7	41.4	66.7	/	49.1	51.6	56.7	56.2	9.8	/
RoboPEPP†	84.1	83.0	75.3	78.5	80.5	77.5	76.1	76.2	34.4	74.0
RoboTAG	84.3	82.5	83.1	75.7	78.3	77.5	76.6	75.0	58.8	76.9
RoPE-Flow	87.1	86.6	90.9	81.6	85.1	82.1	87.3	86.9	74.1	84.6

Table 1: Comparison of robot pose estimation on DREAM dataset using AUC on the ADD metric. *DREAM and its variants need ground-truth joint state parameters. ‡ Results with ground-truth bounding box. † Results with predicted bounding box. **The Best** , **Second Best** and **Third Best** results are marked by corresponding colors. Curves of ADD can be found in *Appendix* D.1.

Methods	Panda						Kuka		Baxter
	Photo	DR	AK	XK	RS	ORB	Photo	DR	DR
DREAM-F	785.8	552.0	>1k	>1k	>1k	>1k	/	/	/
DREAM-Q	569.0	359.0	>1k	>1k	27.2	>1k	/	/	59.0
DREAM-H	518.8	325.6	56.5	>1k	23.6	>1k	>1k	>1k	/
RoboPose	21.5	**7.9**	24.4	**14.0**	23.1	19.4	33.5	19.2	**15.5**
RoboPose	21.4	32.2	34.3	22.3	26.0	30.1	89.0	34.2	87.9
HPE	18.0	18.4	18.9	24.0	24.8	24.8	29.0	28.0	45.0
RoboPEPP	29.8	49.5	29.0	22.0	23.0	27.0	72.4	84.2	157.5
Ours	**12.9**	13.8	**9.1**	18.6	**14.9**	**18.4**	**13.0**	**14.5**	30.6

Table 2: Comparison of Mean Average Distance (mm). **The best results** among all methods (including methods with given joint states) are in bold, while the second best one is underlined. **The Best** and **Second Best** results among methods *with no need of joint state parameters* are marked by corresponding colors.

4.3 Joint Angle Estimation

Methods	Panda		Kuka		Panda			
	Photo	DR	Photo	DR	AK	XK	RS	ORB
RoboPose	5.7	4.9	7.2	6.2	9.3	13.7	12.2	7.6
HPE	4.5	4.5	4.5	4.5	6.9	6.1	10.4	7.6
RoboPEPP	3.2	3.9	3.9	4.4	8.5	8.0	15.7	15.6
Ours	3.2	3.4	2.9	3.1	6.2	4.8	9.4	6.3

Table 3: Comparison of joint angle estimation results using average MAE (°) of joint angles. Detailed MAE on each joint can be found in *Appendix* D.3. **The Best** and **Second Best** results are marked by corresponding colors.

We compare our method with RoboPose, HPE and RoboPEPP for joint angle prediction on the Panda and Kuka splits of the DREAM dataset, using the mean average error (MAE) of all joint angles. Since the last Franka joint's rotation aligns with its corresponding 3D keypoint axis, we only compute errors for the first six joints (Goswami et al., 2025). Quantitative results are listed in Table 3, and per-joint details are given in the Appendix. Even though HPE and RoboPEPP rely on ground truth or preprocessed bounding boxes, RoPE-Flow achieves the best joint angle accuracy on both Franka and Kuka. We reconstruct robot 3D models from predicted joint angles and 3D keypoints using provided URDFs for qualitative evaluation. As shown in Fig. 4, our method yields more accurate robot models than prior SOTA approaches.

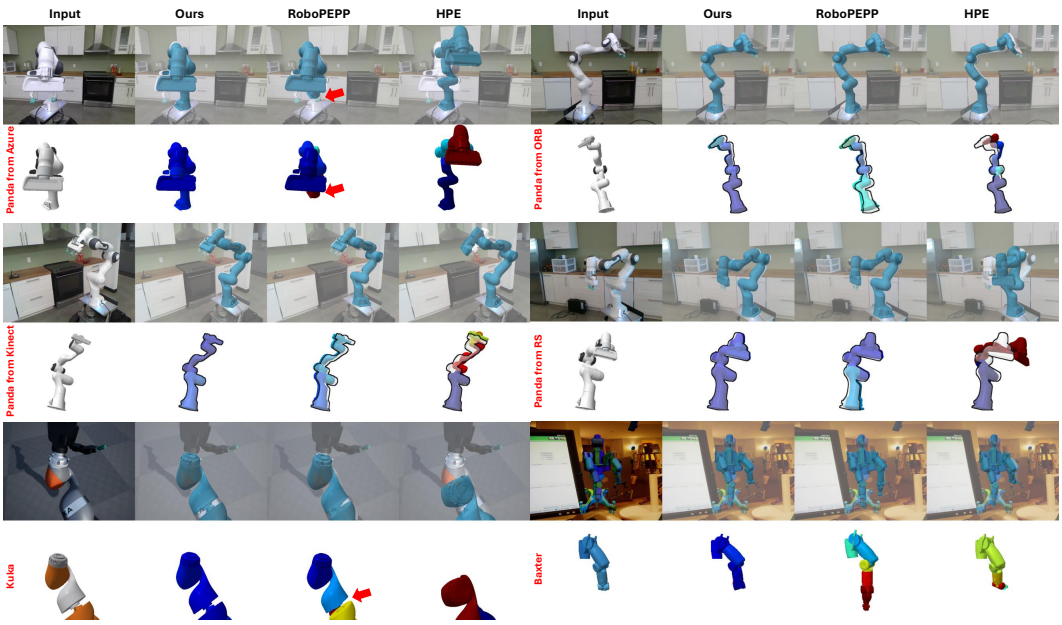

Figure 4: Qualitative comparison on URDF-based reconstructed models. More qualitative results can be found in *Appendix* D.4. **Results on videos** can be found in *Supplementary Material*.

4.4 Zero-shot Cross-scene Estimation

We build LabRoPE (details in Appendix C), a novel zero-shot benchmark containing 50 samples captured by RealSense from diverse views, to complement DREAM's realistic single-background scenes. Inspired by RoboPose (Labbé et al., 2021), we adopt mIoU and Dice coefficient between robot masks of input image and masks from reconstructed 3D models for zero-shot evaluation. We compare RoPE-Flow with two publicly available SOTA methods, RoboPEPP and HPE. Table 4 verifies our method's strong zero-shot generalization, and qualitative results are presented in Fig. 5.

