Linear-Covariance Loss for End-to-End Learning of 6D Pose Estimation (Supplementary Material)

1. Linearization of PnP Solver

Implicit Function Theorem. The implicit function theorem (IFT) [5] states the following:

Given $f : \mathbb{R}^{n+m} \rightarrow \mathbb{R}^m$ a continuously differentiable function with input $(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{R}^n \times \mathbb{R}^m$, if a point $(\boldsymbol{a}^*, \boldsymbol{b}^*)$ satisfies

$$f(\boldsymbol{a}^*, \boldsymbol{b}^*) = \mathbf{0} \ , \tag{1}$$

and the Jacobian matrix $\frac{\partial f}{\partial \boldsymbol{b}}(\boldsymbol{a}^*, \boldsymbol{b}^*)$ is invertible, then there exists a unique continuously differentiable function $g(\boldsymbol{a}) : \mathbb{R}^n \rightarrow \mathbb{R}^m$ such that

$$\boldsymbol{b}^* = g(\boldsymbol{a}^*) \ , \tag{2}$$

and

$$f(\boldsymbol{a}^*, g(\boldsymbol{a}^*)) = \mathbf{0} \ . \tag{3}$$

The Jacobian matrix $\frac{\partial g}{\partial \boldsymbol{a}}(\boldsymbol{a}^*)$ is given by

$$\frac{\partial g}{\partial \boldsymbol{a}}(\boldsymbol{a}^*) = -\left[\frac{\partial f}{\partial \boldsymbol{b}}(\boldsymbol{a}^*, \boldsymbol{b}^*)\right]^{-1} \cdot \frac{\partial f}{\partial \boldsymbol{a}}(\boldsymbol{a}^*, \boldsymbol{b}^*) \ . \tag{4}$$

PnP Linearization. Following the same notation as in the main paper, the PnP solver computes the function

$$g(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{w}) = \arg\min_{\boldsymbol{y}} \frac{1}{2} \sum_{i}^{N} \|\boldsymbol{w}_i \circ \boldsymbol{r}_i\|^2 \ , \tag{5}$$

where $\boldsymbol{x}_i$ is the i-th image 2D point, $\boldsymbol{z}_i$ is the i-th 3D point, $\boldsymbol{w}_i$ is the corresponding weight, and

$$\boldsymbol{r}_i = \boldsymbol{x}_i - \pi(\boldsymbol{z}_i, \boldsymbol{y}) \tag{6}$$

is the reprojection residual for the i-th correspondence given pose $\boldsymbol{y}$.

Eq. 5 implies that the solution $\boldsymbol{y}^*$ is the stationary point of the negative log likelihood (NLL) function

$$nll(\boldsymbol{y}) = \frac{1}{2} \sum_{i}^{N} \|\boldsymbol{w}_i \circ \boldsymbol{r}_i\|^2 \ . \tag{7}$$

Since $\boldsymbol{y}^*$ is the stationary point of the NLL function, the first order derivative of the NLL w.r.t. $\boldsymbol{y}^*$ should be zero, i.e.,

$$\left.\frac{\partial nll(\boldsymbol{y})}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}=\boldsymbol{y}^*} = \mathbf{0} \ . \tag{8}$$

Eqs. 1, 2 and 3 in the PnP case can subsequently be specialized as

$$f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{w})|_{\boldsymbol{y}=\boldsymbol{y}^*} = \left.\frac{\partial nll(\boldsymbol{y})}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}=\boldsymbol{y}^*} = \mathbf{0} \ , \tag{9}$$

$$\boldsymbol{y}^* = g(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{w}) \ , \tag{10}$$

and

$$f(\boldsymbol{x}, g(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{w}), \boldsymbol{z}, \boldsymbol{w})|_{\boldsymbol{y}=\boldsymbol{y}^*} = \mathbf{0} \ . \tag{11}$$

According to Eq. 4, the gradient of the pose $\boldsymbol{y}$ w.r.t. the 2D locations $\boldsymbol{x}$ at $\boldsymbol{y}^*$ is

$$\begin{aligned}
\left.\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{y}^*} &= \left.\frac{\partial g(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{w})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{y}^*} \ , \\
&= -\left.\left[\left[\frac{\partial^2 nll(\boldsymbol{y})}{\partial \boldsymbol{y}^2}\right]^{-1} \cdot \frac{\partial^2 nll(\boldsymbol{y})}{\partial \boldsymbol{y} \partial \boldsymbol{x}}\right]\right|_{\boldsymbol{y}^*} \ , \\
&= -H^{-1} \cdot \left.\frac{\partial^2 nll(\boldsymbol{y})}{\partial \boldsymbol{y} \partial \boldsymbol{x}}\right|_{\boldsymbol{y}^*} \ ,
\end{aligned} \tag{12}$$

with $nll(\boldsymbol{y})$ defined by Eq. 7.

