HybridPose: 6D Object Pose Estimation under Hybrid Representations

Chen Song[*], Jiaru Song[*], Qixing Huang
The University of Texas at Austin

song@cs.utexas.edu, jiarus@cs.utexas.edu, huangqx@cs.utexas.edu

Abstract

We introduce HybridPose, a novel 6D object pose estimation approach. HybridPose utilizes a hybrid intermediate representation to express different geometric information in the input image, including keypoints, edge vectors, and symmetry correspondences. Compared to a unitary representation, our hybrid representation allows pose regression to exploit more and diverse features when one type of predicted representation is inaccurate (e.g., because of occlusion). Different intermediate representations used by HybridPose can all be predicted by the same simple neural network, and outliers in predicted intermediate representations are filtered by a robust regression module. Compared to state-of-the-art pose estimation approaches, HybridPose is comparable in running time and is significantly more accurate. For example, on Occlusion Linemod [3] dataset, our method achieves a prediction speed of 30 fps with a mean ADD(-S) accuracy of 79.2%, representing a 67.4% improvement from the current state-of-the-art approach. The implementation of HybridPose is available at https://github.com/chensong1995/HybridPose.

1. Introduction

Estimating the 6D pose of an object from an RGB image is a fundamental problem in 3D vision and has diverse applications in object recognition and robot-object interaction. Advances in deep learning have led to significant breakthroughs in this problem. While early works typically formulate pose estimation as end-to-end pose classification [39] or pose regression [16, 42], recent pose estimation methods usually leverage keypoints as an intermediate representation [38, 34], and align predicted 2D keypoints with ground-truth 3D keypoints. In addition to ground-truth pose labels, these methods incorporate keypoints as an intermediate supervision, facilitating smooth model training. Keypoint-based methods are built upon two assumptions: (1) a machine learning model can accurately predict 2D keypoint locations; and (2) these predictions provide suf-
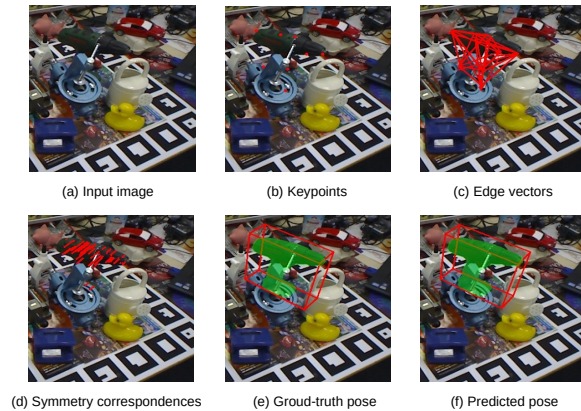

Figure 1. HybridPose predicts keypoints, edge vectors, and symmetry correspondences. In (a), we show the input RGB image, in which the object of interest (driller) is partially occluded. In (b), red markers denote predicted 2D keypoints. In (c), edge vectors are defined by a fully-connected graph among all keypoints. In (d), symmetry correspondences connect each 2D pixel on the object to its symmetric counterpart. For illustrative purposes, we only draw symmetry correspondences of 50 random samples from 5755 detected object pixels in this example. The predicted pose (f) is obtained by jointly aligning all predictions with the 3D template, which involves solving a non-linear optimization problem.

ficient constraints to regress the underlying 6D pose. Both assumptions easily break in many real-world settings. Due to object occlusions and representational limitations of the prediction network, it is often impossible to accurately predict 2D keypoint coordinates from an RGB image alone.

In this paper, we introduce HybridPose, a novel 6D pose estimation approach that leverages multiple intermediate representations to express the geometric information in the input image. In addition to keypoints, HybridPose integrates a prediction network that outputs edge vectors between adjacent keypoints. As most objects possess a (partial) reflection symmetry, HybridPose also utilizes predicted dense pixel-wise correspondences that reflect the underlying symmetric relations between pixels. Compared to a unitary representation, this hybrid representation enjoys a multitude of advantages. First, HybridPose integrates more signals in the input image: edge vectors encode spa-

cial relations among object parts, and symmetry correspondences incorporate interior details. Second, HybridPose offers more constraints than using keypoints alone for pose regression, enabling accurate pose prediction even if a significant fraction of predicted elements are outliers (e.g., because of occlusion). Finally, it can be shown that symmetry correspondences stabilize the rotation component of pose prediction, especially along the normal direction of the reflection plane (details are provided in the supp. material).

Given the intermediate representation predicted by the first module, the second module of HybridPose performs pose regression. In particular, HybridPose employs trainable robust norms to prune outliers in predicted intermediate representation. We show how to combine pose initialization and pose refinement to maximize the quality of the resulting object pose. We also show how to train HybridPose effectively using a training set for the pose prediction module, and a validation set for the pose regression module.

We evaluate HybridPose on two popular benchmark datasets, Linemod [12] and Occlusion Linemod [3]. In terms of accuracy (under the ADD(-S) metric), HybridPose leads to considerable improvements from all state-of-the-art methods that merely utilize keypoints. On Occlusion Linemod [3], HybridPose achieves an accuracy of 79.2%, which represents a 67.4% improvement from DPOD [44], the current state-of-the-art method on this benchmark dataset.

Despite the gain in accuracy, our approach is efficient and runs at 30 frames per second on a commodity workstation. Compared to approaches which utilize sophisticated network architecture to predict one single intermediate representation (such as Pix2Pose [30]), HybridPose achieves considerably better performance by using a relative simple network to predict hybrid representations.

2. Related Works

Intermediate representation for pose. To express the geometric information in an RGB image, a prevalent intermediate representation is keypoints, which achieves state-of-the-art performance [34, 32, 36]. The corresponding pose estimation pipeline combines keypoint prediction and pose regression initialized by the PnP algorithm [18]. Keypoint predictions are usually generated by a neural network, and previous works use different types of tensor descriptors to express 2D keypoint coordinates. A common approach represents keypoints as peaks of heatmaps [28, 48], which becomes sub-optimal when keypoints are occluded, as the input image does not provide explicit visual cues for their locations. Alternative keypoint representations include vector-fields [34] and patches [14]. These representations allow better keypoint predictions under occlusion, and eventually lead to improvement in pose estimation accuracy. However, keypoints alone are a sparse representation of the

object pose, whose potential in improving estimation accuracy is limited.

Besides keypoints, another common intermediate representation is the coordinate of every image pixel in the 3D physical world, which provides dense 2D-3D correspondences for pose alignment, and is robust under occlusion [3, 4, 30, 20]. However, regressing dense object coordinates is much more costly than keypoint prediction. They are also less accurate than keypoints due to the lack of corresponding visual cues. In addition to keypoints and pixel-wise 2D-3D correspondences, depth is another alternative intermediate representation in visual odometry settings, which can be estimated together with pose in an unsupervised manner [47]. In practice, the accuracy of depth estimation is limited by the representational power of neural networks.

Unlike previous approaches, HybridPose combines multiple intermediate representations, and exhibits collaborative strength for pose estimation.

Multi-modal input. To address the challenges for pose estimation from a single RGB image, several works have considered inputs from multiple sensors. A popular approach is to leverage information from both RGB and depth images [47, 40, 42]. In the presence of depth information, pose regression can be reformulated as the 3D point alignment problem, which is then solved by the ICP algorithm [42]. Although HybridPose utilizes multiple intermediate representations, all intermediate representations are predicted from an RGB image alone. HybridPose handles situations in which depth information is absent.

