Linear-Covariance Loss for End-to-End Learning of 6D Pose Estimation

Fulin Liu
Beihang University
liufulin@buaa.edu.cn

Yinlin Hu
MagicLeap
yhu@magicleap.com

Mathieu Salzmann
EPFL, ClearSpace
mathieu.salzmann@epfl.ch

Abstract

Most modern image-based 6D object pose estimation methods learn to predict 2D-3D correspondences, from which the pose can be obtained using a PnP solver. Because of the non-differentiable nature of common PnP solvers, these methods are supervised via the individual correspondences. To address this, several methods have designed differentiable PnP strategies, thus imposing supervision on the pose obtained after the PnP step. Here, we argue that this conflicts with the averaging nature of the PnP problem, leading to gradients that may encourage the network to degrade the accuracy of individual correspondences. To address this, we derive a loss function that exploits the ground truth pose before solving the PnP problem. Specifically, we linearize the PnP solver around the ground-truth pose and compute the covariance of the resulting pose distribution. We then define our loss based on the diagonal covariance elements, which entails considering the final pose estimate yet not suffering from the PnP averaging issue. Our experiments show that our loss consistently improves the pose estimation accuracy for both dense and sparse correspondence based methods, achieving state-of-the-art results on both Linemod-Occluded and YCB-Video.

1. Introduction

Estimating the 6D pose of 3D objects from monocular images is a core computer vision task, with many real world applications, such as robotics manipulation [53, 54], autonomous driving [9, 48] and augmented reality [10, 32]. Although this task can be facilitated by the use of RGBD input, depth sensors are not ubiquitous, and thus 6D object pose estimation from RGB images remains an active research area.

With the development of deep neural networks (DNNs), early methods [1, 14, 26, 49] formulated pose estimation as a regression problem, directly mapping the input image to the 6D object pose. More recently, most works [5, 21, 23, 33, 35, 38, 39, 41, 42, 43, 44] draw inspiration from geometry and seek to predict 2D-3D correspondences, from which the 6D pose can be obtained by solving the Perspective-n-Points (PnP) problem. While effective, these methods supervise the training process with the individual correspondences, and not with the ground-truth pose itself, as standard PnP solvers are not differentiable.

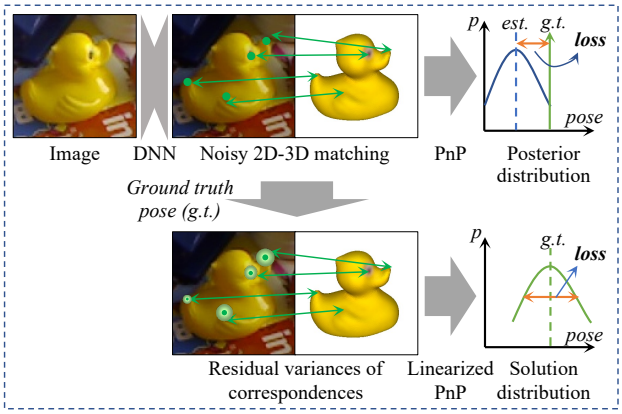


Figure 1. **Difference between other differentiable PnP losses and our proposed loss.** Other methods (first row) first solve for the object pose from noisy correspondences, and then compute the loss based on the resulting pose and the ground truth. By contrast, we (second row) utilize the ground-truth pose to estimate the residual variance of the correspondences and compute the covariance of the pose distribution. We define our loss based on the diagonal elements of this covariance.

To enable end-to-end training, several attempts have been made to incorporate the PnP solver as a differentiable network layer [3, 4, 6]. While these methods make it possible to employ pose-driven loss functions to train the DNN, they only leverage the optimal pose as supervision, thus not imposing constraints on other pose candidates. In [7], this was addressed by deriving a loss function based on the posterior pose distribution, encouraging a larger posterior for the ground truth and smaller posteriors for the other poses.

Nevertheless, to the best of our knowledge, all of these differentiable PnP layers have a common property: They first solve the PnP problem to obtain either the pose [3, 4, 6] or the posterior pose distribution [7], and then compute the error to be backpropagated based on a dedicated loss func-

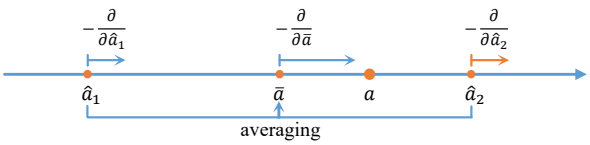


Figure 2. **Gradients after averaging.** In this toy example, the gradient will drive $\hat{a}_2$ away from the true a value, although $\bar{a}$ is driven closer to a.

tion and the ground-truth pose. That is, they introduce the ground-truth information only *after* the pose has been computed. While this may seem a natural strategy when incorporating a differentiable layer, we argue that this conflicts with the averaging nature of the PnP problem, which aggregates multiple noisy measurements into a single estimate.

To illustrate this, let us consider a simpler problem with two noisy measurements, $\hat{a}_1$ and $\hat{a}_2$, of a value a, and seek to minimize the distance between the average value $\bar{a} = (\hat{a}_1 + \hat{a}_2)/2$ and a, i.e., $|\bar{a} - a|$. As $\hat{a}_1$ and $\hat{a}_2$ contribute to the loss in the same manner, they are updated with the same gradient, irrespectively of their individual error w.r.t. a. For example, as depicted in Fig. 2, if $\hat{a}_1$ and $\hat{a}_2$ are on either side of a, the gradient will drive one of these estimates further from the true a value, i.e., degrade the individual prediction quality of this estimate.

In the case of PnP, 4 2D-3D correspondences are in general sufficient to obtain a unique pose [28]. If more than 4 correspondences are provided, the PnP solver performs a form of "averaging", and thus may yield a similar effect as in the previous toy example: It may encourage a decrease in accuracy of some of the correspondences, thus essentially providing confusing signal in the network training process. While this can be alleviated by using the pose-driven loss in conjunction with correspondence-level supervision, this strategy circumvents the problem instead of addressing it.

By contrast, in this paper, we introduce an approach to explicitly tackle the gradient issue arising from the averaging process of the PnP problem. To this end, we leverage the covariance of the pose distribution computed by exploiting the ground-truth pose *before* solving for the pose, as illustrated by Fig. 1. Given noisy 2D-3D correspondences as input to the PnP solver, we consider the distribution of PnP-computed poses around the ground-truth one. We then approximate the covariance of this distribution by linearizing the PnP solver around the ground truth. This lets us design a loss function that minimizes the diagonal covariance elements, which entails minimizing the 2D-3D correspondence residuals while nonetheless considering the pose estimated via PnP. Our formalism also applies to the weighted PnP scenario, allowing us to compute weights for the individual correspondences so as to emphasize the ones that benefit the pose estimate.

Our experiments on several datasets and using both sparse and dense correspondence based methods evidence the benefits of our approach. In particular, applying our loss to the state-of-the-art ZebraPose [43], lets us achieve state-of-the-art performance on both the LM-O and YCB-V datasets. Our code is available at https://github.com/fulliu/lc.

2. Related Work

In this section, we focus on the geometry-driven methods, which are the most relevant to our core contributions. Geometry-driven methods solve the pose estimation task by first extracting 2D-3D correspondences from RGB images, and then computing the 6D pose using a PnP solver. They can be roughly categorized into sparse keypoints-based methods and dense prediction ones.