Given the noisy correspondences $\{\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{w}\}$, we compute the perfect correspondences $\{\boldsymbol{x}_p, \boldsymbol{z}, \boldsymbol{w}\}$ with $\boldsymbol{x}_{p,i} = \pi(\boldsymbol{z}_i, \boldsymbol{y}_{gt})$ under the ground-truth pose $\boldsymbol{y}_{gt}$. We then linearize the PnP solver around $\{\boldsymbol{x}_p, \boldsymbol{z}, \boldsymbol{w}\}$ and $\boldsymbol{y}_{gt}$ using the first-order Taylor expansion as

$$\boldsymbol{y} = \boldsymbol{y}_{gt} + A(\boldsymbol{z}, \boldsymbol{w}) \cdot \boldsymbol{r}_{gt} \ , \tag{13}$$

with

$$\boldsymbol{r}_{gt} = \boldsymbol{x} - \boldsymbol{x}_{gt} \tag{14}$$

being the residual vector at $\boldsymbol{y}_{gt}$, and

$$A(\boldsymbol{z}, \boldsymbol{w}) = -H^{-1} \cdot \left.\frac{\partial^2 nll(\boldsymbol{y})}{\partial \boldsymbol{y} \partial \boldsymbol{x}}\right|_{\boldsymbol{y}=\boldsymbol{y}_{gt}, \boldsymbol{x}=\boldsymbol{x}_p} . \tag{15}$$

The Hessian H of the NLL function is also used to compute the prior loss, as stated in Sec. 3.3 in the main paper.

2. Detailed Results on Gradient Correctness

We further provide the whole correctness curves to show how the correctness evolves as training progresses.

As illustrated in Fig. 1, at the very beginning, when the correspondences have large errors, both EPro-PnP [2] and BPnP [1] have good correctness. However, their correctness drops when the training proceeds. Since the linear-covariance loss is designed to address this problem, it always maintains a correctness close to 100%.

3. Details on ZebraPose-based Experiments

Implementation Details. Our coordinate-wise encoding scheme assigns 3 binary codes to a vertex, eliminating the look up operation. To reduce the number of binary bits for

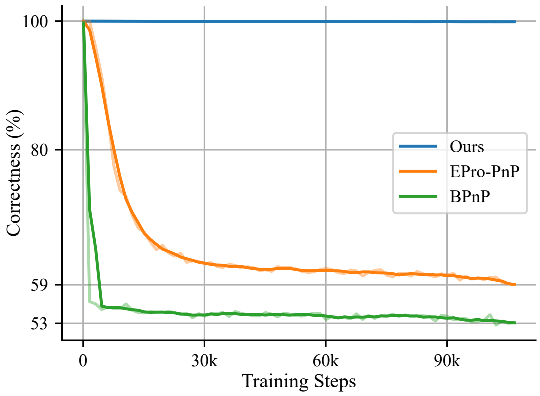

Figure 1. **Correctness curves of the PnP layers.** A 3D point is considered to have a correct gradient if moving in the negative gradient direction leads to a smaller 2D reprojection error. The LC loss yields almost 100% correctness. The correctness of EPro-PnP drops slowly, and ending with about 59% correctness. BPnP drops quickly when training begins, and ends with about 53% correctness. The dark curves are smoothed versions of the light ones.

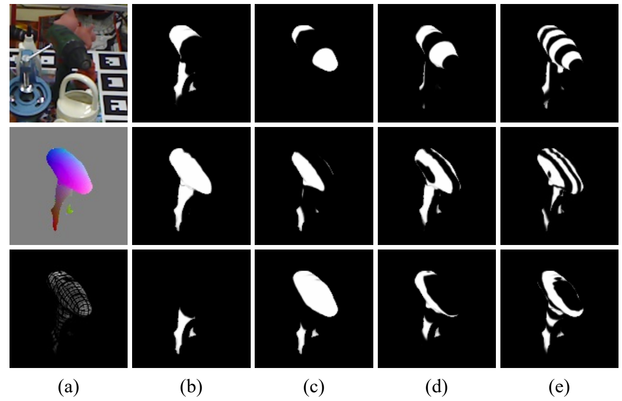

Figure 2. **Visualizations for the ZebraPose-based model.** (a) Visualizations of the input image patch, decoded object coordinates and the predicted weight map. (b)-(e) Visualizations of the predicted masks of coordinate components with the most significant bit at the left and the x component at the top. The pixels predicted as background are masked out for clarity.