Edge features. Edges are known to capture important image features such as object contours [2], salient edges [23], and straight line segments [45]. Unlike these low-level image features, HybridPose leverages semantic edge vectors defined between adjacent keypoints. This representation, which captures correlations between keypoints and reveals underlying structure of object, is concise and easy to predict. Such edge vectors offer more constraints than keypoints alone for pose regressions and have clear advantages under occlusion. Our approach is similar to [5], which predicts directions between adjacent keypoints to link keypoints into a human skeleton. However, we predict both the direction and the magnitude of edge vectors, and use these vectors to estimate object poses.

Symmetry detection from images. Symmetry detection has received significant attention in computer vision. We refer readers to [22, 27] for general surveys, and [1, 41] for recent advances. Traditional applications of symmetry detection include face recognition [31], depth estimation [21], and 3D reconstruction [13, 43]. In the context of object pose estimation, people have studied symmetry from the perspective that it introduces ambiguities for pose estimation (c.f. [25, 36, 42]), since symmetric objects with different poses can have the same appearance in image. Several

(a) Input image

(b) Predicted keypoints, edge vectors, and symmetry correspondences

Initialization sub-module

Refinement sub-module
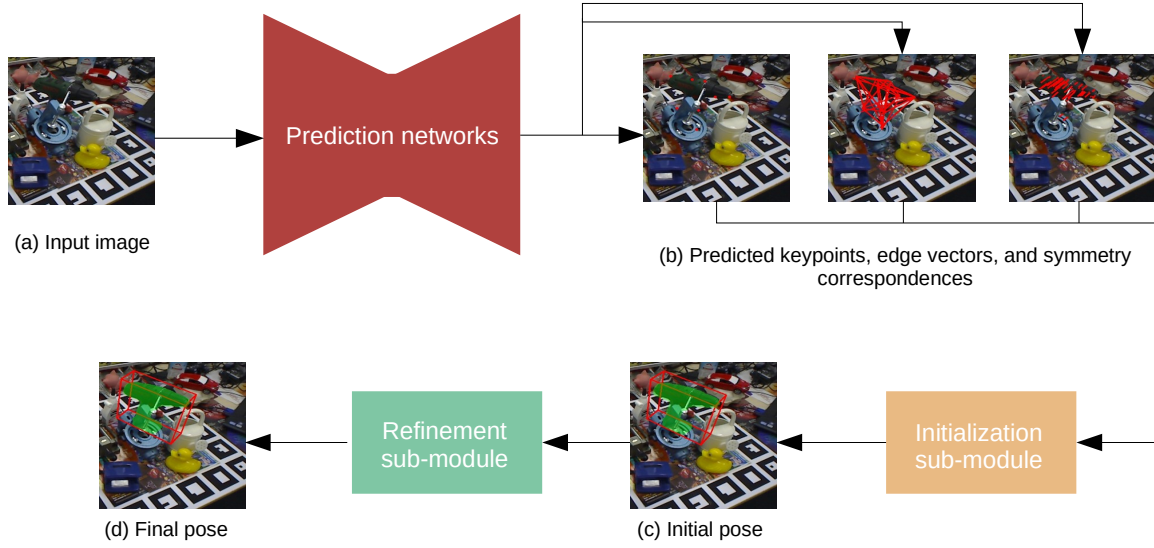
(d) Final pose

(c) Initial pose

Figure 2. **Approach overview.** HybridPose consists of intermediate representation prediction networks and a pose regression module. The prediction networks take an image as input, and output predicted keypoints, edge vectors, and symmetry correspondences. The pose regression module consists of a initialization sub-module and a refinement sub-module. The initialization sub-module solves a linear system with predicted intermediate representations to obtain an initial pose. The refinement sub-module utilizes GM robust norm and optimizes (9) to obtain the final pose prediction.

works [36, 42, 6, 25, 30] have explored how to address such ambiguities, e.g., by designing loss functions that are invariant under symmetric transformations.

Robust regression. Pose estimation via intermediate representation is sensitive to outliers in predictions, which are introduced by occlusion and cluttered backgrounds [37, 32, 40]. To mitigate pose error, several works assign different weights to different predicted elements in the 2D-3D alignment stage [34, 32]. In contrast, our approach additionally leverages robust norms to automatically filter outliers in the predicted elements.

Besides the reweighting strategy, some recent works propose to use deep learning-based refiners to boost the pose estimation performance [19, 26, 44]. [44, 19] use point matching loss and achieve high accuracy. [26] predicts pose updates using contour information. Unlike these works, our approach considers the critical points and the loss surface of the robust objective function, and does not involve a fixed pre-determined iteration count used in recurrent network based approaches.

3. Approach

The input to HybridPose is an image I containing an object in a known class, taken by a pinhole camera with known intrinsic parameters. Assuming that the class of objects has a canonical coordinate system Σ (i.e. the 3D point cloud), HybridPose outputs the 6D camera pose ($R_I \in SO(3), t_I \in \mathbb{R}^3$) of the image object under Σ, where R_I is the rotation and t_I is the translation component.

3.1. Approach Overview

As illustrated in Figure 2, HybridPose consists of a prediction module and a pose regression module.

Prediction module (Section 3.2). HybridPose utilizes three prediction networks $f_\theta^{\mathcal{K}}$, $f_\phi^{\mathcal{E}}$, and $f_\gamma^{\mathcal{S}}$ to estimate a set of keypoints $\mathcal{K} = \{p_k\}$, a set of edges between keypoints $\mathcal{E} = \{v_e\}$, and a set of symmetry correspondences between image pixels $\mathcal{S} = \{(q_{s,1}, q_{s,2})\}$. $\mathcal{K}$, $\mathcal{E}$, and $\mathcal{S}$ are all expressed in 2D. θ, ϕ, and γ are trainable parameters.

The keypoint network $f_\theta^{\mathcal{K}}$ employs an off-the-shelf prediction network [34]. The other two prediction networks, $f_\phi^{\mathcal{E}}$, and $f_\gamma^{\mathcal{S}}$, are introduced to stabilize pose regression when keypoint predictions are inaccurate. Specifically, $f_\phi^{\mathcal{E}}$ predicts edge vectors along a pre-defined graph of keypoints, which stabilizes pose regression when keypoints are cluttered in the input image. $f_\gamma^{\mathcal{S}}$ predicts symmetry correspondences that reflect the underlying (partial) reflection symmetry. A key advantage of this symmetry representation is that the number of symmetry correspondences is large: every image pixel on the object has a symmetry correspondence. As a result, even with a large outlier ratio, symmetry correspondences still provide sufficient constraints for estimating the plane of reflection symmetry for regularizing the underlying pose. Moreover, symmetry correspondences incorporate more features within the interior of the underlying object than keypoints and edge vectors.

Pose regression module (Section 3.3). The second module of HybridPose optimizes the object pose (R_I, t_I) to fit the output of the three prediction networks. This module combines a trainable initialization sub-module and a train-

able refinement sub-module. In particular, the initialization sub-module performs SVD to solve for an initial pose in the global affine pose space. The refinement sub-module utilizes robust norms to filter out outliers in the predicted elements for accurate object pose estimation.

Training HybridPose (Section 3.4). We train HybridPose by splitting the dataset into a training set and a validation set. We use the training set to learn the prediction module, and the validation set to learn the hyper-parameters of the pose regression module. We have tried training HybridPose end-to-end using one training set. However, the difference between the prediction distributions on the training set and testing set leads to sub-optimal generalization performance.

3.2. Hybrid Representation

This section describes three intermediate representations used in HybridPose.

Keypoints. The first intermediate representation consists of keypoints, which have been widely used for pose estimation. Given the input image I, we train a neural network $f_\theta^{\mathcal{K}}(I) \in \mathbb{R}^{2 \times |\mathcal{K}|}$ to predict 2D coordinates of a pre-defined set of $|\mathcal{K}|$ keypoints. In our experiments, HybridPose incorporates an off-the-shelf architecture called PVNet [34], which is the state-of-the-art keypoint-based pose estimator that employs a voting scheme to predict both visible and invisible keypoints.