Methods	mIoU	Dice
HPE	0.1167 ± 0.049	0.2056 ± 0.077
RoboPEPP	0.4456 ± 0.211	0.5836 ± 0.229
Ours	**0.7328 ± 0.036**	**0.8453 ± 0.024**

Table 4: Quantitative comparison on zero-shot cross-scene dataset.

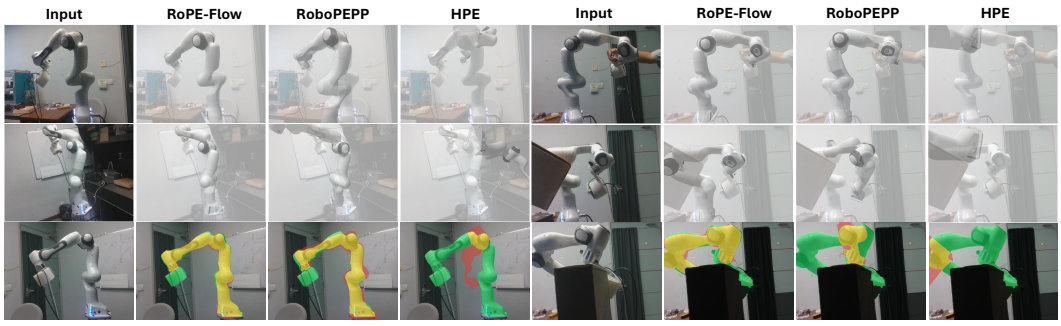

Figure 5: Qualitative comparison for zero-shot cross-scene samples. At last row, we use green mask to denote the ground truth, while yellow mask represents overlapping area and red mask is the mask of reconstructed robot. More examples can be found in *Appendix* D.5.

Ablations	Panda						Kuka		Baxter	Avg
	Photo	DR	AK	XK	RS	ORB	Photo	DR	DR	
Full Model	**87.1**	**86.6**	**90.9**	**81.6**	**85.1**	**82.1**	**87.3**	**86.9**	**74.1**	**84.6**
Keypoints-centered Feature Extraction										
3×3 Patch	86.0	85.8	89.3	81.4	84.6	76.4	86.8	86.2	70.8	83.0
7×7 Patch	86.6	86.3	90.2	79.4	84.6	79.4	86.9	86.5	69.9	83.3
11×11 Patch	87.0	86.7	89.0	75.5	77.2	79.7	87.6	87.2	74.2	82.7
w/o Patch	83.2	83.2	89.3	78.2	83.9	80.5	84.8	84.2	73.3	82.3
w/o Position	39.6	38.2	0.0	0.0	0.0	0.0	45.4	45.7	29.7	22.1
2D Graph-guided 3D Flow Matching										
w/o CFM	83.4	83.2	88.8	75.6	84.2	79.0	67.2	63.5	54.1	75.4
w/o GCN	84.1	83.8	88.6	76.0	85.0	78.9	83.5	83.1	72.7	81.7
Loss Function										
w/o $\mathcal{L}_{bone}$	87.0	86.5	88.9	66.1	83.0	80.1	87.2	86.6	74.0	82.2
w/o $\mathcal{L}_{FK}$	86.3	86.1	90.1	76.4	83.2	79.7	87.2	86.7	73.7	83.3
w/o $\mathcal{L}_{3D}$	60.8	54.9	50.1	34.2	4.5	42.8	56.5	53.7	66.9	47.2
w/o w_k	63.0	57.6	45.6	23.1	0.2	34.9	66.3	63.2	43.9	44.2

Table 5: Results of ablations on feature extraction, condition generation, flow matching and loss functions. **The Best** , **Second Best** and **Third Best** results are marked by corresponding colors.

4.5 Ablation Studies

We conduct ablation experiments on the Keypoints Feature Extraction (KFE) module using single patches of varying sizes (**N×N Patch**) instead of multi-scale patches. We also evaluate patch-wise contextual feature extraction (**w/o Patch**), position embedding (**w/o Position**), graph convolution (**w/o GCN**). Moreover, we replace the 3D CFM pathway with FK and PnP from RoboPEPP (Goswami et al., 2025) (**w/o CFM**) to validate the effect of flow matching. Besides, we ablate core loss components including the proposed **Bone Length Loss** $\mathcal{L}_{bone}$ and **Joint-wise Weighting** w_k.

Ablation results in Table 5 show larger single patches perform better on synthetic data while smaller patches favor sim-to-real fine-tuning. The full model with multi-scale patches and position embeddings achieves optimal overall performance. Graph-guided 3D flow matching outperforms the PnP-FK pipeline, and every loss component contributes substantially to the performance. Qualitative ablation results and flow matching step ablations are provided in *Appendix* E.3 and E.1, while ablation results for zero-shot estimation on LabRoPE dataset can be found in *Appendix* E.2.

Besides, different from RoboPEPP, we introduce **Bone Length Loss** $\mathcal{L}_{bone}^{S2R}$ and **Ray-depth Loss** $\mathcal{L}_{depth}$ for sim-to-real. Therefore, we implement ablation studies on these two losses for sim-to-real fine-tuning. Results in Table 6 demonstrates the positive effect of $\mathcal{L}_{bone}^{S2R}$ and $\mathcal{L}_{depth}$ for robot pose estimation on realistic data.

Ablations	AK	XK	RS	ORB	Avg
Full Model	**90.9**	**81.6**	**85.1**	**82.1**	**84.9**
w/o $\mathcal{L}_{bone}^{S2R}$	90.9	76.7	84.5	81.5	83.4
w/o $\mathcal{L}_{depth}$	82.5	70.1	76.5	62.0	72.8

Table 6: Results of ablations on Sim-to-Real Tuning.

5 Conclusion

In this work, we propose RoPE-Flow, the first flow matching-based approach for camera-to-robot pose estimation using monocular RGB images. Leveraging keypoint feature extraction from multi-scale keypoint-centered patches alongside graph-guided flow matching, RoPE-Flow outperforms existing methods by delivering highly accurate predictions of robot poses and joint angles. Furthermore, we introduce a novel benchmark for evaluating zero-shot generalization, which further validates the superior practical performance of RoPE-Flow compared with state-of-the-art (SOTA) approaches. Despite achieving outstanding results, our method still leaves room for future optimization, particularly regarding the prediction of self-rotating joints and distant joints, as well as inference efficiency.