Sparse Correspondences based Estimation. Sparse keypoints-based methods predict the 2D projected locations of predefined 3D keypoints. Standard 3D keypoint choices include the object centroid, the corners of the object bounding box, semantic keypoints, or keypoints sampled from the object model, e.g., via the farthest point sampling algorithm [41]. Specifically, BB8 [42] predicts the image locations of the 8 corners of the object bounding box; Bugra *et al*. [44] add the centroid of the object 3D model to the correspondence list; Pavlakos *et al*. [39] use heatmaps to predict the 2D locations of semantic keypoints. A voting scheme is also utilized to aggregate multiple predictions for the same 2D point so as to improve robustness to partial occlusions. Oberweger *et al*. [35] combine different heatmap predictions generated from different image patches; Hu *et al*. [23] aggregate the 2D keypoint predictions from all pixels belonging to the given target; Similarly, PVNet [41] regresses the vector-field pointing from each object pixel to the 2D locations. Chen *et al*. [5] combine heatmap-based keypoint predictions at different scales and use occlude-and-blackout data augmentation to boost occlusion robustness. All of these methods use a traditional PnP solver as second stage, with the exception of Hu *et al*. [22] that replaces the PnP solver with an MLP, enabling end-to-end training of the whole pipeline.

Dense Correspondences based Estimation. Methods based on dense correspondences predict pixel-wise 3D object coordinates inside the object instance mask. iPose [21] achieves this by first cropping the image patch containing the target based on the instance mask; Pix2Pose [38] uses the 2D image bounding box from a detector to crop the object, and then regress both the 3D coordinates and their errors, using generative adversarial training for occlusion robustness. Similarly to the sparse methods, the PnP solver can be replaced by an MLP. CDPN [30] uses an MLP to regress the object translation but a PnP solver to compute the rotation. When an MLP is used, the correspondences are not restricted to 2D-3D points. GDR-

net [46] additionally feeds surface region attention maps into an MLP-based Patch-PnP module. Based on GDR-Net, SO-Pose [12] exploits a self-occlusion map, replacing the surface region attention maps as another input modality to exploit object self-occlusions in pose estimation. While all previous works directly regress the 3D object coordinates, DPOD [51] regresses a dense UV texture map, the UV values are further mapped to predefined 3D coordinates. EPOS [19] divides object surface into fragments and predicts both fragments and local coordinates relative to the fragments. ZebraPose [43] proposes a hierarchical binary vertex encoding defined by grouping surface vertices. A segmentation network is employed to predict such codes in a coarse-to-fine manner. 2D-3D correspondences are thus extracted by predicting pixel-wise codes inside the object mask and mapping the codes to 3D coordinates.

End-to-End Learning with PnP Solvers. The PnP solver is typically treated as a non-differentiable layer, making it impossible to impose loss functions directly on the pose. Instead, surrogate loss functions are used in the first stage, encouraging the network to generate correct 2D-3D correspondences, but discarding information about the global object structure and the subsequent PnP step. Nevertheless, several attempts at differentiating through the PnP solver have been made. To this end, Brachmann *et al*. [3] turn to numerical central differences, which introduce a computational burden. In their subsequent work [4], the authors instead rely on approximations from the last step of the Gauss-Newton iterations. MLP-based solvers [12, 30, 22, 46], where target pose is regressed from input geometry features are also deployed. Although these methods provide more flexibility on input modalities, it is hard to further improve their accuracy because of the lack of precise geometry model. Chen *et al*. [6] observe that the gradient of the optimal pose can be calculated by applying the implicit function theorem [27] around the optimal solution. By contrast, EPro-PnP [7] relies on a loss function based on the whole posterior pose distribution instead of only its maximum, i.e., the true pose. However, as discussed in Section 1, the averaging nature of PnP solvers makes loss functions based on the PnP solution suboptimal, as they will lead to gradients that degrade the accuracy of some of the 2D-3D correspondences. This is what we address in this paper.

3. Method

3.1. Overview

Let us now describe our method. To this end, we start with a general geometry-driven approach to 6D object pose estimation. Given a single RGB image, the first stage of geometry-driven 6D pose estimation pipelines aims to predict N noisy 2D-3D correspondences, potentially with associated weights. These can be expressed as

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1^T & \boldsymbol{x}_2^T & \cdots & \boldsymbol{x}_N^T \end{bmatrix}^T \in \mathbb{R}^{2N} , \tag{1}$$

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{z}_1^T & \boldsymbol{z}_2^T & \cdots & \boldsymbol{z}_N^T \end{bmatrix}^T \in \mathbb{R}^{3N} , \tag{2}$$

$$\boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}_1^T & \boldsymbol{w}_2^T & \cdots & \boldsymbol{w}_N^T \end{bmatrix}^T \in \mathbb{R}^{2N} , \tag{3}$$

where $\{\boldsymbol{x}_i, \boldsymbol{z}_i, \boldsymbol{w}_i\}$ encodes the i-th correspondence with weight. Specifically, $\boldsymbol{z}_i$ is the 3D object point, $\boldsymbol{x}_i$ is its 2D projection, and $\boldsymbol{w}_i$ is the associated weight. A PnP solver, compactly denoted by $g(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{w})$, can then be thought of as producing a maximum likelihood estimate of the pose $\boldsymbol{y}$, relying on the sum of the squared reprojection errors as a negative log likelihood (NLL). That is, we can write

$$g(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{w}) = \arg\min_{\boldsymbol{y}} \frac{1}{2} \sum_{i}^{N} \left\| \boldsymbol{w}_i \circ \boldsymbol{r}_i \right\|^2 , \tag{4}$$

where

$$\boldsymbol{r}_i = \boldsymbol{x}_i - \pi(\boldsymbol{z}_i, \boldsymbol{y}) \tag{5}$$

is the reprojection residual for the i-th correspondence given pose $\boldsymbol{y}$, $\pi(\boldsymbol{z}_i, \boldsymbol{y})$ is the perspective projection involving the camera intrinsics, and $\circ$ denotes the element-wise product.

If more than 4 correspondences are supplied, the PnP problem is over-determined, and the optimal solution can be thought of as a form of "weighted average" of the candidate poses obtained from all possible minimal correspondence sets. When averaging is involved, penalizing the difference between the obtained pose and the ground truth does not guarantee to yield gradient directions that improve all the correspondences. In fact, Chen *et al*. [6] illustrated cases where the final pose has successfully converged to the ground truth while the correspondences had not.

To overcome this, we propose to introduce the ground truth *before* solving the PnP problem and estimate the solution distribution around the ground-truth pose. This lets us build a loss function on top of this distribution, specifically, based on the distribution covariance. We discuss this in detail below.

3.2. Covariance of the Pose Distribution

Linear Approximation of the PnP Solver. Let $\{\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{w}\}$ denotes a set of noisy correspondences with weights, and $\boldsymbol{y}_{gt}$ be the ground-truth pose. Then, following a first-order Taylor expansion, the solution $\boldsymbol{y}$ obtained by a PnP solver $g(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{w})$ can be approximated as

$$\boldsymbol{y} = \boldsymbol{y}_{gt} + A(\boldsymbol{z}, \boldsymbol{w}) \cdot \boldsymbol{r}_{gt} , \tag{6}$$

where $\boldsymbol{r}_{gt} \in \mathbb{R}^{2N \times 1}$ is the residual vector given by $\boldsymbol{r}_{gt} = \boldsymbol{x} - \boldsymbol{x}_p$, with $\boldsymbol{x}_{p,i} = \pi(\boldsymbol{z}_i, \boldsymbol{y}_{gt})$. $A(\boldsymbol{z}, \boldsymbol{w})$ encodes the gradient of pose $\boldsymbol{y}$ w.r.t. the perfect 2D locations $\boldsymbol{x}_p$, i.e.,

$$A(\boldsymbol{z}, \boldsymbol{w}) = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} = \left. \frac{\partial g(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{w})}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x}=\boldsymbol{x}_p} . \tag{7}$$

which can be computed following the implicit function theorem [6, 27].