Besides outliers in predicted keypoints, another limitation of keypoint-based techniques is that when the difference (direction and distance) between adjacent keypoints characterizes important information of the object pose, inexact keypoint predictions incur large pose error.

Edges. The second intermediate representation, which consists of edge vectors along a pre-defined graph, explicitly models the displacement between every pair of keypoints. As illustrated in Figure 2, HybridPose utilizes a simple network $f_\phi^{\mathcal{E}}(I) \in \mathbb{R}^{2 \times |\mathcal{E}|}$ to predict edge vectors in the 2D image plane, where $|\mathcal{E}|$ denotes the number of edges in the pre-defined graph. In our experiments, $\mathcal{E}$ is a fully-connected graph, i.e., $|\mathcal{E}| = \frac{|\mathcal{K}| \cdot (|\mathcal{K}| - 1)}{2}$.

Symmetry correspondences. The third intermediate representation consists of predicted pixel-wise symmetry correspondences that reflect the underlying reflection symmetry. In our experiments, HybridPose extends the network architecture of FlowNet 2.0 [15] that combines a dense pixel-wise flow and the semantic mask predicted by PVNet. The resulting symmetry correspondences are given by predicted pixel-wise flow within the mask region. Compared to the first two representations, the number of symmetry correspondences is significantly larger, which provides rich constraints even for occluded objects. However, symmetry correspondences only constrain two degrees of freedom in the rotation component of the object pose (c.f. [24]). It is necessary to combine symmetry correspondences with other intermediate representations.

A 3D model may possess multiple reflection symmetry planes. For these models, we train HybridPose to predict symmetry correspondences with respect to the most salient reflection symmetry plane, i.e., one with the largest number of symmetry correspondences on the original 3D model.

Summary of network design. In our experiments, $f_\theta^{\mathcal{K}}(I)$, $f_\phi^{\mathcal{E}}(I)$, and $f_\gamma^{\mathcal{S}}$ are all based on ResNet [11], and the implementation details are discussed in Section 4.1. Trainable parameters are shared across all except the last convolutional layer. Therefore, the overhead of introducing the edge prediction network $f_\phi^{\mathcal{E}}(I)$ and the symmetry prediction network $f_\gamma^{\mathcal{S}}$ is insignificant.

3.3. Pose Regression

The second module of HybridPose takes predicted intermediate representations $\{\mathcal{K}, \mathcal{E}, \mathcal{S}\}$ as input and outputs a 6D object pose $(R_I \in SO(3), \boldsymbol{t}_I \in \mathbb{R}^3)$ for the input image I. Similar to state-of-the-art pose regression approaches [35], HybridPose combines an initialization sub-module and a refinement sub-module. Both sub-modules leverage all predicted elements. The refinement sub-module additionally leverages a robust function to model outliers in the predicted elements.

In the following, we denote 3D keypoint coordinates in the canonical coordinate system as $\overline{\boldsymbol{p}}_k, 1 \leq k \leq |\mathcal{K}|$. To make notations uncluttered, we denote output of the first module, i.e., predicted keypoints, edge vectors, and symmetry correspondences as $\boldsymbol{p}_k \in \mathbb{R}^2, 1 \leq k \leq |\mathcal{K}|$, $\boldsymbol{v}_e \in \mathbb{R}^2, 1 \leq e \leq |\mathcal{E}|$, and $(\boldsymbol{q}_{s,1} \in \mathbb{R}^2, \boldsymbol{q}_{s,2} \in \mathbb{R}^2), 1 \leq s \leq |\mathcal{S}|$, respectively. Our formulation also uses the homogeneous coordinates $\hat{\boldsymbol{p}}_k \in \mathbb{R}^3$, $\hat{\boldsymbol{v}}_e \in \mathbb{R}^3, \hat{\boldsymbol{q}}_{s,1} \in \mathbb{R}^3$ and $\hat{\boldsymbol{q}}_{s,2} \in \mathbb{R}^3$ of $\boldsymbol{p}_k$, $\boldsymbol{v}_e$, $\boldsymbol{q}_{s,1}$ and $\boldsymbol{q}_{s,2}$ respectively. The homogeneous coordinates are normalized by the camera intrinsic matrix.

Initialization sub-module. This sub-module leverages constraints between $(R_I, \boldsymbol{t}_I)$ and predicted elements and solves $(R_i, \boldsymbol{t}_I)$ in the affine space, which are then projected to $SE(3)$ in an alternating optimization manner. To this end, we introduce the following difference vectors for each type of predicted elements:

$$\overline{\boldsymbol{r}}_{R,\boldsymbol{t}}^{\mathcal{K}}(\boldsymbol{p}_k) := \hat{\boldsymbol{p}}_k \times (R\overline{\boldsymbol{p}}_k + \boldsymbol{t}), \tag{1}$$

$$\overline{\boldsymbol{r}}_{R,\boldsymbol{t}}^{\mathcal{E}}(\boldsymbol{v}_e, \boldsymbol{p}_{e_s}) := \hat{\boldsymbol{v}}_e \times (R\overline{\boldsymbol{p}}_{e_t} + \boldsymbol{t}) + \hat{\boldsymbol{p}}_{e_s} \times (R\overline{\boldsymbol{v}}_e) \tag{2}$$

$$r_{R,\boldsymbol{t}}^{\mathcal{S}}(\boldsymbol{q}_{s,1}, \boldsymbol{q}_{s,2}) := (\hat{\boldsymbol{q}}_{s,1} \times \hat{\boldsymbol{q}}_{s,2})^T R\overline{\boldsymbol{n}}_r. \tag{3}$$

where e_s and e_t are end vertices of edge e, $\overline{\boldsymbol{v}}_e = \overline{\boldsymbol{p}}_{e_t} - \overline{\boldsymbol{p}}_{e_s} \in \mathbb{R}^3$, and $\overline{\boldsymbol{n}}_r \in \mathbb{R}^3$ is the normal of the reflection symmetry plane in the canonical system.

HybridPose modifies the framework of EPnP [18] to generate the initial poses. By combining these three constraints from predicted elements, we generate a linear system of the form $A\boldsymbol{x} = \boldsymbol{0}$, where A is matrix and its dimension is $(3|\mathcal{K}| + 3|\mathcal{E}| + |\mathcal{S}|) \times 12$. $\boldsymbol{x} = [\boldsymbol{r}_1^T, \boldsymbol{r}_2^T, \boldsymbol{r}_3^T, \boldsymbol{t}^T]_{12 \times 1}^T$ is a vector that contains rotation and translation parameters in

affine space. To model the relative importance among keypoints, edge vectors, and symmetry correspondences, we rescale (2) and (3) by hyper-parameters α_E and α_S, respectively, to generate A.

Following EPnP [18], we compute $\boldsymbol{x}$ as

$$\boldsymbol{x} = \sum_{i=1}^{N} \gamma_i \boldsymbol{v}_i \tag{4}$$

where $\boldsymbol{v}_i$ is the i_{th} smallest right singular vector of A. Ideally, when predicted elements are noise-free, $N = 1$ with $\boldsymbol{x} = \boldsymbol{v}_1$ is an optimal solution. However, this strategy performs poorly given noisy predictions. Same as EPnP [18], we choose $N = 4$. To compute the optimal $\boldsymbol{x}$, we optimize latent variables γ_i and the rotation matrix R in an alternating optimization procedure with following objective function:

$$\min_{R \in \mathbb{R}^{3 \times 3}, \gamma_i} \| \sum_{i=1}^{4} \gamma_i R_i - R \|_{\mathcal{F}}^2 \tag{5}$$

where $R_i \in \mathbb{R}^{3 \times 3}$ is reshaped from the first 9 elements of $\boldsymbol{v}_i$. After obtaining optimal γ_i, we project the resulting affine transformation $\sum_{i=1}^{4} \gamma_i R_i$ into a rigid transformation. Due to space constraint, we defer details to the supp. material.