Row	Method	ADD(-S)
A0	ZebraPose [7]	76.91
A1	ZebraPose baseline	75.19
A2	A1 + LC loss	**78.06**

Table 1. **Results of the ZebraPose [7] based experiments on the LM-O dataset.**

prediction, we rotate some of the objects to minimize their span along the x, y, z directions. We use 7 bits to represent the coordinate component with the largest span, and calculate the binary count of the other components based on their relative span w.r.t. largest one. Specifically, given the sizes $s_i, i \in \{x, y, z\}$, of an object and their maximum s, the bit count of each component is calculated as $n_i = \text{round}(n + log_2(s_i/s))$, where $n = 7$ is the maximum bit count per component. This is to reduce the unpredictable bits for flat-shaped objects such as scissors.

Results. As shown in Tab. 1, after switching from the global vertex encoding to our coordinate-wise encoding (A0 *vs.* A1), the performance drops by about 1.7 points. When the LC loss is applied, the performance drop is compensated, surpassing the original ZebraPose [7].

Visualizations. As illustrated by Fig. 2, the learned weight map successfully captures the error distribution of the predicted 3D coordinates in a geometry-aware manner, generating low weights for code transition regions and high weights for object endpoint regions.

4. Detailed Results on LM-O and YCB-V

For the LM-O dataset, we provide the detailed comparison of ADD(-S) scores with state-of-the-art methods, when the linear-covariance (LC) loss is applied to GDR-Net and ZebraPose on LM-O in Tab. 3.

For the YCB-V dataset, we provide the detailed comparison of ADD(-S) scores (Tab. 2) and AUC scores (Tab. 4) between the baseline methods and the versions where the LC loss is applied.

Object	[8]	[7]	[8]-LC	[7]-LC
002_master_chef_can	41.5	**62.6**	38.7	51.6
003_cracker_box	83.2	98.5	96.2	**99.7**
004_sugar_box	91.5	96.3	98.1	**99.4**
005_tomato_soup_can	65.9	**80.5**	77.6	79.6
006_mustard_bottle	90.2	**100**	77.0	99.7
007_tuna_fish_can	44.2	70.5	63.2	**86.1**
008_pudding_box	2.8	**99.5**	81.3	99.1
009_gelatin_box	61.7	**97.2**	81.8	94.9
010_potted_meat_can	64.9	**76.9**	68.1	73.9
011_banana	64.1	71.2	71.0	**95.8**
019_pitcher_base	99.0	**100**	**100**	**100**
021_bleach_cleanser	73.8	75.9	69.9	**85.6**
024_bowl*	37.7	18.5	**44.1**	35.2
025_mug	61.5	77.5	46.2	**88.7**
035_power_drill	78.5	97.4	**99.7**	99.2
036_wood_block*	59.5	87.6	**91.7**	82.6
037_scissors	3.9	**71.8**	14.9	56.9
040_large_marker	7.4	23.3	**29.3**	27.8
051_large_clamp*	69.8	**87.6**	80.5	84.4
052_extra_large_clamp*	90.0	98.0	95.5	**99.1**
061_foam_brick*	71.9	**99.3**	57.6	91.3
mean	60.1	80.5	70.6	**82.4**

Table 2. **Detailed ADD(-S) scores on YCB-V.** We report the scores of the original baseline methods, GDR-Net [8] and ZebraPose [7], and also the scores after applying our LC loss, respectively (denoted by "-LC"). (*) denotes symmetric objects on which the ADD-S score is reported.

Method	RePOSE [4]	RNNPose [9]	SO-Pose [3]	DProST [6]	GDR-Net [8]	ZebraPose [7]	GDR-LC	Zebra-LC
ape	31.1	37.18	48.4	51.4	46.8	57.9	44.44	**61.57**
can	80.0	88.07	85.8	78.7	90.8	95.0	89.06	**97.35**
cat	25.6	29.15	32.7	48.1	40.5	60.6	49.87	**64.49**
driller	73.1	88.14	77.4	77.4	82.6	**94.8**	87.81	94.65
duck	43.0	49.17	48.9	45.4	46.9	64.5	56.08	**66.82**
eggbox*	51.7	66.98	52.4	55.3	54.2	70.9	62.81	**71.77**
glue*	54.3	63.79	78.3	76.9	75.8	**88.7**	68.88	86.35
holepuncher	53.6	62.76	75.3	67.4	60.1	**83.0**	72.89	81.49
mean	51.6	60.65	62.3	62.6	62.2	76.9	66.48	**78.06**

Table 3. **Comparison with the state of the art on LM-O.** (*) denotes symmetric objects on which the ADD-S score is reported. "GDR-LC" denotes the LC loss with the GDR-Net [8] baseline, "Zebra-LC" denotes the LC loss with the ZebraPose [7] baseline.