AI USE STATEMENT

In this work, we used generative AI tools to polish writing of Abstract, Introduction and Related Work sections. Meanwhile, we used generative AI for related works retrieval to avoid missing references. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

For reproducibility, we have pre-released the source code on this anonymous GitHub link: https://anonymous.4open.science/r/RoPE-Flow-A4AB. Furthermore, we provide details of each module of RoPE-Flow in *Appendix* A and training details in *Appendix* B.

REFERENCES

Shikun Ban, Juling Fan, Xiaoxuan Ma, Wentao Zhu, Yu Qiao, and Yizhou Wang. Real-time holistic robot pose estimation with unknown states. In *European Conference on Computer Vision*, pp. 1–17. Springer, 2024.

Simon Bultmann, Raphael Memmesheimer, and Sven Behnke. External camera-based mobile robot pose estimation for collaborative perception with smart edge sensors. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8194–8200. IEEE, 2023.

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.

Raktim Gautam Goswami, Prashanth Krishnamurthy, Yann LeCun, and Farshad Khorrami. Robopepp: Vision-based robot pose and joint angle estimation through embedding predictive pre-training. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6930–6939. IEEE, 2025.

Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Single-view robot pose and joint angle estimation via render & compare. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1654–1663. IEEE, 2021.

Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 734–750, 2018.

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 *2020 IEEE international conference on robotics and automation (ICRA)*, pp. 9426–9432. IEEE, 2020.

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)*, pp. 683–698, 2018.

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. In *European conference on computer vision*, pp. 38–55. Springer, 2024.

Yifan Liu, Fangneng Zhan, Wanhua Li, Haowen Sun, Katerina Fragkiadaki, and Hanspeter Pfister. Robotag: End-to-end robot pose estimation via topological alignment graph. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35146–35155, 2026a.

Ze'an Liu, Xuanyin Wang, Ziyue Hao, and Jiaqi Sun. Eevs: Monocular end-effector pose estimation-based visual servoing against uncertainties. *Journal of Field Robotics*, 43(2):1033–1047, 2026b.

Jingpei Lu, Florian Richter, and Michael C Yip. Markerless camera-to-robot pose estimation via self-supervised sim-to-real transfer. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21296–21306. IEEE, 2023.

Jingpei Lu, Zekai Liang, Tristin Xie, Florian Richter, Shan Lin, Sainan Liu, and Michael C Yip. Ctrnet-x: Camera-to-robot pose estimation in real-world conditions using a single camera. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1914–1920. IEEE, 2025.

Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science robotics*, 4(26): eaau4984, 2019.

Douglas Morrison, Peter Corke, and Jürgen Leitner. Closing the loop for robotic grasping: a real-time, generative grasp synthesis approach. In *Robotics: Science and Systems*, 2018. doi: 10.15607/RSS.2018.XIV.021.

Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation*, pp. 3400–3407. IEEE, 2011.

Dongwon Park, Yonghyeok Seo, and Se Young Chun. Real-time, highly accurate robotic grasp detection using fully convolutional neural network with rotation ensemble module. In *2020 IEEE international conference on robotics and automation (ICRA)*, pp. 9397–9403. IEEE, 2020.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. In *International Conference on Learning Representations*, volume 2025, pp. 28085–28128, 2025.

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 *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8917–8926. IEEE, 2023.

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. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5375–5381. IEEE, 2024.

Yanpu Yun, Guohao Peng, Yichen Zhou, Jun Zhang, Yiyao Liu, Kaimin Mao, and Danwei Wang. Overlapping free: Anchorless uwb-assisted relative pose estimation for multi-robot systems. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12586–12592. IEEE, 2025.

Chuye Zhang, Xiaoxiong Zhang, Linfang Zheng, Wei Pan, and Wei Zhang. Generative visual foresight meets task-agnostic pose estimation in robotic table-top manipulation. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 2823–2846. PMLR, 27–30 Sep 2025.

Kangjian Zhu, Haobo Jiang, Yigong Zhang, Jianjun Qian, Jian Yang, and Jin Xie. Monose (3)-diffusion: A monocular se (3) diffusion framework for robust camera-to-robot pose estimation. *IEEE Robotics and Automation Letters*, 2025.

APPENDIX

A MODEL ARCHITECTURE DETAILS

We list the details of each layer of RoPE-Flow in Table 7. In detail, we subsequently list the upsampling layer in heatmap decoder (HmDec) along with the final projection layer, multi-scale keypoints-centered patch embedding, graph convolution layers for condition generation along with camera embedding and layers for final velocity prediction.

Layer Function	Detail	Ouput
Image Encoder Input: $B \times 3 \times 256 \times 256$		
Encoding	ViT-Large+LoRA	$B \times 1024 \times 16 \times 16$
Keypoints/Joint Decoder		
Upsample	ConvTranspose $1024 \to 256$ (k=4, s=2, p=1) BN+ReLU+Sigmoid	$B \times 256 \times 32 \times 32$
Upsample	ConvTranspose $256 \to 256$ (k=4, s=2, p=1) BN+ReLU+Sigmoid	$B \times 256 \times 64 \times 64$
Upsample	ConvTranspose $256 \to 256$ (k=4, s=2, p=1) BN+ReLU+Sigmoid	$B \times 256 \times 128 \times 128$
Upsample	ConvTranspose $256 \to 256$ (k=4, s=2, p=1) BN+ReLU+Sigmoid	$B \times 256 \times 256 \times 256$
Projection	Conv $256 \to K$ (k=1)+Sigmoid	$B \times K \times 256 \times 256$
Keypoints-centered Feature Extractor		
3×3 **Patch Embedding**	Linear $(3 \times 3) \to 32$ SiLU Linear $32 \to 32$	$B \times K \times 32$
7×7 **Patch Embedding**	Linear $(7 \times 7) \to 32$ SiLU Linear $32 \to 32$	$B \times K \times 32$
11×11 **Patch Embedding**	Linear $(11 \times 11) \to 32$ SiLU Linear $32 \to 32$	$B \times K \times 32$
Downsample	Linear $(32 \times 3) \to 32$ SiLU Linear $32 \to 32$	$B \times K \times 32$
Concatenation	concat$((x, y, \mathbf{H}_{(x,y)}) \in \mathbb{R}^{B \times K \times 3}, p \in \mathbb{R}^{B \times K \times 32})$	$B \times K \times 35$
Graph-guided Condition Generator		
Node Embedding	Linear $35 \to 64$+ReLU	$B \times K \times 64$
GCN	Linear $64 \to 512$+ReLU	$B \times K \times 512$
GCN	Linear $512 \to 512$+ReLU	$B \times K \times 512$
GCN	Linear $512 \to 512$+ReLU	$B \times K \times 512$
Camera Embdding	Linear $4 \to 512$ SiLU Linear $512 \to 512$	$B \times K \times 512$
Projection	Linear $(K \cdot 512) \to 512$	$B \times 512$
Conditional Flow Matching Input=$[\{(x, y, z)_i\}_{i=1,\ldots,7}, t, c] \in \mathbb{R}^{B \times 3 \times 7+1+512=534}$		
Velocity Pred	Linear $534 \to 512$+SiLU	$B \times 512$
Velocity Pred	Linear $512 \to 512$+SiLU	$B \times 512$
Velocity Pred	Linear $512 \to 512$+SiLU	$B \times 512$
Velocity Pred	Linear $512 \to (K \cdot 3)$	$B \times K \times 3$