Refinement sub-module. Although (5) combines hybrid intermediate representations and admits good initialization, it does not directly model outliers in predicted elements. Another limitation comes from (1) and (2), which do not minimize the projection errors (i.e., with respect to keypoints and edges), which are known to be effective in keypoint-based pose estimation (c.f. [35]).

Benefited from having an initial object pose $(R^{init}, \boldsymbol{t}^{init})$, the refinement sub-module performs local optimization to refine the object pose. We introduce two difference vectors that involve projection errors: $\forall k, e, s$,

$$\boldsymbol{r}_{R,\boldsymbol{t}}^{\mathcal{K}}(\boldsymbol{p}_k) := \mathcal{P}_{R,\boldsymbol{t}}(\overline{\boldsymbol{p}}_k) - \boldsymbol{p}_k, \tag{6}$$

$$\boldsymbol{r}_{R,\boldsymbol{t}}^{\mathcal{E}}(\boldsymbol{v}_e) := \mathcal{P}_{R,\boldsymbol{t}}(\overline{\boldsymbol{p}}_{e_t}) - \mathcal{P}_{R,\boldsymbol{t}}(\overline{\boldsymbol{p}}_{e_s}) - \boldsymbol{v}_e, \tag{7}$$

where $\mathcal{P}_{R,\boldsymbol{t}} : \mathbb{R}^3 \to \mathbb{R}^2$ is the projection operator induced from the current pose $(R, \boldsymbol{t})$.

To prune outliers in the predicted elements, we consider a generalized German-Mcclure (or GM) robust function

$$\rho(x, \beta) := \beta_1^2 / (\beta_2^2 + x^2). \tag{8}$$

With this setup, HybridPose solves the following non-linear

optimization problem for pose refinement:

$$\min_{R,\boldsymbol{t}} \sum_{k=1}^{|\mathcal{K}|} \rho(\|\boldsymbol{r}_{R,\boldsymbol{t}}^{\mathcal{K}}(\boldsymbol{p}_k)\|, \beta_{\mathcal{K}}) \|\boldsymbol{r}_{R,\boldsymbol{t}}^{\mathcal{K}}(\boldsymbol{p}_k)\|_{\Sigma_k}^2$$

$$+ \frac{|\mathcal{K}|}{|\mathcal{E}|} \sum_{e=1}^{|\mathcal{E}|} \rho(\|\boldsymbol{r}_{R,\boldsymbol{t}}^{\mathcal{E}}(\boldsymbol{v}_e)\|, \beta_{\mathcal{E}}) \|\boldsymbol{r}_{R,\boldsymbol{t}}^{\mathcal{E}}(\boldsymbol{v}_e)\|_{\Sigma_e}^2$$

$$+ \frac{|\mathcal{K}|}{|\mathcal{S}|} \sum_{s=1}^{|\mathcal{S}|} \rho(r_{R,\boldsymbol{t}}^{\mathcal{S}}(\boldsymbol{q}_{s,1}, \boldsymbol{q}_{s,2}), \beta_{\mathcal{S}}) r_{R,\boldsymbol{t}}^{\mathcal{S}}(\boldsymbol{q}_{s,1}, \boldsymbol{q}_{s,2})^2$$

$$\tag{9}$$

where $\beta_{\mathcal{K}}$, $\beta_{\mathcal{E}}$, and $\beta_{\mathcal{S}}$ are separate hyper-parameters for keypoints, edges, and symmetry correspondences. Σ_k and Σ_e denote the covariance information attached to the keypoint and edge predictions. $\|\boldsymbol{x}\|_A = (\boldsymbol{x}^T A \boldsymbol{x})^{\frac{1}{2}}$. When covariances of predictions are unavailable, we simply set $\Sigma_k = \Sigma_e = I_2$. The above optimization problem is solved by Gauss-Newton method starting from R^{init} and $\boldsymbol{t}^{init}$.

In the supp. material, we provide a stability analysis of (9), and show how the optimal solution of (9) changes with respect to noise in predicted representations. We also show collaborative strength among all three intermediate representations. While keypoints significantly contribute to the accuracy of $\boldsymbol{t}$, edge vectors and symmetry correspondences can stablize the regression of R.

3.4. HybridPose Training

This section describes how to train the prediction networks and hyper-parameters of HybridPose using a labeled dataset $\mathcal{T} = \{I, (\mathcal{K}_I^{gt}, \mathcal{E}_I^{gt}, \mathcal{S}_I^{gt}, (R_I^{gt}, \boldsymbol{t}_I^{gt}))\}$. With I, $\mathcal{K}_I^{gt}$, $\mathcal{E}_I^{gt}$, $\mathcal{S}_I^{gt}$, and $(R_I^{gt}, \boldsymbol{t}_I^{gt})$, we denote the RGB image, labeled keypoints, edges, symmetry correspondences, and ground-truth object pose, respectively. A popular strategy is to train the entire model end-to-end, e.g., using recurrent networks to model the optimization procedure and introducing loss terms on the object pose output as well as the intermediate representations. However, we found this strategy suboptimal. The distribution of predicted elements on the training set differs from that on the testing set. Even by carefully tuning the trade-off between supervisions on predicted elements and the final object pose, the pose regression model, which fits the training data, generalizes poorly on the testing data.

Our approach randomly divides the labeled set $\mathcal{T} = \mathcal{T}_{train} \cup \mathcal{T}_{val}$ into a training set and a validation set. $\mathcal{T}_{train}$ is used to train the prediction networks, and $\mathcal{T}_{val}$ trains the hyper-parameters of the pose regression model. Implementation and training details of the prediction networks are presented in Section 4.1. In the following, we focus on training the hyper-parameters using $\mathcal{T}_{val}$.

Initialization sub-module. Let R_I^{init} and $\boldsymbol{t}_I^{init}$ be the output of the initialization sub-module. We obtain the optimal

hyper-parameters α_E and α_S by solving the following optimization problem:

$$\min_{\alpha_E, \alpha_S} \sum_{I \in \mathcal{T}_{\text{pose}}} \left(\|R_I^{init} - R_I^{gt}\|_{\mathcal{F}}^2 + \|\boldsymbol{t}_I^{init} - \boldsymbol{t}_I^{gt}\|^2 \right). \quad (10)$$

Since the number of hyper-parameters is rather small, and the pose initialization step does not admit an explicit expression, we use the finite-difference method to compute numerical gradient, i.e., by fitting the gradient to samples of the hyper-parameters around the current solution. We then apply back-track line search for optimization.

Refinement sub-module. Let $\beta = \{\beta_{\mathcal{K}}, \beta_{\mathcal{E}}, \beta_{\mathcal{S}}\}$ be the hyper-parameters of this sub-module. For each instance $(I, (\mathcal{K}_I^{gt}, \mathcal{E}_I^{gt}, \mathcal{S}_I^{gt}, (R_I^{gt}, \boldsymbol{t}_I^{gt}))) \in \mathcal{T}_{val}$, denote the objective function in (9) as $f_I(\boldsymbol{c}, \beta)$, where $\boldsymbol{c} = (\boldsymbol{c}^T, \bar{\boldsymbol{c}}^T)^T \in \mathbb{R}^6$ is a local parameterization of R_I and $\boldsymbol{t}_I$, i.e., $R_I = \exp(\boldsymbol{c}\times)R_I^{gt}, \boldsymbol{t}_I = \boldsymbol{t}_I^{gt} + \bar{\boldsymbol{c}}$. $\boldsymbol{c}$ encodes the different the current estimated pose and the ground-truth pose in $SE(3)$.

