

EPro-PnP Network Analysis & Optimization

Yixuan Wu¹ and Chichun Ho¹

National Taiwan University, Taipei, Taiwan
{R14944009, B11902126}@ntu.edu.tw

Abstract. This report presents a comprehensive analysis of the End-to-End Probabilistic Perspective-n-Points (EPro-PnP) framework by benchmarking different feature extraction backbones. We replace the original ResNet baseline with three modern architectures: Swin Transformer, ConvNeXt, and HRNet, to evaluate their impact on 6DoF object pose estimation. Our experiments on the LineMOD dataset demonstrate that high-resolution features are critical for precise pose estimation. Specifically, HRNet achieves the state-of-the-art performance among the compared models, particularly under strict evaluation metrics (e.g., ADD 0.02d), validating the importance of maintaining spatial resolution in dense correspondence tasks.

Keywords: 6D Pose Estimation · EPro-PnP · Swin Transformer · ConvNeXt · HRNet.

1 Introduction

Estimating the 6D pose (rotation and translation) of an object from a single RGB image is a fundamental challenge in computer vision. Traditional two-stage methods, which first detect 2D keypoints and then solve the Perspective-n-Point (PnP) problem, suffer from non-differentiability, preventing end-to-end optimization. The EPro-PnP [1] framework addresses this by transforming the deterministic PnP problem into a probabilistic layer, enabling the gradient to flow through the PnP solver. In this project, our objective is to explore the potential of this framework by replacing its standard ResNet backbone with more advanced architectures. We aim to determine whether modern backbones capable of capturing global context (Swin Transformer [2]), balancing accuracy with efficiency (ConvNeXt [3]), or maintaining high-resolution representations (HRNet [4]) can further enhance pose estimation accuracy.

To this end, we present a modified PyTorch [5] implementation based on the official EPro-PnP repository¹. This project extends the original work by benchmarking these various modern backbones to evaluate their effectiveness in high-precision pose estimation. Our complete source code and modifications are publicly available at our project repository².

¹ Official implementation: <https://github.com/tjiiiv-cprg/EPro-PnP>

² Our implementation: https://github.com/Kiri487/2025-Fall-3DCV-Final_Project

2 Methodology

2.1 EPro-PnP Framework

Our work builds upon the EPro-PnP framework proposed by Chen et al. [1]. As illustrated in Fig. 1, the framework utilizes a backbone network to predict dense 2D-3D correspondences, consisting of 3D coordinates (x^{3D}) and importance weights (w^{2D}) for each pixel. These predictions are then processed by the EPro-PnP layer—a probabilistic derivative of the PnP solver—to output a pose distribution. This distribution is optimized end-to-end via KL Divergence loss against the ground truth.

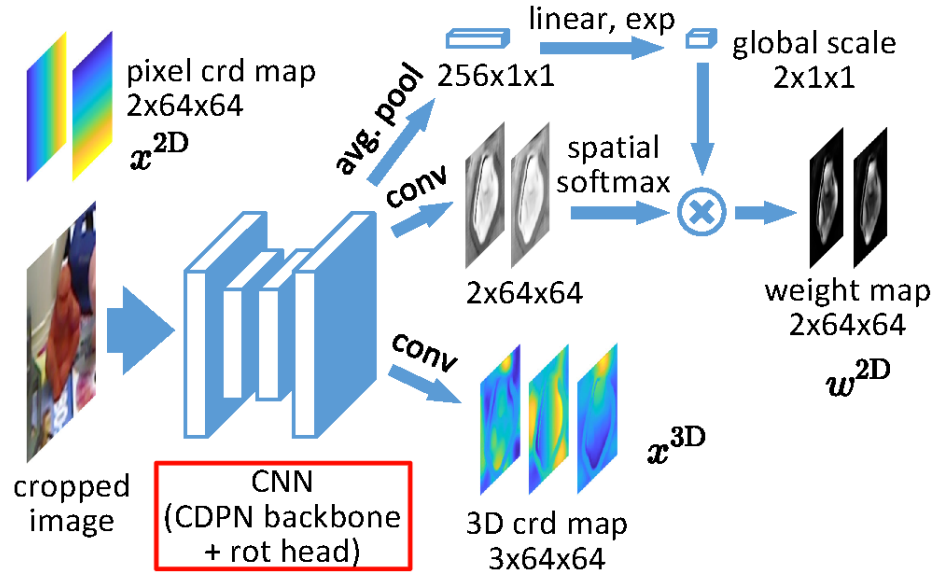


Fig. 1. Overview of the original EPro-PnP framework (Adapted from [1]). The system predicts dense correspondences for probabilistic pose estimation. **In this project, we focus on the network component (red box) and replace the original ResNet backbone with Swin Transformer, ConvNeXt, and HRNet to evaluate their impact.**