Table 7: Details of each layer. k,s,p respectively denote kernel size, stride and padding of convolution layer. 'Linear $c_{in} \to c_{out}$' denotes Linear layer with input channel of c_{in}, while output channel is c_{out}. $\mathbf{H}$ is the predicted heatmap from Keypoints Decoder followed by hard-argmax, while x, y is the coordinate of each keypoint from the peak of $\mathbf{H}$.

B TRAINING DETAILS

We respectively provide details of training on synthetic data and sim-to-real finetuning in Table 8 and Table 9. In detail, we separately list settings of training epochs, optimizer, scheduler, trainable modules and learning rates.

Moreover, We perform validation every 5 epochs of synthetic training and save the best weight according to corresponding metrics (ADD AUC for Stage I/II, Joint MAE for Stage III). Each stage of training on synthetic data use the best weight from previous stage for initialization. We validate every epoch during sim-to-real finetuning and save the best weight according to corresponding metrics. At the beginning of sim-to-real finetuning for joint angle prediction, we use the best weight from sim-to-real finetuning of 3D keypoints prediction to initialize LoRA modules for the first 18 blocks of image encoder.

Item	Stage I	Stage II	Stage III
epochs	100	300	300
optimizer	AdamW		
weight decay	1×10^{-5}		
scheduler	OneCycle (final_div_factor=1×10^{-4})		
trainable	HmDec KP	$LoRA_{kp}$ + HmDec KP KFE+GCG+3D CFM	$LoRA_q$+HmDec Joint KFE_q+GCG_q+JointHead
peak lr for LoRA	0	1×10^{-4}	1×10^{-4}
peak lr for HmDec	1×10^{-4}	3×10^{-5}	0
peak lr for KFE+GCG+CFM	0	1×10^{-4}	0
peak lr for JH	0	0	1×10^{-4}

Table 8: Details of training on synthetic data. lr denotes learning rate.

Item	3D Keypoints Prediction	Joint Angle Prediction
epochs	20	
optimizer	AdamW	
weight decay	3×10^{-7}	
scheduler	OneCycle (pct_start=0.05; final_div_factor=1×10^{-4})	
trainable	$LoRA_{kp}$ KFE+GCG+3D CFM	$LoRA_q$ + $HmDec_q$ KFE_q+GCG_q+ JointHead
peak lr for LoRA	1×10^{-6}	
peak lr for HmDec	0	3×10^{-6} (AK/RS/ORB) 3×10^{-5} (XK)
peak lr for KFE+GCG+CFM	3×10^{-6} (AK/RS/ORB) 3×10^{-5} (XK)	0
peak lr for JH	0	3×10^{-6} (AK/RS/ORB) 3×10^{-5} (XK)

Table 9: Details of sim-to-real finetuning.

C DETAILS OF PROPOSED DATASET *LabRoPE*

To construct a dataset for zero-shot cross-scene robot pose estimation, we use Intel RealSense Camera to capture 50 images within a Franka Emika Panda robot from 5 different views with varied backgrounds in a laboratory. Moreover, to create a more complex situation, we replace the standard grasper on Franka with a DIY laser transmitter. Besides, we consider human-robot interaction and occlusion in the proposed dataset to further evaluate the robustness of methods. The gap between the proposed LabRoPE and existing dataset DREAM is shown in Fig. 6. We use GroundingDINO (Liu et al., 2024) and SAM2 (Ravi et al., 2025) to detect and segement the ground truth masks of the robot for zero-shot evaluation.

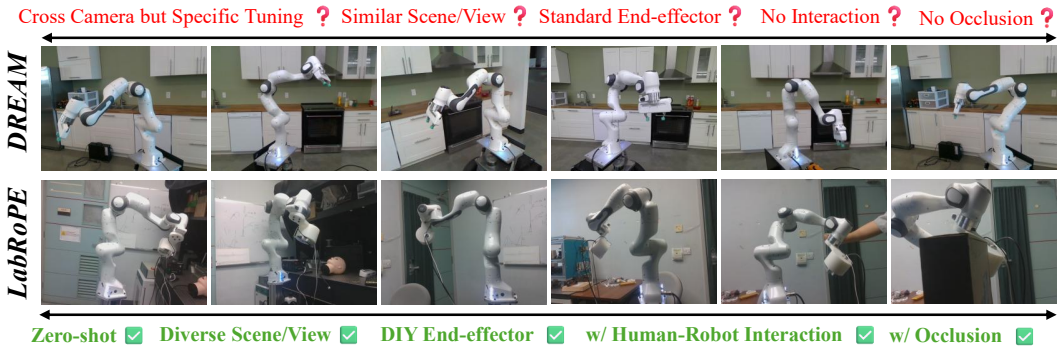

Figure 6: Comparison of LabRoPE and DREAM.

D Supplementary Comparison Results

D.1 Curves of ADD AUC

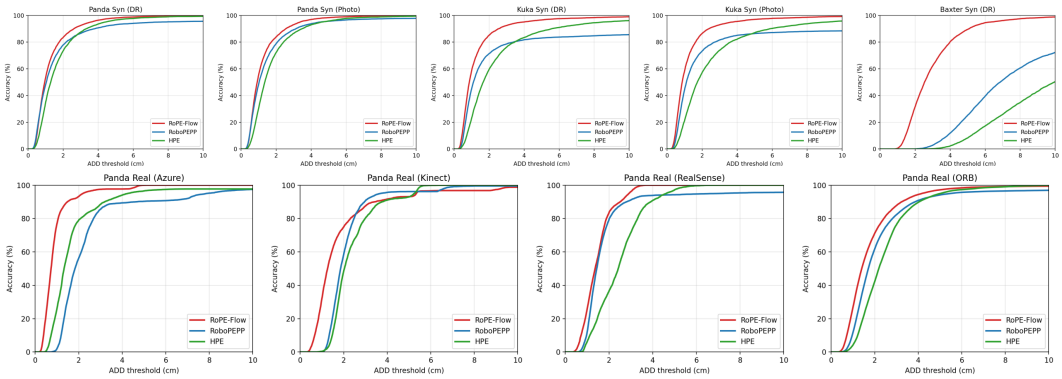

Figure 7: Curves of ADD.