The refinement module solves an unconstrained optimization problem, whose optimal solution is dictated by its critical points and the loss surface around the critical points. We consider two simple objectives. The first objective forces $\frac{\partial f_I}{\partial \boldsymbol{c}}(\boldsymbol{0}, \beta) \approx 0$, or in other words, the ground-truth is approximately a critical point. The second objective minimizes the condition number $\kappa\left(\frac{\partial^2 f_I}{\partial^2 \boldsymbol{c}}(\boldsymbol{0}, \beta)\right) = \lambda_{\max}\left(\frac{\partial^2 f_I}{\partial^2 \boldsymbol{c}}(\boldsymbol{0}, \beta)\right) / \lambda_{\min}\left(\frac{\partial^2 f_I}{\partial^2 \boldsymbol{c}}(\boldsymbol{0}, \beta)\right)$. This objective regularizes the loss surface around each optimal solution, promoting a large converge radius for $f_I(\boldsymbol{c}, \beta)$. With this setup, we formulate the following objective function to optimize β:

$$\min_{\beta} \sum_{I \in \mathcal{T}_{val}} \|\frac{\partial f_I}{\partial \boldsymbol{c}}(\boldsymbol{0}, \beta)\|^2 + \gamma \kappa\left(\frac{\partial^2 f_I}{\partial^2 \boldsymbol{c}}(\boldsymbol{0}, \beta)\right) \quad (11)$$

where γ is set to 10^{-4}. The same strategy used in (10) is then applied to optimize (11).

4. Experimental Evaluation

This section presents an experimental evaluation of the proposed approach. Section 4.1 describes the experimental setup. Section 4.2 quantitatively and qualitatively compares HybridPose with other 6D pose estimation methods. Section 4.3 presents an ablation study to investigate the effectiveness of symmetry correspondences, edge vectors, and the refinement sub-module.

4.1. Experimental Setup

Datasets. We consider two popular benchmark datasets that are widely used in the 6D pose estimation problem, Linemod [12] and Occlusion Linemod [3]. In comparsion to Linemod, Occlusion Linemod contains more examples where the objects are under occlusion. Our keypoint annotation strategy follows that of [34], i.e., we choose $|\mathcal{K}| = 8$

keypoints via the farthest point sampling algorithm. Edge vectors are defined as vectors connecting each pair of keypoints. In total, each object has $|\mathcal{E}| = \frac{|\mathcal{K}| \cdot (|\mathcal{K}| - 1)}{2} = 28$ edges. We further use the algorithm proposed in [8] to annotate Linemod and Occlusion Linemod with reflection symmetry labels. On each dataset, we randomly select 80% of the examples for training, 20 instances for validation, and the rest for testing.

Implementation details. We use ResNet [11] with pretrained weights on ImageNet [7] to build the prediction networks $f_\theta^{\mathcal{K}}$, $f_\phi^{\mathcal{E}}$, and $f_\gamma^{\mathcal{S}}$. The prediction networks take an RGB image I of size $(3, H, W)$ as input, and output a tensor of size (C, H, W), where (H, W) is the image resolution, and $C = 1 + 2|\mathcal{K}| + 2|\mathcal{E}| + 2$ is the number of channels in the output tensor.

The first channel in the output tensor is a binary segmentation mask M. If $M(x, y) = 1$, then (x, y) corresponds to a pixel on the object of interest in the input image I. The segmentation mask is trained using the cross-entropy loss.

The $2|\mathcal{K}|$ channels afterwards in the output tensor give x and y components of all $|\mathcal{K}|$ keypoints. A voting-based keypoint localization scheme [34] is applied to extract the coordinates of 2D keypoints from this $2|\mathcal{K}|$-channel tensor and the segmentation mask M.

The next $2|\mathcal{E}|$ channels in the output tensor give the x and y components of all $|\mathcal{E}|$ edges, which we denote as $Edge$. Let i $(0 \leq i < |\mathcal{E}|)$ be the index of an edge. Then

$$Edge_i = \{(Edge(2i, x, y), Edge(2i + 1, x, y)) | M(x, y) = 1\}$$

is a set of 2-tuples containing pixel-wise predictions of the ith edge vector in $Edge$. The mean of $Edge_i$ is extracted as the predicted edge.

The final 2 channels in the output tensor define the x and y components of symmetry correspondences. We denote this 2-channel "map" of symmetry correspondences as Sym. Let (x, y) be a pixel on the object of interest in the input image, i.e. $M(x, y) = 1$. Assuming $\Delta x = Sym(0, x, y)$ and $\Delta y = Sym(1, x, y)$, we consider (x, y) and $(x + \Delta x, y + \Delta y)$ to be symmetric with respect to the reflection symmetry plane.

We train all three intermediate representations using the smooth ℓ_1 loss described in [9]. Network training employs the Adam [17] optimizer for 500 epochs. The learning rate is 0.02 before the 200th epoch, and 0.002 after the 200th epoch. Training weights of the segmentation mask, keypoints, edge vectors, and symmetry correspondences are 1.0, 10.0, 0.1, and 0.1, respectively.

The architecture described above achieves good performance in terms of detection accuracy. Nevertheless, it should be emphasized that the framework of HybridPose can incorporate future improvements in keypoint, edge vector, and symmetry correspondence detection techniques. Besides, Hybridpose can be extended to handling multiple objects within an image. One approach is to pre-
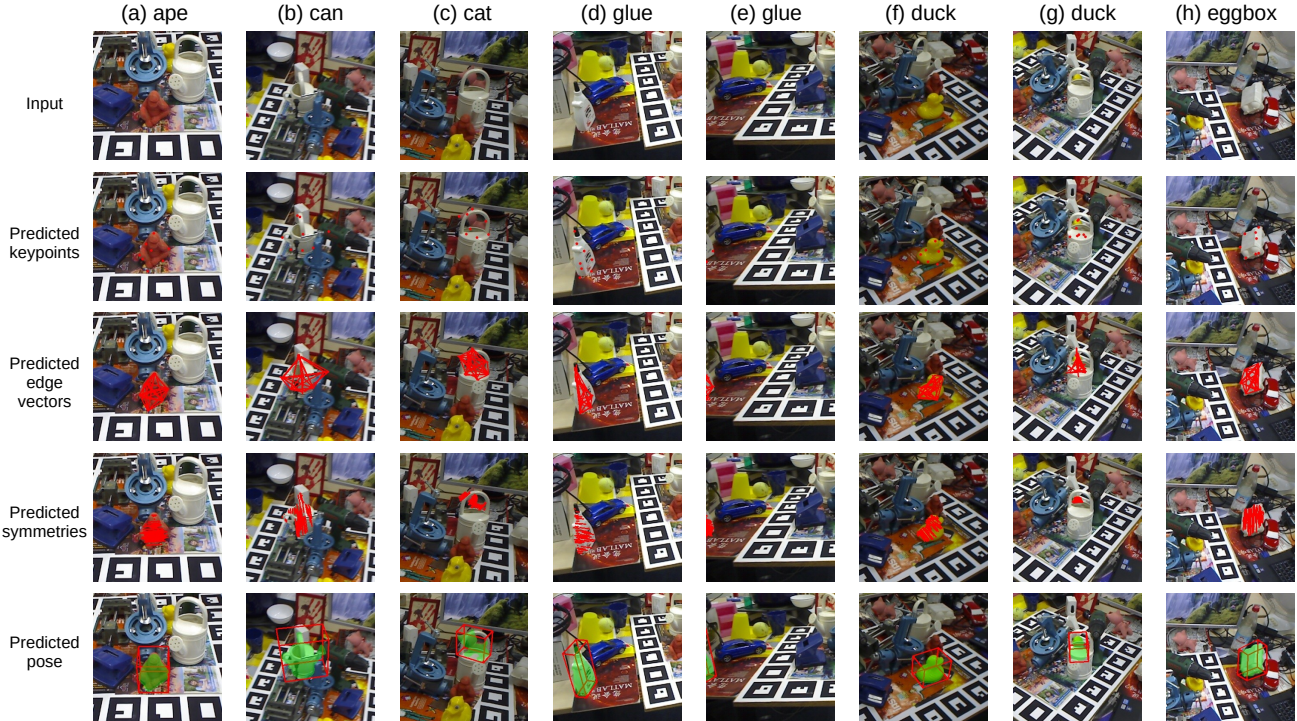
Figure 3. **Pose regression results.** HybridPose is able to accurately predict 6D poses from RGB images. HybridPose handles situations where the object has no occlusion (a, d, f, h), light occlusion (b, c), and severe occlusion (e, g). For illustrative purposes, we only draw 50 randomly selected symmetry correspondences in each example.

dict instance-level rather than semantic-level segmentation masks by methods such as Mask R-CNN [10]. Intermediate representations are then extracted from each instance, and fed to the pose regression module in 3.3.