Mathematical Formulation. To provide context for our experiments, we briefly review the core formulation. The EPro-PnP framework interprets the PnP problem probabilistically. The posterior probability density of the pose y given the observations X is deduced to be:

$$p(y | X) = \frac{\exp\left(-\frac{1}{2} \sum_{i=1}^N f_i(y)^2\right)}{\int \exp\left(-\frac{1}{2} \sum_{i=1}^N f_i(y)^2\right) dy} \quad (1)$$

where $f_i(y)$ represents the weighted reprojection error for the i -th correspondence. Specifically, $f_i(y) = \sqrt{w_i} \|\pi(Rx_i^{3D} + t) - u_i^{2D}\|$, where $\pi(\cdot)$ is the projection function, and w_i is the predicted weight.

To optimize the network, the loss designed from the KL Divergence \mathcal{L}_{KL} is minimized to align the predicted distribution with the ground truth pose y_{gt} :

$$\mathcal{L}_{KL} = \frac{1}{2} \sum_{i=1}^N \|f_i(y_{gt})\|^2 + \log \int \exp\left(-\frac{1}{2} \sum_{i=1}^N \|f_i(y)\|^2\right) dy \quad (2)$$

The integral term (normalization constant Z) in the loss is intractable and is approximated using Monte Carlo importance sampling. By drawing K samples from a proposal distribution $q(y)$, Z is estimated as:

$$Z \approx \frac{1}{K} \sum_{j=1}^K \frac{\exp\left(-\frac{1}{2} \sum_{i=1}^N \|f_i(y_j)\|^2\right)}{q(y_j)} \quad (3)$$

where y_j represents the sampled pose, and $q(y_j)$ is the probability density of the proposal distribution at y_j . For detailed derivations and implementation notes, please refer to our online supplementary notes³.

2.2 Backbone Architectures

We integrated three distinct backbones into the framework to compare with the ResNet baseline:

Swin Transformer Swin Transformer [2] introduces a hierarchical Transformer architecture with shifted windows. This mechanism allows the model to capture global semantic context and long-range dependencies, which theoretically aids in understanding the overall structure of objects under occlusion.

ConvNeXt ConvNeXt [3] is a modern CNN architecture that incorporates design choices from Transformers (e.g., larger kernel sizes, layer normalization) while retaining the efficiency of convolutional networks. It serves as a strong representative of modern CNNs.

³ <https://hackmd.io/@YWTC2MbjSMYHTvsQjHAYeA/rkY6sNsGWL>

HRNet (High-Resolution Net) Unlike typical architectures that downsample features to low resolutions, HRNet [4] maintains high-resolution representations throughout the network. It connects high-to-low resolution convolution streams in parallel. **Head Design:** Notably, while ResNet, Swin, and ConvNeXt require a 3-layer MLP head (256 channels) to process features, our HRNet implementation uses a lightweight head with **0 hidden layers** and directly outputs from the 32-channel high-resolution stream. This direct mapping preserves precise spatial information critical for pixel-level coordinate regression.

3 Experiments

3.1 Experimental Setup

- **Dataset:** We utilized the standard LineMOD dataset [6]. The training set includes real images augmented with 1,000 synthetic images per class. Background augmentation was performed using the VOC2012 dataset [7].
- **Implementation Details:** To ensure a fair comparison, all models were trained with identical settings: 160 epochs, batch size of 32, and the RMSProp optimizer [8]. The loss function is defined as $L_{total} = L_{rot} + 0.02 \times L_{MC}$, where L_{rot} represents the dense 3D coordinate regression loss and L_{MC} denotes the Monte Carlo probabilistic pose loss (KL divergence).
- **System Environment:** The experiments were conducted on a workstation running WSL2 (Ubuntu 24.04). The software stack includes Python 3.10 and PyTorch 2.10.0 [5] (Nightly Build, dev20251207) with CUDA 12.8 support. Hardware acceleration was provided by a single **NVIDIA RTX 5070 Ti** GPU.
- **Evaluation Metrics:** We report the ADD(-S) metric, which accounts for symmetric objects, and the n°, n cm accuracy.

Table 1. Detailed model specifications and training hyperparameters. Note that HRNet utilizes a significantly lightweight head design compared to other backbones.