With different thresholds τ, we calculate the proportion of predicted results with Average Distance (ADD) lower than threshold, named as Accuracy (ACC). Given such formulation $\mathrm{Acc}(\tau) = \frac{|\{i|ADD(\hat{X}_i, X_i)<\tau\}|}{N}$, we can draw Accuracy-Threshold $\mathrm{Acc} - \tau$ curves on each split of DREAM dataset, shown in Fig. 7.

D.2 Qualitative Comparison of 3D Keypoints Prediction

Based on 3D keypoints, we reconstruct the skeleton with keypoints and compare results from RoPE-Flow with results from RoboPEPP and HPE in Fig. 8, which can more intuitively demonstrate the outstanding performance of RoPE-Flow for 3D keypoints prediction.

D.3 Joint-wise Error for Joint Angles Prediction

Due to varied difficulties for prediction of different joints, we evaluate the mean average error on each joint of robot predicted by RoPE-Flow and SOTA competitors. The experimental results are shown in Table 10. We find that the proposed method may provide relatively inaccurate results on several joints of Franka, but still achieve best performance on average error and all joints prediction for Kuka robot.

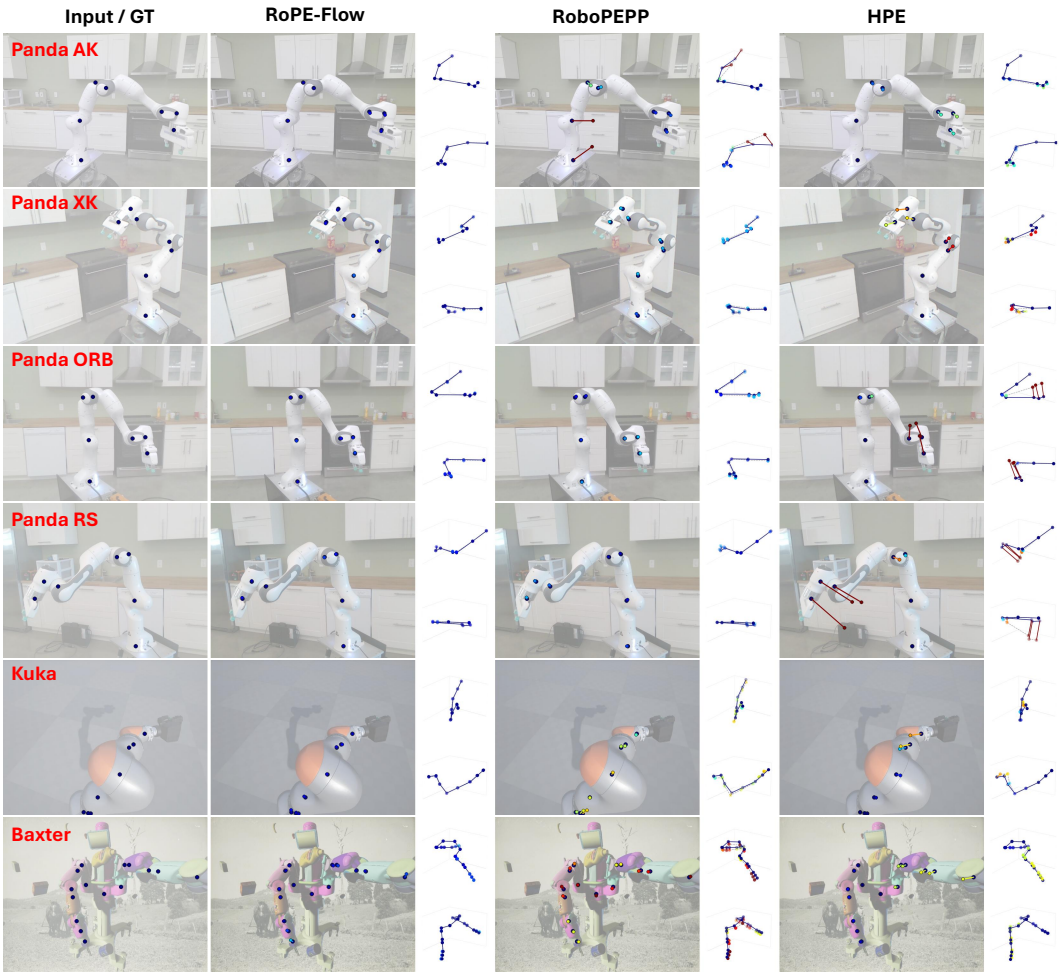

Figure 8: Qualitative Comparison of 3D Keypoints Prediction.

Comp	Franka													
	Photo							DR						
	J1	J2	J3	J4	J5	J6	Avg	J1	J2	J3	J4	J5	J6	Avg
Pose	7.7	3.5	4.3	3.4	7.3	8.1	5.7	6.1	2.7	3.6	2.5	6.3	8.1	4.9
HPE	6.1	2.2	3.6	2.0	6.2	6.6	4.5	6.2	2.2	3.9	1.9	5.9	6.6	4.5
PEPP	4.4	1.8	**2.2**	1.8	**4.4**	**4.8**	**3.2**	4.9	2.4	**2.9**	2.4	5.2	5.6	3.9
Ours	**3.2**	**1.7**	2.5	**1.6**	4.9	5.2	**3.2**	**3.6**	**1.7**	**2.9**	**1.7**	**4.9**	**5.5**	**3.4**
	AK							XK						
HPE	11.1	7.0	11.3	3.1	4.9	4.0	6.9	10.7	4.8	4.7	2.8	6.6	6.7	6.1
PEPP	29.0	**3.1**	**3.5**	**1.5**	6.0	8.0	8.5	31.2	4.8	**3.9**	**2.4**	4.1	**2.0**	8.0
Ours	**10.7**	8.1	12.0	**1.5**	**2.2**	**2.6**	**6.2**	**10.2**	**3.7**	4.5	3.1	**3.7**	3.4	**4.8**
	RS							ORB						
HPE	34.6	7.0	6.4	5.3	5.7	3.3	10.4	18.2	3.9	8.2	3.8	5.0	6.6	7.6
PEPP	77.6	**2.8**	**4.5**	**2.2**	3.8	3.0	15.7	77.6	**2.8**	**4.5**	**2.2**	3.8	3.0	15.6
Ours	**34.0**	6.2	7.4	3.9	**3.2**	**1.8**	**9.4**	**18.0**	3.9	5.9	3.3	**3.4**	**2.1**	**6.3**
Comp	**Kuka**													
	Photo							DR						
	J1	J2	J3	J4	J5	J6	Avg	J1	J2	J3	J4	J5	J6	Avg
Pose	4.9	5.1	6.7	6.0	10.8	9.6	7.2	4.4	2.8	5.4	3.4	12.5	8.5	6.2
HPE	4.8	3.8	5.0	2.8	4.9	5.9	4.5	4.6	3.6	4.9	2.8	5.2	6.1	4.5
PEPP	3.8	2.8	4.6	3.1	3.8	5.4	3.9	3.7	3.5	5.1	3.5	4.1	6.2	4.4
Ours	**3.1**	**1.7**	**3.4**	**1.8**	**3.2**	**4.6**	**2.9**	**3.1**	**1.8**	**3.5**	**1.8**	**3.5**	**4.8**	**3.1**