Evaluation protocols. We use two metrics to evaluate the performance of HybridPose:

1. ADD(-S) [12, 42] first calculates the distance between two point sets transformed by predicted pose and ground-truth pose respectively, and then extracts the mean distance. When the object possesses symmetric pose ambiguity, the mean distance is computed from the closest points between two transformed sets. ADD(-S) accuracy is defined as the percentage of examples whose calculated mean distance is less than 10% of the model diameter.

2. In the ablation study, we compute and report the the angular rotation error $\|\frac{\log(R_{gt}^T R_I)}{2}\|$ and the relative translation error $\frac{\|t_I - t_{gt}\|}{d}$ between the predicted pose (R_I, t_I) and the ground-truth pose (R_{gt}, t_{gt}), where d is object diameter.

4.2. Analysis of Results

As shown in Table 1, Table 2, and Figure 3, HybridPose leads to accurate pose estimation. On Linemod and Occlusion Linemod, HybridPose has an average ADD(-S) accuracy of 94.5 and 79.2, respectively. The result on Linemod outperforms all except one state-of-the-art approaches that

object	Tekin	BB8	Pix2Pose	PVNet	CDPN	DPOD	Ours
ape	21.6	40.4	58.1	43.6	64.4	**87.7**	80.7
benchvise	81.8	91.8	91.0	99.9	97.8	98.5	**100.0**
cam	36.6	55.7	60.9	86.9	91.7	**96.1**	95.5
can	68.8	64.1	84.4	95.5	95.9	**99.7**	97.3
cat	41.8	62.6	65.0	79.3	83.8	**94.7**	90.7
driller	63.5	74.4	76.3	96.4	96.2	**98.8**	98.2
duck	27.2	44.3	43.8	52.6	66.8	**86.3**	83.5
eggbox†	69.6	57.8	96.8	99.2	99.7	99.9	**100.0**
glue†	80.0	41.2	79.4	95.7	**99.6**	96.8	98.7
holepuncher	42.6	67.2	74.8	81.9	85.8	86.9	**91.2**
iron	75.0	84.7	83.4	98.9	97.9	**100.0**	97.6
lamp	71.1	76.5	82.0	99.3	**97.9**	96.8	97.3
phone	47.7	54.0	45.0	92.4	90.8	94.7	**98.3**
average	56.0	62.7	72.4	86.3	89.9	**95.2**	94.5

Table 1. **Quantitative evaluation: ADD(-S) accuracy on Linemod.** Baseline approaches: Tekin et al. [38], BB8 [36], Pix2Pose [30], PVNet [34], CDPN [20], and DPOD [44]. Objects annotated with (†) possess symmetric pose ambiguity.

regress poses from intermediate representations. The result on Occlusion-Linemod outperforms all state-of-the-art approaches.

Baseline comparison on Linemod. HybridPose outperforms PVNet [34], the backbone model we use to predict keypoints. The improvement is consistent across all except one object classes, which demonstrates clear advantage of using a hybrid as opposed to unitary intermediate representation. HybridPose shows competitive results against

object	PoseCNN	Oberweger	Hu	Pix2Pose	PVNet	DPOD	Ours
ape	9.6	12.1	17.6	22.0	15.8	-	**53.3**
can	45.2	39.9	53.9	44.7	63.3	-	**86.5**
cat	0.93	8.2	3.3	22.7	16.7	-	**73.4**
driller	41.4	45.2	62.4	44.7	65.7	-	**92.8**
duck	19.6	17.2	19.2	15.0	25.2	-	**62.8**
eggbox[†]	22.0	22.1	25.9	25.2	50.2	-	**95.3**
glue[†]	38.5	35.8	39.6	32.4	49.6	-	**92.5**
holepuncher	22.1	36.0	21.3	49.5	39.7	-	**76.7**
average	24.9	27.0	27.0	32.0	40.8	47.3	**79.2**

Table 2. **Quantitative evaluation: ADD(-S) accuracy on Occlusion Linemod.** Baseline approaches: PoseCNN [42], Oberweger et al. [29], Hu et al. [14], PVNet [34], and DPOD [44]. Objects annotated with (†) possess symmetric pose ambiguity.

	keypoints		keypoints + symmetries		full model	
	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.
ape	1.914°	0.107	1.809°	0.113	1.543°	0.092
can	1.472°	0.059	1.710°	0.073	0.912°	0.041
cat	1.039°	0.119	0.888°	0.117	0.751°	0.055
driller	1.180°	0.057	1.180°	0.057	0.803°	0.027
duck	1.773°	0.116	1.679°	0.115	1.439°	0.068
eggbox	1.675°	0.107	1.587°	0.105	1.052°	0.052
glue	1.796°	0.097	1.681°	0.099	1.224°	0.066
holepuncher	2.319°	0.141	2.192°	0.140	1.704°	0.051
mean	1.648°	0.100	1.590°	0.102	1.179°	0.057

Table 3. **Quantitative evaluation with different intermediate representations.** We report errors using two metrics: the median of absolute angular error in rotation, and the median of relative error in translation with respect to object diameter.

DPOD [44], winning on six object classes. The advantage of DPOD comes from data augmentation and explicit modeling of dense correspondences between input and projected images, both of which cater to situations without object occlusion. A detailed analysis reveals that the classes of objects on which HybridPose exhibits sub-optimal performance are among the smallest objects in Linemod. It suggests that pixel-based descriptors used in our pipeline are limited by image resolution.

Baseline comparison on Occlusion Linemod. HybridPose outperforms all baselines by a considerable margin. In terms of ADD(-S), our approach improves PVNet [34] from 40.8 to 79.2, representing a 94.1% enhancement, which clearly shows the advantage of HybridPose on occluded objects, where predictions of invisible keypoints can be noisy, and visible keypoints may not provide sufficient constraints for pose regression alone. HybridPose also outperforms DPOD, the state-of-the-art model on this dataset, by 67.4%. One explanation is that rendering-based approaches like DPOD work less well on occluded objects, due to the difficulty in modeling occlusions in data augmentation and correspondence computation.

Running time. On a desktop with 16-core Intel(R) Xeon(R) E5-2637 CPU and GeForce GTX 1080 GPU, HybridPose takes 0.6 second to predict the intermediate representations, 0.4 second to regress the pose. Assuming a batch size of 30, this gives an an average processing speed of around 30 fps, enabling real-time analysis.

4.3. Ablation Study

Table 3 summarizes the performance of HybridPose using different predicted intermediate representations on the Occlusion Linemod dataset. Ablation study on Linemod is deferred to the supp. material.

With keypoints. As a baseline approach, we estimate object poses by only utilizing keypoint information. As shown in Table 3, this gives a mean absolute rotation error of 1.648°, and a mean relative translation error of 0.100.