Pose Distribution and Covariance. Looking at Eq. 6 reveals that, in our linearized model, the pose estimate is computed from a linear combination of the residuals in $\boldsymbol{r}_{gt}$. Let $M \in \mathbb{R}^{2N \times 2N}$ be the covariance matrix of the residuals. Then, the pose covariance matrix C is given by

$$C = A(\boldsymbol{z}, \boldsymbol{w}) \cdot M \cdot A(\boldsymbol{z}, \boldsymbol{w})^T \ , \tag{8}$$

and its expected value is $\boldsymbol{y}_{gt}$.

This formalism still requires us to define the residuals covariance M. To this end, we assume independence of the measurements, and express M as the diagonal matrix

$$M = diag\{\boldsymbol{r}_{gt} \circ \boldsymbol{r}_{gt}\}. \tag{9}$$

During the calculation of the coefficient matrix $A(\boldsymbol{z}, \boldsymbol{w})$, both the object 3D structure and the properties of the PnP solver are exploited. Furthermore, $A(\boldsymbol{z}, \boldsymbol{w})$ is evaluated with perfect correspondences at the ground-truth pose $\boldsymbol{y}_{gt}$, eliminating the need to solve the PnP problem, since $\boldsymbol{y}_{gt}$ is the solution. This makes the covariance very fast to evaluate. It is worth noting that 3D coordinates vector $\boldsymbol{z}$ is treated as a constant during the calculation of the coefficient matrix $A(\boldsymbol{z}, \boldsymbol{w})$, even if $\boldsymbol{z}$ is the output of the DNN's first stage, as is the case with dense prediction frameworks. This means that no gradient will be propagated from A to $\boldsymbol{z}$.

3.3. Linear-Covariance Loss

Pose Representation. Different representations of the 6D pose have been proposed in the past [15, 29, 31, 37, 52]. The mathematical form of the pose covariance matrix in Eq. 8 is valid for any such representation. Nevertheless, to simplify the presentation of our loss function, we now discuss the representation that will be used in our experiments.

Specifically, we advocate for the use of a representation that reflects the evaluation metric of the target task. In particular, considering robotics applications where 3D error matters, we represent the pose with the transformed 3D coordinates of the object bounding box corners. That is, we write

$$\boldsymbol{y} = \begin{bmatrix} T(\boldsymbol{b}_1; \boldsymbol{R}, \boldsymbol{t})^T & \cdots & T(\boldsymbol{b}_8; \boldsymbol{R}, \boldsymbol{t})^T \end{bmatrix}^T \in \mathbb{R}^{24} \ , \tag{10}$$

where $\boldsymbol{b}_i$ is a 3D bounding box corner, and $T(\cdot; \boldsymbol{R}, \boldsymbol{t})$ is rigid transformation performed with a pose encoded with a rotation matrix $\boldsymbol{R}$ and a translation vector $\boldsymbol{t}$, as would be output by a PnP solver.

Covariance Loss. Given the covariance matrix $C \in \mathbb{R}^{24 \times 24}$, a natural choice of loss function is $trace(C)$, i.e., the sum of the diagonal covariance elements. This is because each diagonal element of the covariance matrix encodes the square error of the corresponding pose parameter, i.e., the square of its difference w.r.t. the ground-truth value. Such a square error, however, is sensitive to outliers, and does not reflect the nature of our pose representation, i.e., the fact that it contains 8 3D point coordinates and not 24 independent values.

Taking this into account, we therefore define our covariance loss as

$$E_{cov}(\boldsymbol{w}, \boldsymbol{r}_{gt}) = \frac{1}{8} \sum_{i=1}^{8} \sqrt{\sum_{j=3i-2}^{3i} C_{jj}} \ , \tag{11}$$

which we express as a function of $\boldsymbol{w}$ and $\boldsymbol{r}_{gt}$ to indicate that the gradient is computed w.r.t. only $\boldsymbol{w}$ and $\boldsymbol{r}_{gt}$, both of which depend on the network parameters, while all other quantities involved are treated as constants during back-propagation.

Weight-related Losses. Minimizing only the loss function of Eq. 11 does not sufficiently constrain the weights $\boldsymbol{w}$, as for example, scaling them does not affect the solution of Eq. 4. Recall that, in linearly weighted least square problems, the weights are typically taken to be inversely proportional to the residual errors. Therefore, we propose to treat the weights as priors on the reprojection errors, and define a loss based on the covariance C_{prior} of the pose prior.

One way to compute C_{prior} would consist of re-defining the residuals as the inverse of the weights and reusing Eqs. 9 and 8. Instead, we approximate C_{prior} as the inverse Hessian H^{-1} of the NLL in Eq. 4 at $\boldsymbol{y}_{gt}$ [40], i.e.,

$$C_{prior} = H(\boldsymbol{y}_{gt})^{-1}. \tag{12}$$

This formulation is more accurate than using the inverse weights and more efficient since $H(\boldsymbol{y}_{gt})^{-1}$ is already evaluated when linearizing the PnP solver. We then define a prior loss as

$$E_{prior}(\boldsymbol{w}) = \frac{1}{8} \sum_{i=1}^{8} \sqrt{\sum_{j=3i-2}^{3i} C_{prior,jj}} \ . \tag{13}$$

We further seek to supervise the weights to encourage them to benefit the final pose estimate obtained by the PnP solver. To achieve this, we rely on our linearized PnP solver and compute an error vector

$$\boldsymbol{e}_{linear}(\boldsymbol{w}) = \boldsymbol{y} - \boldsymbol{y}_{gt} = A(\boldsymbol{z}, \boldsymbol{w}) \cdot \boldsymbol{r}_{gt} \in \mathbb{R}^K \ . \tag{14}$$

We then write a corresponding linear loss as

$$E_{linear}(\boldsymbol{w}) = \frac{1}{8} \sum_{i=1}^{8} \sqrt{\sum_{j=3i-2}^{3i} e_{linear,j}^2} \ . \tag{15}$$

Note that the losses of Eqs. 13 and 15 are considered to be functions of $\boldsymbol{w}$ only, i.e., with the residuals $\boldsymbol{r}_{gt}$ detached from gradient computation.

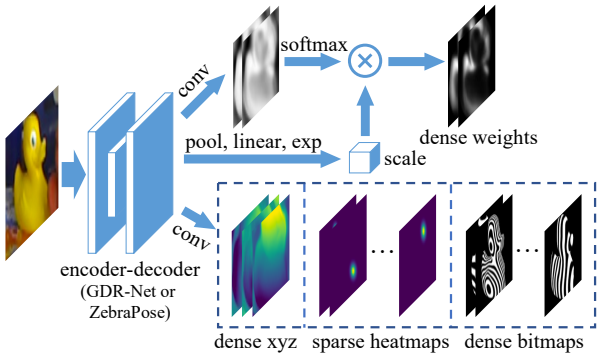


Figure 3. **Overall network structure.** Our experiments share a common pipeline, with differences in the detailed network structures and parameters. We also regress dense visibility masks in the dense correspondence cases, which is omitted here for simplicity.