Table 10: Comparison of mean MAE (°) on each joint of robot.



Figure 9: Supplementary qualitative results from DREAM dataset.

D.4 Supplementary Examples from DREAM

We randomly select 5 samples from each split of DREAM dataset including Franka Panda DR, Photo, Azure(AK), Kinect360(XK), Realsense (RS), ORB, Kuka DR, Kuka Photo and Baxter DR. The supplementary qualitative results are shown in Fig. 9

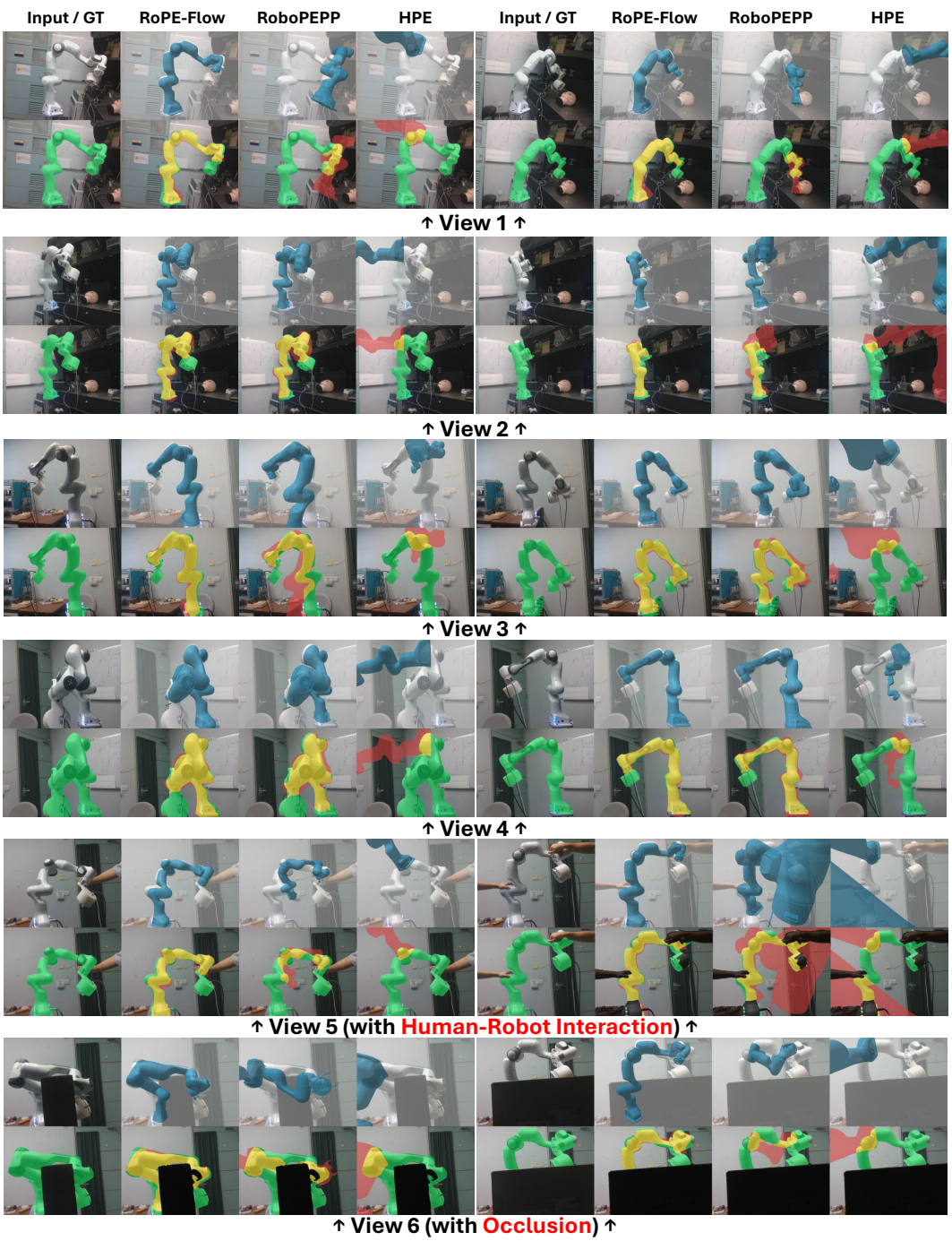

Figure 10: Supplementary qualitative results from LabRoPE dataset.

D.5 Supplementary Examples from LabRoPE

We randomly select 2 samples from each view of proposed LabRoPE dataset. The supplementary qualitative results are shown in Fig. 10. Specially, we also include some samples with **human-robot interaction** and **occlusion** in qualitative comparison.

E SUPPLEMENTARY RESULTS OF ABLATION STUDIES

E.1 QUALITATIVE COMPARISON FOR ABLATION STUDIES

Firstly, we compare some qualitative results of sim-to-real finetuning without Bone Length Loss or Ray-depth Loss in Fig. 11. We observe that the results without ray-depth loss are relatively accurate under the original camera view but degrade significantly when viewed from a different perspective. This verifies the projection ambiguity that motivates our work.