Category	Parameter	ResNet (Base)	Swin Trans.	ConvNeXt	HRNet
Model	Backbone Architecture	CNN (34 layers)	Transformer	Modern CNN	High-Res Streams
	Head Design (Layers)	3 hidden layers	3 hidden layers	3 hidden layers	0 (Direct output)
	Head Design (Channels)	256	256	256	32 (Lightweight)
Training (Common)	Input / Output Res.	256 × 256 / 64 × 64			
	Epochs / Batch Size	160 / 32			
	Learning Rate	1e-4 (Decay at 50, 100, 150)			
	Loss Function	$L_{rot} + 0.02 \times L_{MC}$			
Dataset (LineMOD)	Training Set	Real Images + 1,000 Synthetic Images/Class			
	Synthetic Strategy	Random VOC2012 Background Replacement			

4 Results

4.1 Performance Comparison

Table 2 summarizes the performance of the four backbones. HRNet demonstrates superior performance across most metrics.

Table 2. Performance comparison on the LineMOD dataset. HRNet demonstrates superior performance in strict metrics (0.02d, 2° 2cm) and overall mean scores. (Best results are **bolded**)

Backbone	ADD(-S)				$n^\circ, n \text{ cm}$			
	0.02d	0.05d	0.10d	Mean	2°, 2 cm	5°, 5 cm	10°, 10 cm	Mean
ResNet (Baseline)	32.99	73.28	92.59	61.83	65.79	96.72	99.57	85.88
Swin Transformer	36.13	76.39	94.55	64.47	67.42	97.46	99.75	86.63
ConvNeXt	38.28	77.11	94.27	65.26	71.02	97.73	99.84	87.68
HRNet	41.21	78.58	93.98	66.63	76.53	97.10	98.92	88.64

HRNet excels in the strict 2°, 2 cm metric (76.53%), significantly outperforming the baseline. This validates that high-resolution features are essential for high-precision pose estimation. ConvNeXt shows strong stability in looser thresholds (5°, 5 cm), making it a balanced choice for general applications.

5 Conclusion

In this study, we successfully integrated modern backbone architectures into the EPro-PnP framework by replacing the standard ResNet baseline with Swin Transformer, ConvNeXt, and HRNet. Our experimental results demonstrate that HRNet delivers the best overall performance, achieving the highest mean scores across both ADD(-S) and $n^\circ, n \text{ cm}$ metrics. Notably, HRNet excels under strict evaluation thresholds, securing the strongest results at 0.02d and 0.05d, which highlights its capability in high-precision tasks. Meanwhile, ConvNeXt shows competitive stability, performing best at medium-to-loose thresholds such as 5°, 5 cm and 10°, 10 cm. Ultimately, these findings suggest that high-resolution architectures like HRNet consistently provide superior accuracy for dense correspondence-based pose estimation compared to other modern alternatives.

6 Division of Work

The contributions of each author to this project are summarized as follows:

- **Yixuan Wu:**
 - Implementation of the HRNet backbone.
 - Execution and management of model training processes across all architectures.
 - Drafting the experimental results and analysis sections of the final report.
 - Overall aggregation and final formatting of the project report.
- **Chichun Ho:**
 - Implementation of the Swin Transformer and ConvNeXt backbones.
 - Compiling detailed mathematical derivations and theoretical notes (available online⁴).
 - Drafting the mathematical framework and methodology sections of the final report.

7 LLMs Used

The following Large Language Models (LLMs) were utilized to assist in the preparation of this project and report:

- **ChatGPT (GPT-5.1):**
 - Improving the clarity, grammar, and academic tone of the English descriptions.
 - Clarifying complex mathematical formulas and theoretical concepts related to EPro-PnP.
- **Gemini 3:**
 - Improving the clarity, grammar, and academic tone of the English descriptions.
 - Debugging Python code to resolve execution errors during the implementation phase.
 - Assisting in the creation of visualization scripts (e.g., `compare_viz.py`).

References

1. Chen, H., Wang, P., Wang, F., Tian, W., Xiong, L., Li, H.: EPro-PnP: Generalized End-to-End Probabilistic Perspective-n-Points for Monocular Object Pose Estimation. In: CVPR. pp. 2781–2790 (2022)
2. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., et al.: Swin Transformer: Hierarchical vision transformer using shifted windows. In: ICCV. pp. 10012–10022 (2021)
3. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A ConvNet for the 2020s. In: CVPR. pp. 11976–11986 (2022)

⁴ <https://hackmd.io/@YWTC2MbjSMYHTvsQjHAYeA/rkY6sNsGw1>

4. Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Wei, Y., et al.: Deep high-resolution representation learning for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **43**(10), 3349–3364 (2020)
5. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: *NeurIPS* (2019)
6. Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., Navab, N.: Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In: *ACCV*. pp. 548–562. Springer (2012)
7. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The Pascal Visual Object Classes (VOC) challenge. *Int. J. Comput. Vis.* **88**(2), 303–338 (2010)
8. Tieleman, T., Hinton, G.: Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* **4**(2), 26–31 (2012)