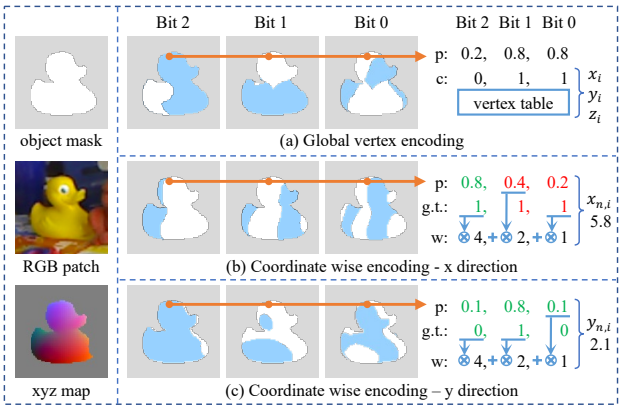


Figure 4. **Binary vertex encoding schemes.** (a) Global vertex encoding. Vertex codes are used as indices of the vertex table for coordinates lookup. (b) Coordinate wise encoding. Treating the vertex codes as normalized object coordinates, we keep the probability of the most significant mispredicted bit un-rounded and correct other bits before weighted summation. (c) If all predictions are correct, the least significant bit is kept un-rounded. In this figure, "p" is short for "probability", "c" is "code", "w" is "weight", and "g.t." is "ground-truth". We omit the z direction for simplicity.

Linear-Covariance Loss. We write the final Linear-Covariance (LC) loss following a Laplace NLL formalism. That is, we exploit the Laplace NLL loss, originating from the Laplace distribution and expressed as

$$L_{nll} = \log(b) + \frac{|x-u|}{b}, \tag{16}$$

where $|x-u|$ is the error of x, and b is a scale parameter acting as a prior prediction of the error. The Laplace NLL loss encourages a small error and a scale parameter equal to the error. In our context, it translates to the linear-covariance loss

$$L_{LC} = \log(E_{prior}) + 0.5 \cdot \frac{E_{cov} + E_{linear}}{E_{prior}}. \tag{17}$$

Ultimately, this loss seeks to minimize the sum of the covariance error and linear error, while encouraging the prior error to reflect this sum.

4. Experiments

We employ our linear-covariance loss in both dense and sparse correspondence-based methods. To this end, we use GDR-Net [46] as a first dense correspondence baseline, from which we build a competitive sparse correspondence baseline by replacing the GDR-Net output with 2D keypoint heatmaps. Furthermore, we exploit our linear-covariance loss in the ZebraPose [43] framework, allowing us to achieve state-of-the-art performance.

4.1. Network Structure

As illustrated by Fig. 3, the networks used in our experiments, whether based on GDR-Net or ZebraPose, take as input a cropped image region within a detected target bounding box. An encoder-decoder network is used to extract geometry feature maps from this input patch. The specific network structures follow those of the baseline methods. We use the same structure as EPro-PnP [7] for dense weights regression, obtaining the weights using a scaled spatial Softmax. For methods whose loss functions cannot fully supervise the weights, we remove the scale branch, using the output of the Softmax as weights, which further constrains the weights to sum up to 1. For dense correspondence methods, the dense xyz or dense bitmaps are used for correspondence extraction with weights from the dense weights channel. The visibility mask channel is employed for correspondence selection. For sparse correspondence methods, we only keep sparse heatmaps, each of which corresponds to a specific 2D keypoint. As the network acts on detected target image patches, we use the same setting of GDR-Net and Zebrapose, which is mainly based on Faster R-CNN [17] and FCOS [45].

4.2. Implementation Details

Correspondence Extraction. With GDR-Net, dense correspondences can be directly obtained from the 3D coordinates and associated weights output by the network. When weights are not available, the correspondences are selected based on the visibility mask. In the sparse case, keypoint locations and their standard deviations are estimated from the heatmaps with a DSNT [34] layer. We use the inverse of the standard deviations of the 2D keypoint predictions as weights.

Coordinate-wise Binary Encoding. As illustrated in Fig. 4 (a), the original ZebraPose recursively subdivides the object surface N times in a coarse-to-fine manner, generating N binary bitmaps. For any target pixel, a binary code of length N is obtained by concatenating the binary predic-

tion in each bitmap. The vertex 3D coordinates are then retrieved by indexing a predefined vertex table with the binary codes.

As the lookup operation is not differentiable. We instead assign a code for each component of the vertex 3D coordinates, the code is directly treated as normalized coordinate of the component. Specifically, a coordinate $c \in \{x, y, z\}$ in the range $[c_{min}, c_{max}]$ is normalized to c_n in $[0, 2^{M_c} - 1]$, i.e., to the representable range of M_c bit integers, via a linear transform. The binary representation of integer $round(c_n)$ is directly used as the binary encoding of c. As a result, a single vertex is expressed with 3 binary codes, with no lookup operation involved.

To recover the normalized coordinates with available ground-truth, as illustrated by Fig. 4 (b), instead of directly computing the weighted sum of predicted bits based on their significance, we first check their correctness and find the most significant mispredicted bit. This bit is substituted with the predicted unrounded probability. All other mispredicted bits are corrected before weighted summation, since their errors are negligible compared to the most significant erroneous bit. If all bits are correct or no ground truth available, the probability of the least significant bit is kept unrounded. In this way, the coordinates are differentiable, at the cost of more bits to predict, and the benefits of binary prediction are preserved.

Training Details. Following the baselines' strategy, we train a separate model from an ImageNet-pretrained backbone for each object with same optimizer, scheduler, and training epoch or step count as the baseline methods for fair comparison. The original loss functions for correspondence learning are kept. Specifically, GDR-Net [46] penalizes the L1 difference between a regressed normalized 3D coordinate map $\hat{M}_{XYZ}$ and the ground truth M_{XYZ} within the visible region M_{vis}. We use the binary cross entropy loss to train the visible region estimate $\hat{M}_{vis}$ and use $\hat{M}_{vis}$ for correspondence extraction if no weights are available. The full loss for the GDR-Net baseline is

$$\begin{aligned} L_{GDR-base} =& \|(\hat{M}_{XYZ} - M_{XYZ}) \odot M_{vis}\|_1 \\ &+ \alpha_{GDR} \cdot BCE(\hat{M}_{vis}, M_{vis}) \ , \end{aligned} \tag{18}$$

with $\alpha_{GDR} = 0.25$ and $\odot$ denoting the element-wise product. For ZebraPose [43], the original losses for mask prediction and binary code learning are both kept, giving

$$L_{Zebra-base} = L_{mask} + \alpha_{Zebra} \cdot L_{hier} \ , \tag{19}$$

in which L_{mask} is the L1 loss for object mask prediction, L_{hier} is a hierarchical loss for binary code learning and $\alpha_{Zebra} = 3$. The full loss when our LC loss is applied then is

$$L_{i-full} = L_{i-base} + \beta_i \cdot L_{LC}, \ i \in \{GDR, Zebra\}. \tag{20}$$

For the experiments in Sec. 4.4, we use $\beta_{GDR} = 0.02$ and $\beta_{Zebra} = 0.03$.

The pose loss is fully applied shortly after training begins since the network generates random correspondences at the very start. As matrix inversion is involved during linearization, the loss may generate large gradients in corner cases. We implement a gradient clipper which tracks the magnitude of the gradients and clip overly large ones. For outlier robustness, we employ the Huber function [24] adaptively when evaluating the squared residuals in Eq. 9 and the squared weights involved in the linearization process.