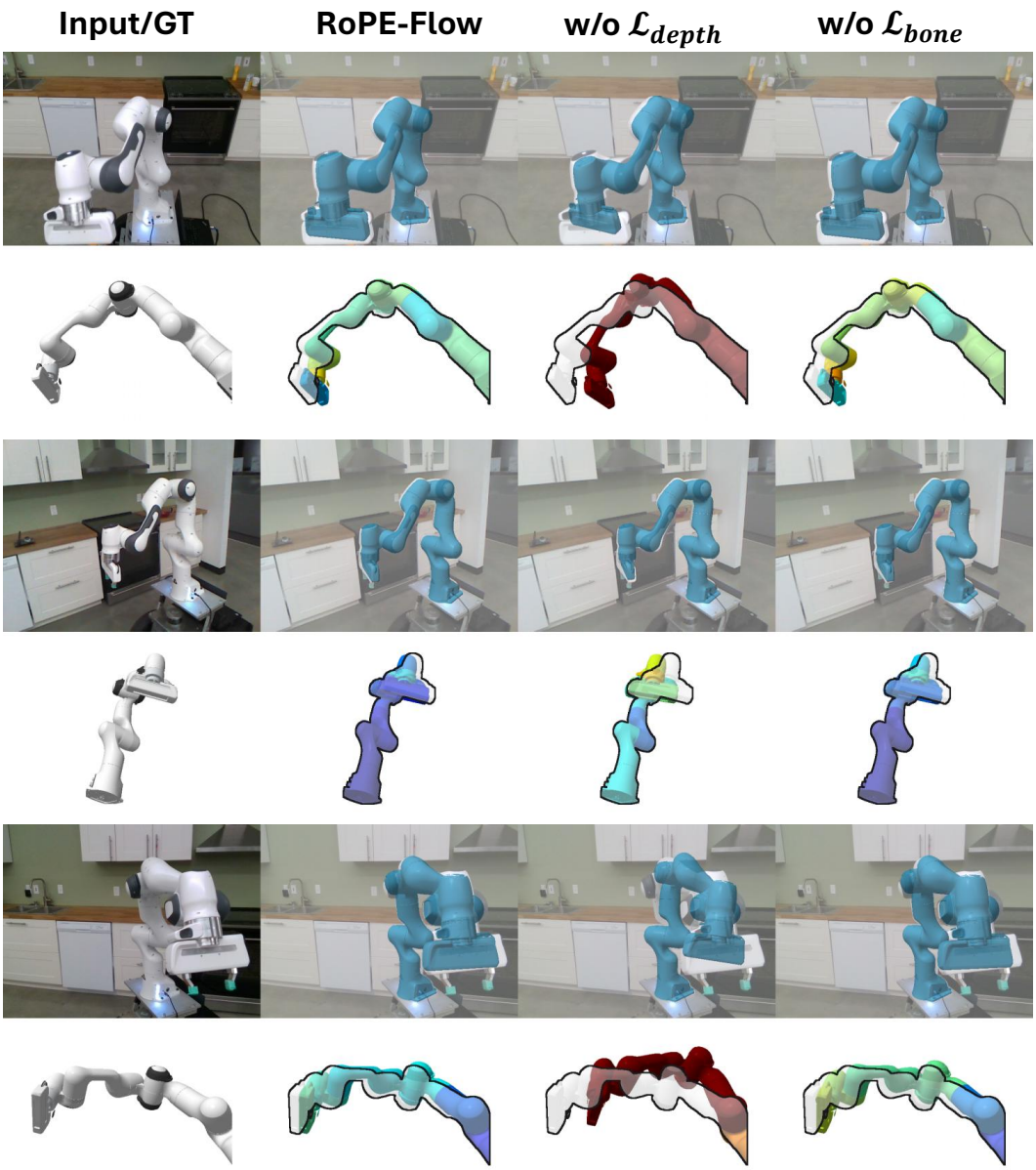

Figure 11: Qualitative results of sim-to-real finetuning with or without Bone Length Loss or Ray-depth Loss

Furthermore, we compare results from multi-scale patch embedding with single patch embedding in Fig. 12. As a results, full model with multi-scale patch-based keypoint-centered feature extraction can provide the most accurate pose estimation compared with model with single-scale patch embedding.

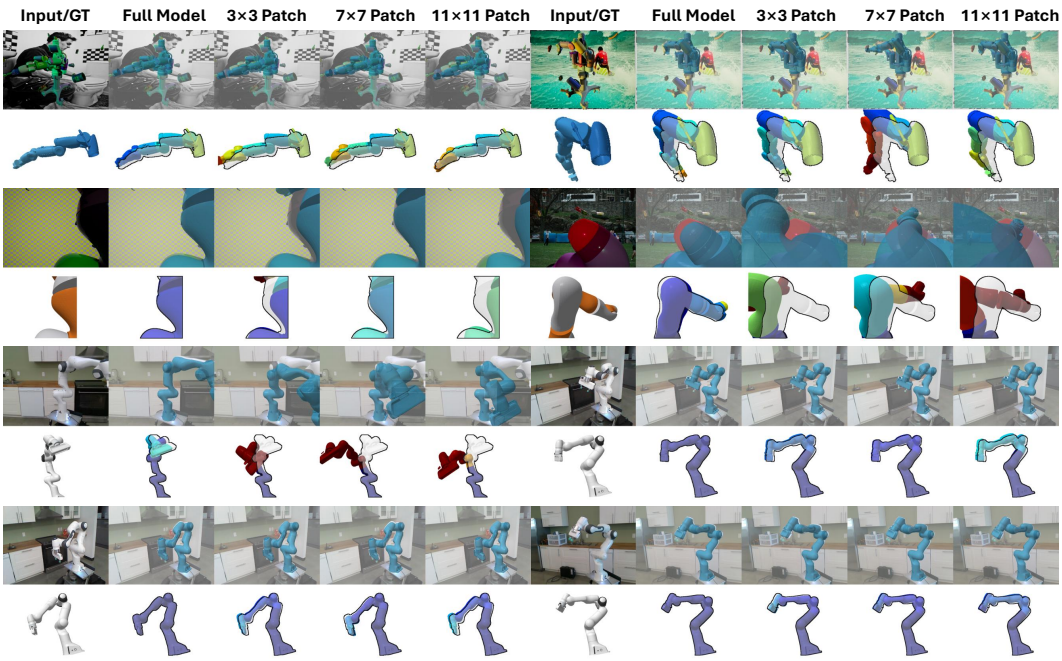

Figure 12: Results from multi-scale patch embedding with single patch embedding.