With keypoints and symmetry. Adding symmetry correspondences to keypoints leads to noticeable performance gains in rotation. The relative performance gain is 3.52%,

and such improvement is almost consistent across all object categories, which demonstrates the effectiveness of symmetry correspondences. On the other hand, the translation error of using keypoints and using keypoints + symmetry remains almost the same. One explanation is that symmetry correspondences only constrain two degrees of freedom in a total of three rotation parameters, and provide no constraint on translation parameters (see (3)).

Full model. Adding edge vectors to keypoints and symmetry correspondences leads to salient performance gain in both rotation and translation estimations. The relative performance gains in rotation and translation are 25.85% and 44.12%, respectively. One explanation is that edge vectors provide more constraints on both translation and rotation (see (2)). Edge vectors provide more constraints on translation than keypoints as they represent adjacent keypoints displacement and provide gradient information for regression. Unlike symmetry correspondences, edge vectors constrain 3 degrees of freedom on rotation parameters which further boosts the performance of rotation estimation.

5. Conclusions and Future Work

In this paper, we introduce HybridPose, a 6D pose estimation approach that utilizes keypoints, edge vectors, and symmetry correspondences. Experiments show that HybridPose enjoys real-time prediction and outperforms current state-of-the-art pose estimation approaches in accuracy. HybridPose is robust to occlusion and extreme poses. In the future, we plan to extend HybridPose to include more intermediate representations such as shape primitives, normals, and planar faces. Another possible direction is to enforce consistency across different representations in a similar way to [46] as a self-supervision loss in network training.

6. Acknowledgement

References

[1] Ibragim R. Atadjanov and Seungkyu Lee. Reflection symmetry detection via appearance of structure descriptor. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III*, pages 3–18, 2016. 2

[2] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4380–4389, 2015. 2

[3] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *European conference on computer vision*, pages 536–551. Springer, 2014. 1, 2, 6

[4] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, et al. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3364–3372, 2016. 2

[5] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1302–1310, 2017. 2

[6] Enric Corona, Kaustav Kundu, and Sanja Fidler. Pose estimation for objects with rotational symmetry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7215–7222. IEEE, 2018. 2

[7] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009. 6

[8] Aleksandrs Ecins, Cornelia Fermüller, and Yiannis Aloimonos. Seeing behind the scene: Using symmetry to reason about objects in cluttered environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7193–7200. IEEE, 2018. 6

[9] Ross Girshick. Fast r-cnn. *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015. 6

[10] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 7

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 4, 6

[12] 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 *Proceedings of the 11th Asian Conference on Computer Vision - Volume Part I*, ACCV'12, pages 548–562, Berlin, Heidelberg, 2013. Springer-Verlag. 2, 6, 7

[13] Wei Hong, Allen Yang Yang, Kun Huang, and Yi Ma. On symmetry and multiple-view geometry: Structure, pose, and calibration from a single image. *International Journal of Computer Vision*, 60(3):241–265, 2004. 2

[14] Yinlin Hu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. Segmentation-driven 6d object pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3385–3394, 2019. 2, 8

[15] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1647–1655, 2017. 4

[16] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015. 1

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. 6

[18] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009. 2, 4, 5

[19] 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)*, pages 683–698, 2018. 3

[20] 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 International Conference on Computer Vision*, pages 7678–7687, 2019. 2, 7

[21] Guilin Liu, Chao Yang, Zimo Li, Duygu Ceylan, and Qixing Huang. Symmetry-aware depth estimation using deep neural networks. *arXiv preprint arXiv:1604.06079*, 2016. 2

[22] Yanxi Liu, Hagit Hel-Or, Craig S. Kaplan, and Luc Van Gool. Computational symmetry in computer vision and computer graphics. *Foundations and Trends in Computer Graphics and Vision*, 5(1-2):1–195, 2010. 2

[23] Yu Liu and Michael S Lew. Learning relaxed deep supervision for better edge detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 231–240, 2016. 2

[24] Yi Ma, Stefano Soatto, Jana Kosecka, and S. Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003. 4

[25] Fabian Manhardt, Diego Martin Arroyo, Christian Rupprecht, Benjamin Busam, Tolga Birdal, Nassir Navab, and Federico Tombari. Explaining the ambiguity of object detection and 6d pose from visual data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6841–6850, 2019. 2

[26] 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)*, pages 800–815, 2018. 3

[27] Niloy J. Mitra, Mark Pauly, Michael Wand, and Duygu Ceylan. Symmetry in 3d geometry: Extraction and applications. *Comput. Graph. Forum*, 32(6):1–23, Sept. 2013. 2

[28] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *Computer Vi-*

sion - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII, pages 483–499, 2016. 2

[29] Markus Oberweger, Mahdi Rad, and Vincent Lepetit. Making deep heatmaps robust to partial occlusions for 3d object pose estimation. *Lecture Notes in Computer Science*, page 125141, 2018. 8

[30] Kiru Park, Timothy Patten, and Markus Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7668–7677, 2019. 2, 7

[31] Georgios Passalis, Panagiotis Perakis, Theoharis Theoharis, and Ioannis A Kakadiaris. Using facial symmetry to handle pose variations in real-world 3d face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10):1938–1951, 2011. 2

[32] 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. 2, 3

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[34] Sida Peng, Yuan Liu, Qixing Huang, Hujun Bao, and Xiaowei Zhou. Pvnet: Pixel-wise voting network for 6dof pose estimation. *CoRR*, abs/1812.11788, 2018. 1, 2, 3, 4, 6, 7, 8

[35] Mikael Persson and Klas Nordberg. Lambda twist: An accurate fast robust perspective three point (p3p) solver. In *The European Conference on Computer Vision (ECCV)*, September 2018. 4, 5

[36] 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*, pages 3828–3836, 2017. 2, 7

[37] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 699–715, 2018. 3

[38] Bugra Tekin, Sudipta N. Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 292–301, 2018. 1, 7

[39] Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1510–1519, 2015. 1

[40] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3343–3352, 2019. 2, 3

[41] Zhaozhong Wang, Zesheng Tang, and Xiao Zhang. Reflection symmetry detection using locally affine invariant edge correspondence. *IEEE Trans. Image Processing*, 24(4):1297–1301, 2015. 2

[42] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*, 2018. 1, 2, 7, 8

[43] Tianfan Xue, Jianzhuang Liu, and Xiaoou Tang. Symmetric piecewise planar object reconstruction from a single image. In *CVPR 2011*, pages 2577–2584. IEEE, 2011. 2

[44] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Dpod: 6d pose object detector and refiner, 2019. 2, 3, 7, 8

[45] Ziheng Zhang, Zhengxin Li, Ning Bi, Jia Zheng, Jinlei Wang, Kun Huang, Weixin Luo, Yanyu Xu, and Shenghua Gao. Ppgnet: Learning point-pair graph for line segment detection. *arXiv preprint arXiv:1905.03415*, 2019. 2

[46] Zaiwei Zhang, Zhenxiao Liang, Lemeng Wu, Xiaowei Zhou, and Qixing Huang. Path-invariant map networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11084–11094, 2019. 8

[47] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1851–1858, 2017. 2

[48] Xingyi Zhou, Arjun Karpur, Linjie Luo, and Qixing Huang. Starmap for category-agnostic keypoint and viewpoint estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 318–334, 2018. 2

Supplemental Material:
HybridPose: 6D Object Pose Estimation under Hybrid Representations

Chen Song*, Jiaru Song*, Qixing Huang

The University of Texas at Austin

song@cs.utexas.edu, jiarus@cs.utexas.edu, huangqx@cs.utexas.edu

This is the supplemental material to "HybridPose: 6D Object Pose Estimation under Hybrid Representations". We provide detailed explanations to our the algorithm used in the initialization sub-module. We also conduct a stability analysis of the refinement sub-module, and show how the optimal solution to the the objective function changes with respect to noise in predicted representations. In addition, we present an ablation study on Linemod [1] dataset. While keypoints alone already achieve reasonable pose estimation performance on Linemod, utilizing symmetry correspondences and edge vectors lead to slight improvements.