Method	GDR-Net [8]		ZebraPose [7]		GDR-Net-LC		ZebraPose-LC	
Metric	AUC of ADD-S	AUC of ADD(-S)	AUC of ADD-S	AUC of ADD(-S)	AUC of ADD-S	AUC of ADD(-S)	AUC of ADD-S	AUC of ADD(-S)
002_master_chef_can	*96.3	*65.2	93.7	75.4	85.6, *90.1	57.5, *61.6	88.4	66.9
003_cracker_box	*97.0	*88.8	93.0	87.8	93.1, *98.1	86.8, *91.6	93.7	88.3
004_sugar_box	*98.9	*95.0	95.1	90.9	95.9, *99.8	92.3, *97.4	94.7	90.3
005_tomato_soup_can	*96.5	*91.9	94.4	90.1	92.8, *96.2	88.2, *93.0	93.4	89.2
006_mustard_bottle	*100	*92.8	96.0	92.6	94.1, *97.6	88.2, *93.1	95.1	90.9
007_tuna_fish_can	*99.4	*94.2	96.9	92.6	96.2, *99.9	92.1, *96.9	97.2	94.1
008_pudding_box	*64.6	*44.7	97.2	95.3	94.4, *99.1	90.4, *95.3	96.7	94.7
009_gelatin_box	*97.1	*92.5	96.8	94.8	95.1, *99.9	91.7, *96.8	96.7	94.6
010_potted_meat_can	*86.0	*80.2	91.7	83.6	85.8, *89.0	79.6, *83.8	91.3	82.5
011_banana	*96.3	*85.8	92.6	84.6	92.2, *97.6	83.2, *88.0	95.3	90.1
019_pitcher_base	*99.9	*98.5	96.4	93.4	96.6, *100	93.5, *98.4	96.4	93.2
021_bleach_cleanser	*94.2	*84.3	89.5	80.0	86.3, *91.2	77.0, *82.0	90.5	82.3
024_bowl*	*85.7	*85.7	37.1	37.1	83.1, *88.6	83.1, *88.6	63.9	63.9
025_mug	*99.6	*94.0	96.1	90.8	92.7, *96.5	83.9, *88.9	96.5	92.3
035_power_drill	*97.5	*90.1	95.0	89.7	96.1, *99.9	92.6, *97.9	95.4	90.8
036_wood_block*	*82.5	*82.5	84.5	84.5	87.1, *92.2	87.1, *92.2	81.2	81.2
037_scissors	*63.8	*49.5	92.5	84.5	75.8, *80.4	63.5, *67.8	88.3	79.0
040_large_marker	*88.0	*76.1	80.4	69.5	77.5, *81.8	68.8, *73.5	77.6	68.5
051_large_clamp*	*89.3	*89.3	85.6	85.6	83.1, *87.9	83.1, *87.9	86.8	86.8
052_extra_large_clamp*	*93.5	*93.5	92.5	92.5	91.4, *95.8	91.4, *95.8	94.6	94.6
061_foam_brick*	*96.9	*96.9	95.3	95.3	90.0, *94.6	90.0, *94.6	93.2	93.2
mean	*91.6	*84.4	90.1	85.3	89.8, *94.1	84.0, *88.8	**90.8**	**86.1**

Table 4. **Detailed AUC scores on YCB-V.** We report the scores of the original baseline methods and the scores after the LC loss is applied. "GDR-Net-LC" denotes the LC loss with the GDR-Net [8] baseline, "ZebraPose-LC" denotes the LC loss with the ZebraPose [7] baseline. A (*) after the object name denotes the symmetric objects on which the ADD-S score is reported. A (*) before the AUC score indicates that the AUC is computed with 11-points interpolation.

References

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

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

[3] Yan Di, Fabian Manhardt, Gu Wang, Xiangyang Ji, Nassir Navab, and Federico Tombari. So-pose: Exploiting self-occlusion for direct 6d pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*

(ICCV), pages 12396–12405, October 2021.

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

[5] Steven George Krantz and Harold R Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.

[6] Jaewoo Park and Nam Ik Cho. Dprost: Dynamic projective spatial transformer network for 6d pose estimation. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 363–379, Cham, 2022. Springer Nature Switzerland.

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

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

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