Efficient Loss Computation. For efficiency, we compute the covariance matrix in the most compact 6D representation and transform it to our target representation. For a specific pose representation $\boldsymbol{y}^K$ with K components, which is transformed from its 6D version $\boldsymbol{y}^6$ as $\boldsymbol{y}^K = f(\boldsymbol{y}^6)$, we calculate the Jacobian matrix J of f w.r.t. $\boldsymbol{y}^6$. Given the covariance matrix $C^{6\times6}$ of $\boldsymbol{y}^6$, the covariance of $\boldsymbol{y}^K$ can be calculated as $C^{K\times K} = J \cdot C^{6\times6} \cdot J^T$; only its diagonal elements are evaluated to compute the loss. Given N correspondences, the complexity of our loss can be reduced from $O(K^2N) + O(K^3)$ to $O(6^2N) + O(6^3) + O(K)$. On a single NVIDIA A100 GPU, with the LC loss, the GDR-Net based network takes around 3 hours on LM-O and 1 hour on YCB-V for a single object, and the ZebraPose based network takes around 24 hours for a single object on both datasets.

4.3. Datasets and Metrics

Datasets. We evaluate our approach on the widely used Linemod-Occluded (LM-O) [2] and YCB-Video (YCB-V) [49] datasets. LM-O is an extension of the Linemod (LM) [18] dataset. It has a total of 1214 images annotated for 8 objects under severe occlusion and is only used for testing. About 1.2k images for each object from the LM dataset are used as real training images. YCB-V is a large challenging video dataset containing about 133k images with strong occlusions and clutter. It contains 21 objects, some of which are symmetric or texture-less. We also adopt the publicly available physically-based rendering (pbr) [11] images for training.

Metrics. We employ the widely used ADD(-S) [18] score to compare with other methods. Under this metric, a pose is correct if the average distance between the object vertices transformed by the predicted pose and by the ground-truth pose is lower than a threshold. ADD-S is used for symmetry objects, replacing the average distance by the average nearest distance. The ADD(-S) score is computed with the threshold corresponding to 10% of the object diameter. For the YCB-V dataset, we also report the AUC (area under the curve) of the ADD(-S), varying the threshold up to a maximum of 10 cm.

As the ADD(-S) metric is defined in 3D space, in our

Method	Training Data	ADD(-S)
RePOSE [25]	real+syn	51.6
RNNPose [50]	real+syn	60.65
SO-Pose [12]	real+pbr	62.3
DProST [36]	real+pbr	62.6
GDR-Net [46]	real+pbr	62.2
ZebraPose [43]	real+pbr	76.9
GDR-Net-LC	real+pbr	66.48
ZebraPose-LC	real+pbr	**78.06**

Table 1. **Comparison with the state of the art on LM-O.** The "LC" postfix indicates the LC loss is applied.

Method	ADD(-S)	AUC of ADD-S	AUC of ADD(-S)
RePose [25]	62.1	88.5	82.0
RNNPose [50]	66.4	-	83.1
SO-Pose [12]	56.8	*90.9	*83.9
DProST [36]	65.1	-	77.4
GDR-Net [46]	60.1	*91.6	*84.4
ZebraPose [43]	80.5	90.1	85.3
GDR-Net-LC	70.6	89.8, *94.1	84.0, *88.8
ZebraPose-LC	**82.4**	**90.8**, ***95.0**	**86.1**, ***90.8**

Table 2. **Comparison with the state of the art on YCB-V.** * indicates that the AUC is calculated with 11-points interpolation.

Method	ADD(-S)	Correctness	Runtime
BPnP [6]	64.1	53.9	$\sim$ 30 ms
EPro-PnP [7]	64.5	59.3	$\sim$ 80 ms
Ours	**66.5**	**99.9**	$\sim$ **15** ms

Table 3. **Comparison between PnP layers on LM-O.**

ablations, we also report the Average Recall of Maximum Symmetry-Aware Projection Distance (MSPD) AR_{MSPD}, a metric from the BOP benchmark [20] based on 2D reprojection errors.

4.4. Comparison with the State of the Art

In this section, we compare the performance obtained when applying our loss to GDR-Net [46] and ZebraPose [43] with other start-of-the-art methods. We also compare our loss function with the state-of-the-art differentiable PnP methods, namely, BPnP [6] and EPro-PnP [7].

Results on LM-O. We report the ADD(-S) score on LM-O in Tab. 1. Applying the LC loss on GDR-Net surpasses most methods, and we achieve state-of-the-art performance when applying it to ZebraPose.

Results on YCB-V. As summarized in Tab. 2, applying the LC loss on GDR-Net produces results second only to ZebraPose. Furthermore, we achieve state-of-the-art performance when the LC loss is applied to ZebraPose. We implement a symmetry-aware training scheme which selects the ground-truth pose on the fly based on the average distance between the predicted 3D coordinates at randomly selected 2D locations and possible 3D coordinates at the same locations under the symmetric ground-truth pose. This scheme is only applied to the GDR-Net-based experiments on YCB-V, including the baseline and the model with our LC loss for fair comparison.

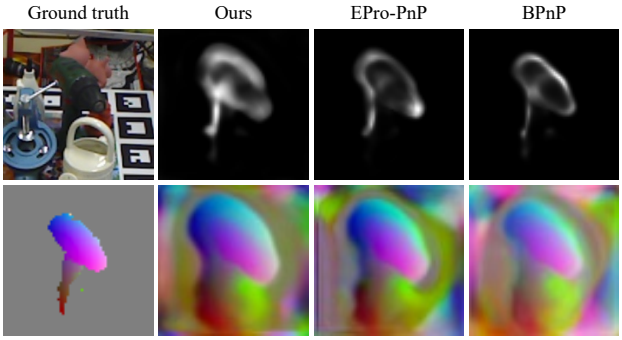


Figure 5. **Visualization of different PnP layers.** In the first column, we show the input image patch in the first row and the ground-truth object coordinates in the second row. The remaining images in the first row visualize the normalized weight maps for different methods. The remaining images in the second row visualize the predicted object coordinates corresponding to the weights in the first row.

Comparison with Differentiable PnP layers. As summarized in Tab. 3, we carry out experiments on LM-O with the GDR-Net baseline, and compare the methods based on several metrics including the ADD(-S) score, the gradient correctness and the runtime per training step, the correctness and runtime are evaluated at the end of training. Note that BPnP [6] does not fully constrain the weights, thus we remove the scale branch as stated in Sec. 4.1. Our method yields the best ADD(-S) score on LM-O. More importantly, it generates a much larger percentage of correct gradients. A 3D point is considered to have correct gradients if moving in the negative gradient direction leads to a smaller 2D reprojection error. A pose loss yielding a higher gradient correctness provides more consistent supervision signals. The consistency is reflected by the dilated weight and coordinate maps shown in Fig. 5, in particular by looking at pixels outside of but close to the actual object region. Such pixels receive supervision only from the pose loss and thus indicate the differences between the different pose losses. Higher correctness helps the network to predict correct correspondences for such pixels. This virtually expands the target object size in 3D object space and in 2D image space, which facilitates better pose estimates. The LC loss yields 99.9% gradient correctness, generating the most dilated maps. By contrast, the other losses have weaker consistency and thus tend to predict less accu-

rate correspondences in these regions. Finally, as shown in Tab. 3, our LC loss yields the fastest runtime, evaluated on an NVIDIA A100 GPU with a batch size of 32. This is due to our linearization of the PnP solver, removing the need for an iterative solution.