E.2 Zero-shot Ablation Studies on LabRoPE Dataset

We select ablation settings with best performance on split captured by RealSense in DREAM dataset for zero-shot ablation studies on LabRoPE dataset. The experimental results are shown in Table 11 and Fig. 13, demonstrating that ablated components indeed critically influence the generalization of the proposed RoPE-Flow.

Methods	mIoU	Dice
Full Model	**0.7328 ± 0.036**	**0.8453 ± 0.024**
3×3 Patch	0.3986 ± 0.066	0.5668 ± 0.067
7×7 Patch	0.4236 ± 0.067	0.5920 ± 0.067
11×11 Patch	0.3727 ± 0.064	0.5399 ± 0.068
No Graph	0.4064 ± 0.077	0.5736 ± 0.080
w/o $\mathcal{L}_{bone}$	0.3929 ± 0.087	0.5576 ± 0.105
w/o $\mathcal{L}_{depth}$	0.3996 ± 0.065	0.5679 ± 0.067
w/o $\mathcal{L}_{FK}$	0.4415 ± 0.094	0.6059 ± 0.103

Table 11: Ablation results of zero-shot pose estimation on LabRoPE dataset.

E.3 Ablation Studies on Flow Matching Steps

We set flow matching steps to 1/2/5/10/12/15/20, respectively implementing experiments on Panda and Kuka splits on DREAM dataset. Based on experimental results in Table 12, 10-step flow matching can provide the best results compared with other settings of steps.

F Runtime and Computational Cost

We calculate the inference speed and computational cost of the proposed method and compare with previous works in Table 13. Although with lower efficiency, RoPE-Flow still can achieve real-time requirement (around 30 FPS) on a relative low GPU memory (4.3GB).

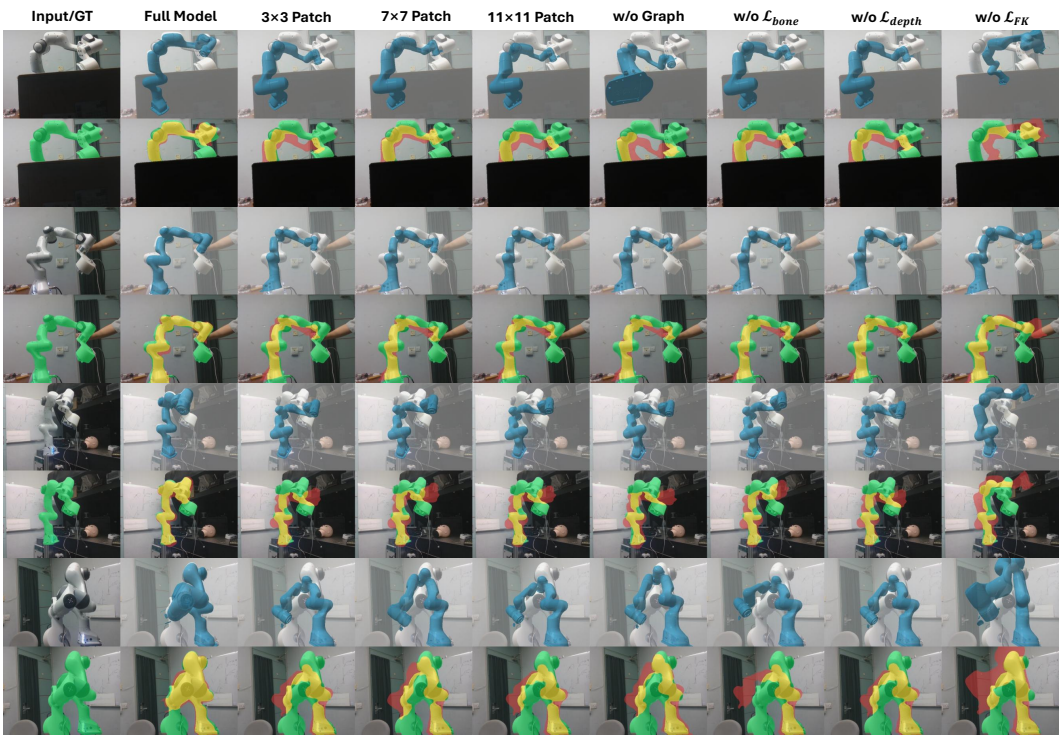

Figure 13: Qualitative results of ablation studies on LabRoPE dataset.

Data Splits	*Flow Matching Steps*						
	1	**2**	**5**	**10**	**12**	**15**	**20**
ADD AUC for 3D Keypoints Prediction							
Panda DR	29.02	74.29	83.68	**86.64**	86.53	86.00	85.66
Panda Photo	29.12	74.47	84.08	**87.09**	86.99	86.43	86.08
Kuka DR	28.86	75.31	83.87	**86.92**	86.87	86.58	86.28
Kuka Photo	28.89	75.66	84.34	**87.34**	87.30	87.04	86.73
Panda AK	1.42	51.00	83.91	**90.93**	90.65	89.71	89.02
Panda XK	0	0.04	63.72	**81.61**	80.21	78.18	76.34
Panda RS	0	0.23	69.98	**85.29**	82.91	79.80	76.85
Panda ORB	0	0.01	65.43	**82.13**	80.80	78.67	76.46
Mean Average Distance of Predicted 3D Keypoints							
Panda DR	77.21	26.11	16.72	**13.75**	13.88	14.42	14.73
Panda Photo	76.55	25.58	15.95	**12.93**	13.03	13.59	13.95
Kuka DR	76.99	25.64	17.04	**13.97**	14.05	14.35	14.65
Kuka Photo	76.90	25.06	16.37	**13.37**	13.43	13.68	14.00
Panda AK	140.77	48.99	16.08	**9.06**	9.34	10.28	10.97
Panda XK	486.69	151.55	36.74	**18.55**	19.93	21.95	23.78
Panda RS	405.94	132.62	30.01	**14.70**	17.08	20.19	23.14
Panda ORB	651.80	168.31	35.09	**18.38**	19.72	21.86	24.07

Table 12: Results of 3D Keypoints Prediction and Joint Angle Prediction on different data splits from DREAM dataset.

Methods	Latency (ms)	FPS	Mem (MB)	Params (M)	GFLOPs
RoPE-Flow	36.5	27.4	4330	813.38	206.4
RoboPEPP	19.0	52.6	2446	334.90	132.0
HPE	35.7	28.0	379	79.62	35.0

Table 13: Comparison of runtime and computational cost.