4.5. Ablation Study on LM-O

We evaluate the influence of each component of our LC loss by applying it to the dense correspondence-based GDR-Net and to the sparse correspondence-based one on the LM-O dataset. The results are summarized in Tab. 4 and Tab. 5.

Effectiveness of the Pose Representation. We validate the ability of the LC loss at integrating different application preferences by switching to a pose representation defined in the 2D image space. Similarly to the 3D case, a pose representation $\boldsymbol{y}^{2D} \in \mathbb{R}^{16}$ can be defined as the projected 2D coordinates of the 8 object bounding box corners. Switching from the LC loss in 3D space to its 2D space counterpart (B0 *vs.* B1) causes a slight ADD(-S) drop. We further compare the MSPD scores between these two versions. The network trained with 2D space LC loss yields an AR_{MSPD} score of 84.14, better than the 83.98 of the 3D case. This validates the effectiveness of using a metric aware pose representation for different applications.

Effectiveness of the Covariance Loss. The covariance loss is designed for both correspondence learning and weight learning. It aims to minimize the residuals of the correspondences, and encourages the weights to be inversely proportional to the residuals. We investigate its effectiveness by detaching, in turn, the residuals $\boldsymbol{r}_{gt}$ and weights $\boldsymbol{w}$ from the covariance loss. When the residuals are detached (B0 *vs.* C0), the only loss for correspondence learning is the original surrogate loss, which lacks supervision on the background pixels. Thus the learned weights and coordinates are restricted to the visible part of the target. When the weights are detached (B0 *vs.* C1), the network yields over-concentrated weights lying outside of the object region. This is evidenced by Fig. 6 (b); although the network learns these emphasized correspondences correctly, its performance still degrades compared to our approach. If the covariance loss is entirely removed (B0 *vs.* C2), the pipeline suffers from a large performance drop.

Effectiveness of the Linear Loss. The linear loss is a linear approximation of the actual pose error. As it acts on the weights, it seeks to emphasizes the correspondences that are more beneficial for the pose. As shown in Tab. 4, after removing E_{linear} from the LC loss, the ADD(-S) score drops slightly (B0 *vs.* C3). However, this makes a significant difference on the learned coordinate and weight maps; as illustrated in Fig. 6 (c), when trained without linear loss, the network tends to confidently extrapolate the 3D coordinates to pixels far away from the target region or occluded.

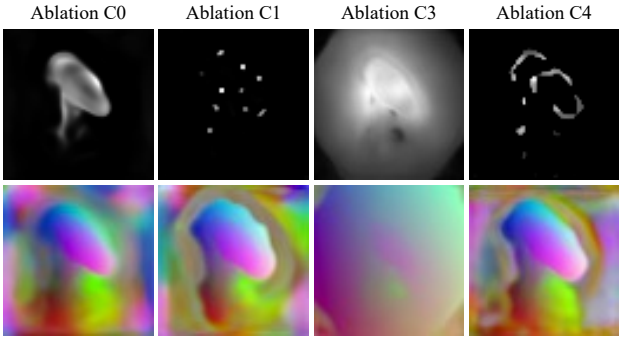


Figure 6. **Visualizations for our different ablations.** We show the visualizations of the same input patch as in Fig. 5. The first row visualizes the normalized weight maps for different settings. The second row shows the predicted object coordinates corresponding to the weights.

Row	Method	ADD(-S)
A0	GDR-Net baseline	59.29
A1	GDR-Net [46]	62.2
B0	A0 + 3D LC loss	**66.48**
B1	A0 + 2D LC loss	65.99
C0	B0 + detach residual from E_{cov}	61.03
C1	B0 + detach weights from E_{cov}	65.23
C2	B0 + remove E_{cov}	45.61
C3	B0 + remove E_{linear}	65.82
C4	B0 + remove E_{prior}	60.69

Table 4. **Ablation study of the dense correspondence-based method on LM-O.** Block A: Comparison of the baseline method and the original method. Block B: Comparison between losses derived from different pose representations. Block C: Ablations of each component of the linear-covariance loss on the dense correspondence-based baseline.

Such extrapolated coordinates are unreliable compared to correspondences near or inside object region, and thus they are suppressed by the linear loss.

Effectiveness of the Prior Loss. Together with the other loss terms, the prior loss fully constrains the weights, and allows for fine-grained weights learning. When the prior loss is removed (B0 *vs.* C4), we also remove the scale branch as stated in Sec. 4.1. As illustrated by Fig. 6 (d), the network also learns concentrated weights but with a larger performance drop.

Effectiveness with a Sparse Correspondence-based Method. In the dense case, the weights serve as both attention mechanism, emphasizing some important or stable points during training, and indicators for well learned correspondences during testing. The sparsity in sparse correspondence methods limits the attention feature. The loss function in sparse cases mostly encourages the network to predict better weights. Our sparse baseline is trained with

Row	Training Loss	Solver	ADD(-S)
0	MLE	PnP RANSAC	57.89
1	MLE	PnP weighted	59.90
2	MLE + LC loss	PnP RANSAC	57.74
3	MLE + LC loss	PnP weighted	**61.08**

Table 5. **Ablation for the sparse correspondence-based method on the LM-O dataset.**

dist.\ rep.	ours-3D	quater.	axis-ang.	two-col. [52]
Laplace	66.48	66.72	65.85	66.39
Gaussian	61.90	60.72	59.79	60.37

Table 6. **Results on different pose representations and NLL distributions in ADD(-S) score.**

a Laplace MLE loss, similar to the Gaussian MLE loss in [33]. The predicted standard deviations are encouraged to capture point location errors, and their inverse are subsequently used as weights in the PnP solver. As shown in Tab. 5, applying our loss in this scenario also brings a performance gain. As the networks yield very close performance when not using weights (Row 0 *vs.* Row 2), this gain comes mostly from better weights learning. Note that the *PnP RANSAC* solver does not use weights but uses RANSAC [16] to evict outliers, thus reflecting only the quality of 2D point locations. The *PnP weighted* solver iteratively solves Eq. 4 using the *PnP RANSAC*'s solution as starting point, which effectively relies on predicted weights to evict outliers, reflecting the quality of the predicted weights.

Experiments on other pose representations and other NLL formulations. We also carried out experiments on losses based on quaternions, axis-angles and two-column [52] pose representations on LM-O. For the Laplace NLL formulation, we used the sum of the square roots of the covariance diagonal for $\{E_{cov}, E_{prior}\}$ and the sum of the absolute values of the approximate error for E_{linear}. To further validate the distribution from which the NLL loss originates, we switched from Laplace to Gaussian, using the trace of the covariance and the sum of the squared errors to build the losses. As shown in Tab. 6, the performance is robust to the pose representation. However, the Laplace NLL losses yield much better results than their Gaussian counterparts.

5. Conclusion

In this work, we have proposed a linear-covariance loss for end-to-end weights and correspondence learning of two stage geometry-based 6D pose estimation networks. This new loss function addresses the problem originating from the averaging nature of PnP solvers, resulting in gradients that may seek to degrade the accuracy of some correspondences. At the heart of our loss is the idea of introducing ground-truth information by linearizing the PnP solver around the ground truth before the pose is actually solved, and building the loss function for correspondence learning on the covariance of the pose distribution. Our extensive experiments have validated the effectiveness of our LC loss on both sparse and dense correspondence-based methods and on two standard benchmarks.

Nevertheless, the LC loss cannot learn correspondences from scratch, and a surrogate loss function to supervise the correspondences remains necessary for providing the LC loss with an initial object 3D structure. In the future, we will seek to apply our loss function to category-level object pose estimation [8, 13, 47], where precise object structure is not available.

Acknowledgement. This work was supported in part by China Scholarship Council (CSC) Grant 202006020218, and in part by the National Natural Science Foundation of China (NSFC) under Grant 52127809 and Grant 51625501.

6. Supplementary

6.1. Linearization of PnP Solver

Implicit Function Theorem. The implicit function theorem (IFT) [27] states the following:

Given $f : \mathbb{R}^{n+m} \to \mathbb{R}^m$ 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

$$f(\boldsymbol{a}^*, \boldsymbol{b}^*) = \mathbf{0} \ , \tag{21}$$

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

$$\boldsymbol{b}^* = g(\boldsymbol{a}^*) \ , \tag{22}$$

and

$$f(\boldsymbol{a}^*, g(\boldsymbol{a}^*)) = \mathbf{0} \ . \tag{23}$$

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

$$\frac{\partial g}{\partial \boldsymbol{a}}(\boldsymbol{a}^*) = -\left[\frac{\partial f}{\partial \boldsymbol{b}}(\boldsymbol{a}^*, \boldsymbol{b}^*)\right]^{-1} \cdot \frac{\partial f}{\partial \boldsymbol{a}}(\boldsymbol{a}^*, \boldsymbol{b}^*) \ . \tag{24}$$

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

$$g(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{w}) = \arg\min_{\boldsymbol{y}} \frac{1}{2} \sum_i^N \left\| \boldsymbol{w}_i \circ \boldsymbol{r}_i \right\|^2 \ , \tag{25}$$

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

$$\boldsymbol{r}_i = \boldsymbol{x}_i - \pi(\boldsymbol{z}_i, \boldsymbol{y}) \tag{26}$$

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

Eq. 25 implies that the solution $\boldsymbol{y}^*$ is the stationary point of the negative log likelihood (NLL) function

$$nll(\boldsymbol{y}) = \frac{1}{2}\sum_{i}^{N} \|\boldsymbol{w}_i \circ \boldsymbol{r}_i\|^2 \ . \tag{27}$$

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.,

$$\frac{\partial nll(\boldsymbol{y})}{\partial \boldsymbol{y}}\bigg|_{\boldsymbol{y}=\boldsymbol{y}^*} = \boldsymbol{0} \ . \tag{28}$$

Eqs. 21, 22 and 23 in the PnP case can subsequently be specialized as

$$f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{w})|_{\boldsymbol{y}=\boldsymbol{y}^*} = \frac{\partial nll(\boldsymbol{y})}{\partial \boldsymbol{y}}\bigg|_{\boldsymbol{y}=\boldsymbol{y}^*} = \boldsymbol{0} \ , \tag{29}$$

$$\boldsymbol{y}^* = g(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{w}) \ , \tag{30}$$

and

$$f(\boldsymbol{x}, g(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{w}), \boldsymbol{z}, \boldsymbol{w})|_{\boldsymbol{y}=\boldsymbol{y}^*} = \boldsymbol{0} \ . \tag{31}$$

According to Eq. 24, the gradient of the pose $\boldsymbol{y}$ w.r.t. the 2D locations $\boldsymbol{x}$ at $\boldsymbol{y}^*$ is

$$\begin{aligned} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\bigg|_{\boldsymbol{y}^*} &= \frac{\partial g(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{w})}{\partial \boldsymbol{x}}\bigg|_{\boldsymbol{y}^*} \ , \\ &= -\left[\left[\frac{\partial^2 nll(\boldsymbol{y})}{\partial \boldsymbol{y}^2}\right]^{-1} \cdot \frac{\partial^2 nll(\boldsymbol{y})}{\partial \boldsymbol{y} \partial \boldsymbol{x}}\right]\bigg|_{\boldsymbol{y}*} \ , \\ &= -H^{-1} \cdot \frac{\partial^2 nll(\boldsymbol{y})}{\partial \boldsymbol{y} \partial \boldsymbol{x}}\bigg|_{\boldsymbol{y}^*} \ , \end{aligned} \tag{32}$$

with $nll(\boldsymbol{y})$ defined by Eq. 27.

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

$$\boldsymbol{y} = \boldsymbol{y}_{gt} + A(\boldsymbol{z}, \boldsymbol{w}) \cdot \boldsymbol{r}_{gt} \ , \tag{33}$$

with

$$\boldsymbol{r}_{gt} = \boldsymbol{x} - \boldsymbol{x}_{gt} \tag{34}$$

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

$$A(\boldsymbol{z}, \boldsymbol{w}) = -H^{-1} \cdot \frac{\partial^2 nll(\boldsymbol{y})}{\partial \boldsymbol{y} \partial \boldsymbol{x}}\bigg|_{\boldsymbol{y}=\boldsymbol{y}_{gt}, \boldsymbol{x}=\boldsymbol{x}_p} \ . \tag{35}$$

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.

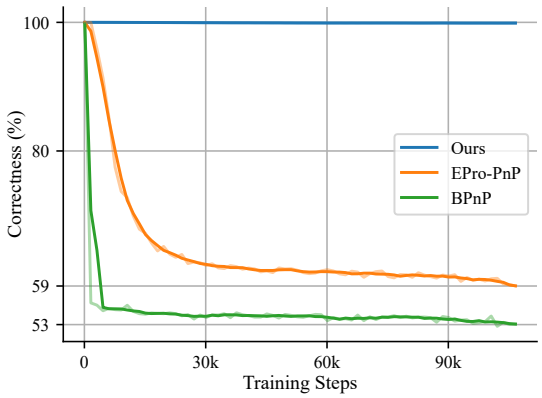


Figure 7. **Correctness curves of the PnP layers.** A 3D point is considered to have a correct gradient if moving in the negative gradient direction leads to a smaller 2D reprojection error. The LC loss yields almost 100% correctness. The correctness of EPro-PnP drops slowly, and ending with about 59% correctness. BPnP drops quickly when training begins, and ends with about 53% correctness. The dark curves are smoothed versions of the light ones.

Row	Method	ADD(-S)
A0	ZebraPose [43]	76.91
A1	ZebraPose baseline	75.19
A2	A1 + LC loss	**78.06**

Table 7. **Results of the ZebraPose [43] based experiments on the LM-O dataset.**

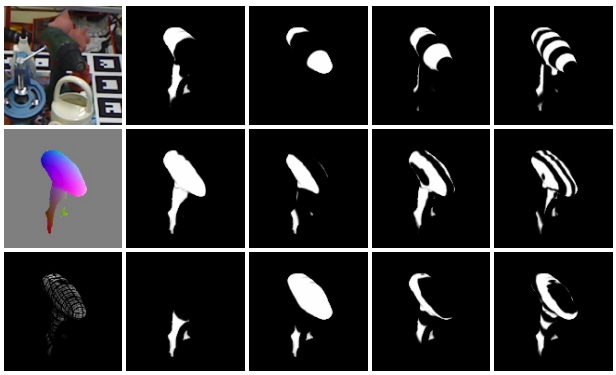
6.2. Detailed Results on Gradient Correctness

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

As illustrated in Fig. 7, at the very beginning, when the correspondences have large errors, both EPro-PnP [7] and BPnP [6] 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%.

6.3. 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 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.



(a) (b) (c) (d) (e)

Figure 8. **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.

Results. As shown in Tab. 7, 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 [43].

Visualizations. As illustrated by Fig. 8, 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.

6.4. 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. 9.

For the YCB-V dataset, we provide the detailed comparison of ADD(-S) scores (Tab. 8) and AUC scores (Tab. 10) between the baseline methods and the versions where the LC loss is applied.

Object	[46]	[43]	[46]-LC	[43]-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 8. **Detailed ADD(-S) scores on YCB-V.** We report the scores of the original baseline methods, GDR-Net [46] and ZebraPose [43], and also the scores after applying our LC loss, respectively (denoted by "-LC"). (*) denotes symmetric objects on which the ADD-S score is reported.

References

[1] Gideon Billings and Matthew Johnson-Roberson. Silhonet: An rgb method for 6d object pose estimation. *IEEE Robotics and Automation Letters*, 4(4):3727–3734, 2019. 1

[2] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 536–551, Cham, 2014. Springer International Publishing. 6

[3] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac - differentiable ransac for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1, 3

[4] Eric Brachmann and Carsten Rother. Learning less is more - 6d camera localization via 3d surface regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 3

[5] Bo Chen, Tat-Jun Chin, and Marius Klimavicius. Occlusion-robust object pose estimation with holistic representation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2929–2939, January 2022. 1, 2

[6] 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. 1, 3, 4, 7, 10

[7] 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. 1, 3, 5, 7, 10

[8] Wei Chen, Xi Jia, Hyung Jin Chang, Jinming Duan, Linlin Shen, and Ales Leonardis. Fs-net: Fast shape-based network for category-level 6d object pose estimation with decoupled rotation mechanism. In *Proceedings of the IEEE/CVF*

Method	RePOSE [25]	RNNPose [50]	SO-Pose [12]	DProST [36]	GDR-Net [46]	ZebraPose [43]	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 9. **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 [46] baseline, "Zebra-LC" denotes the LC loss with the ZebraPose [43] baseline.

Method	GDR-Net [46]		ZebraPose [43]		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 10. **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 [46] baseline, "ZebraPose-LC" denotes the LC loss with the ZebraPose [43] 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.

Conference on Computer Vision and Pattern Recognition (CVPR), pages 1581–1590, June 2021. 9

[9] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1

[10] Alberto Crivellaro, Mahdi Rad, Yannick Verdie, Kwang Moo Yi, Pascal Fua, and Vincent Lepetit. Robust 3d object tracking from monocular images using stable parts. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1465–1479, 2017. 1

[11] Maximilian Denninger, Dominik Winkelbauer, Martin Sundermeyer, Wout Boerdijk, Markus Knauer, Klaus H. Strobl, Matthias Humt, and Rudolph Triebel. Blenderproc2: A procedural pipeline for photorealistic rendering. *Journal of Open Source Software*, 8(82):4901, 2023. 6

[12] 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. 3, 7, 12

[13] Yan Di, Ruida Zhang, Zhiqiang Lou, Fabian Manhardt, Xiangyang Ji, Nassir Navab, and Federico Tombari. Gpv-pose: Category-level object pose estimation via geometry-guided point-wise voting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6781–6791, June 2022. 9

[14] Thanh-Toan Do, Ming Cai, Trung Pham, and Ian Reid. Deep-6dpose: Recovering 6d object pose from a single rgb image, 2018. 1

[15] Thanh-Toan Do, Trung T. Pham, Mingpeng Cai, and Ian D. Reid. Lienet: Real-time monocular object instance 6d pose estimation. In *British Machine Vision Conference*, 2018. 4

[16] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 9

[17] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015. 5

[18] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2012. 6

[19] Tomas Hodan, Daniel Barath, and Jiri Matas. Epos: Estimating 6d pose of objects with symmetries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3

[20] Tomas Hodan, Frank Michel, Eric Brachmann, Wadim Kehl, Anders GlentBuch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, Caner Sahin, Fabian Manhardt, Federico Tombari, Tae-Kyun Kim, Jiri Matas, and Carsten Rother. Bop: Benchmark for 6d object pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 7

[21] Omid Hosseini Jafari, Siva Karthik Mustikovela, Karl Pertsch, Eric Brachmann, and Carsten Rother. ipose: Instance-aware 6d pose estimation of partly occluded objects. In C. V. Jawahar, Hongdong Li, Greg Mori, and Konrad Schindler, editors, *Computer Vision – ACCV 2018*, pages 477–492, Cham, 2019. Springer International Publishing. 1, 2

[22] Yinlin Hu, Pascal Fua, Wei Wang, and Mathieu Salzmann. Single-stage 6d object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 3

[23] Yinlin Hu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. Segmentation-driven 6d object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 2

[24] Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101, 1964. 6

[25] 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. 7, 12

[26] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 1

[27] Steven George Krantz and Harold R Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002. 3, 4, 9

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

[29] 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)*, September 2018. 4

[30] Zhigang Li, Gu Wang, and Xiangyang Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2, 3

[31] Fabian Manhardt, Wadim Kehl, Nassir Navab, and Federico Tombari. Deep model-based 6d pose refinement in rgb. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 4

[32] Eric Marchand, Hideaki Uchiyama, and Fabien Spindler. Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics*, 22(12):2633–2651, 2015. 1

[33] Nathaniel Merrill, Yuliang Guo, Xingxing Zuo, Xinyu Huang, Stefan Leutenegger, Xi Peng, Liu Ren, and Guoquan Huang. Symmetry and uncertainty-aware object slam for 6dof object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14901–14910, June 2022. 1, 9

[34] Aiden Nibali, Zhen He, Stuart Morgan, and Luke Prendergast. Numerical coordinate regression with convolutional neural networks. *arXiv preprint arXiv:1801.07372*, 2018. 5

[35] Markus Oberweger, Mahdi Rad, and Vincent Lepetit. Making deep heatmaps robust to partial occlusions for 3d object pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 1, 2

[36] 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, 12

[37] Keunhong Park, Arsalan Mousavian, Yu Xiang, and Dieter Fox. Latentfusion: End-to-end differentiable reconstruction

and rendering for unseen object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 4

[38] Kiru Park, Timothy Patten, and Markus Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1, 2

[39] Georgios Pavlakos, Xiaowei Zhou, Aaron Chan, Konstantinos G Derpanis, and Kostas Daniilidis. 6-dof object pose from semantic keypoints. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2011–2018. IEEE, 2017. 1, 2

[40] Yudi Pawitan. *In all likelihood: statistical modelling and inference using likelihood*. Oxford University Press, 2001. 4

[41] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 2

[42] Mahdi Rad and Vincent Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 1, 2

[43] 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. 1, 2, 3, 5, 6, 7, 10, 11, 12

[44] Bugra Tekin, Sudipta N. Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 2

[45] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 5

[46] 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. 3, 5, 6, 7, 8, 11, 12

[47] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 9

[48] Di Wu, Zhaoyong Zhuang, Canqun Xiang, Wenbin Zou, and Xia Li. 6d-vnet: End-to-end 6-dof vehicle pose estimation from monocular rgb images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019. 1

[49] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. 2018. 1, 6

[50] 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. 7, 12

[51] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Dpod: 6d pose object detector and refiner. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 3

[52] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 4, 9

[53] Menglong Zhu, Konstantinos G Derpanis, Yinfei Yang, Samarth Brahmbhatt, Mabel Zhang, Cody Phillips, Matthieu Lecce, and Kostas Daniilidis. Single image 3d object detection and pose estimation for grasping. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3936–3943. IEEE, 2014. 1

[54] Yiming Zuo, Weichao Qiu, Lingxi Xie, Fangwei Zhong, Yizhou Wang, and Alan L. Yuille. Craves: Controlling robotic arm with a vision-based economic system. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1