1. Initial Solution for Pose Regression

Recall that we denote 3D keypoint coordinates in the canonical coordinate system as $\overline{\boldsymbol{p}}_k, 1 \leq k \leq |\mathcal{K}|$. To make notations uncluttered, we denote output of the first module, i.e., predicted keypoints, edge vectors, and symmetry correspondences as $\boldsymbol{p}_k \in \mathbb{R}^2, 1 \leq k \leq |\mathcal{K}|$, $\boldsymbol{v}_e \in \mathbb{R}^2, 1 \leq e \leq |\mathcal{E}|$, and $(\boldsymbol{q}_{s,1} \in \mathbb{R}^2, \boldsymbol{q}_{s,2} \in \mathbb{R}^2), 1 \leq s \leq |\mathcal{S}|$, respectively. Our formulation also uses the homogeneous coordinates $\hat{\boldsymbol{p}}_k \in \mathbb{R}^3, \hat{\boldsymbol{v}}_e \in \mathbb{R}^3, \hat{\boldsymbol{q}}_{s,1} \in \mathbb{R}^3$ and $\hat{\boldsymbol{q}}_{s,2} \in \mathbb{R}^3$ of $\boldsymbol{p}_k$, $\boldsymbol{v}_e$, $\boldsymbol{q}_{s,1}$ and $\boldsymbol{q}_{s,2}$ respectively. The homogeneous coordinates are normalized by camera intrinsic matrix.

1.1. Three constraints for object pose.

We seek to generalize the EPnP algorithm which only exploits keypoint 2D-3D correspondences for pose estimation by leveraging hybrid representations, keypoint, edge vector and symmetry correspondence. To this end, we introduce the following difference vectors for each type of predicted elements:

$$\overline{\boldsymbol{r}}^{\mathcal{K}}_{R,\boldsymbol{t}}(\boldsymbol{p}_k) := \hat{\boldsymbol{p}}_k \times (R\overline{\boldsymbol{p}}_k + \boldsymbol{t}), \tag{1}$$

$$\overline{\boldsymbol{r}}^{\mathcal{E}}_{R,\boldsymbol{t}}(\boldsymbol{v}_e, \boldsymbol{p}_{e_s}) := \hat{\boldsymbol{v}}_e \times (R\overline{\boldsymbol{p}}_{e_t} + \boldsymbol{t}) + \hat{\boldsymbol{p}}_{e_s} \times (R\overline{\boldsymbol{v}}_e), \tag{2}$$

$$r^{\mathcal{S}}_{R,\boldsymbol{t}}(\boldsymbol{q}_{s,1}, \boldsymbol{q}_{s,2}) := (\hat{\boldsymbol{q}}_{s,1} \times \hat{\boldsymbol{q}}_{s,2})^T R\overline{\boldsymbol{n}}_r. \tag{3}$$

where e_s and e_t are end vertices of edge e, $\overline{\boldsymbol{v}}_e = \overline{\boldsymbol{p}}_{e_t} - \overline{\boldsymbol{p}}_{e_s} \in \mathbb{R}^3$, and $\overline{\boldsymbol{n}}_r \in \mathbb{R}^3$ is the normal of the reflection symmetry

---

---

plane in the canonical system.

Proposition 1 *If there is a perfect alignment between the predicted elements and the corresponding 3D keypoint template with respect to the ground-truth pose $R^\star, \boldsymbol{t}^\star$. Then*

$$\overline{\boldsymbol{r}}^{\mathcal{K}}_{R^\star,\boldsymbol{t}^\star}(\boldsymbol{p}_k) = \boldsymbol{0}, \overline{\boldsymbol{r}}^{\mathcal{E}}_{R^\star,\boldsymbol{t}^\star}(\boldsymbol{v}_e \boldsymbol{p}_{e_s}) = \boldsymbol{0}, r^{\mathcal{S}}_{R,\boldsymbol{t}}(\boldsymbol{q}_{s,1}, \boldsymbol{q}_{s,2}) = 0$$

Proof:

1. The proof of the first equality is straight-forward as there exists a "depth" $\lambda_k > 0$ so that

$$\lambda_k \hat{\boldsymbol{p}}_k = R^\star \overline{\boldsymbol{p}}_k + \boldsymbol{t}^\star$$

It follows that

$$\boldsymbol{0} = \lambda_k \hat{\boldsymbol{p}}_k \times \hat{\boldsymbol{p}}_k = \hat{\boldsymbol{p}}_k \times (R^\star \overline{\boldsymbol{p}}_k + \boldsymbol{t}^\star)$$

2. The proof of the second equality follows the first equality. So we have

$$\hat{\boldsymbol{p}}_{e_s} \times (R^\star \overline{\boldsymbol{p}}_{e_s} + \boldsymbol{t}^\star) = \boldsymbol{0} \quad \hat{\boldsymbol{p}}_{e_t} \times (R^\star \overline{\boldsymbol{p}}_{e_t} + \boldsymbol{t}^\star) = \boldsymbol{0}$$

Replacing $\hat{\boldsymbol{p}}_{e_t}$ by $\hat{\boldsymbol{v}}_e + \hat{\boldsymbol{p}}_{e_s}$, we have

$$\hat{\boldsymbol{v}}_e \times (R^\star \overline{\boldsymbol{p}}_{e_t} + \boldsymbol{t}^\star) + \hat{\boldsymbol{p}}_{e_s} \times (R^\star \overline{\boldsymbol{p}}_{e_t} + \boldsymbol{t}^\star) = \boldsymbol{0}$$

Replacing the second $\overline{\boldsymbol{p}}_{e_t}$ by $\overline{\boldsymbol{v}}_e + \overline{\boldsymbol{p}}_{e_s}$ in the above equation, we have

$$\hat{\boldsymbol{v}}_e \times (R^\star \overline{\boldsymbol{p}}_{e_t} + \boldsymbol{t}^\star) + \hat{\boldsymbol{p}}_{e_s} \times R^\star \overline{\boldsymbol{v}}_e = \boldsymbol{0}$$

3. To prove the third equality, define the depths of $\hat{\boldsymbol{q}}_{s,1}$ and $\hat{\boldsymbol{q}}_{s,2}$ as $\lambda_{s,1}$ and $\lambda_{s,2}$ and the corresponding 3D model points in the canonical system as $\overline{\boldsymbol{q}}_{s,1}$ and $\overline{\boldsymbol{q}}_{s,2}$. $\overline{\boldsymbol{p}}_s$ is a point on the reflectional symmetry plane, whose normal is $\overline{\boldsymbol{n}}_r$. Given a symmetry correspondence pair $(\overline{\boldsymbol{q}}_{s,1}, \overline{\boldsymbol{q}}_{s,2})$, we have

$$\overline{\boldsymbol{q}}_{s,2} = (I_3 - 2\overline{\boldsymbol{n}}_r \overline{\boldsymbol{n}}_r^{\mathrm{T}})\overline{\boldsymbol{q}}_{s,1} + 2\overline{\boldsymbol{n}}_r \overline{\boldsymbol{n}}_r^{\mathrm{T}} \overline{\boldsymbol{p}}_s \tag{4}$$

Let $R_s = I_3 - 2\overline{\boldsymbol{n}}_r\overline{\boldsymbol{n}}_r^{\mathrm{T}}, \boldsymbol{t}_s = 2\overline{\boldsymbol{n}}_r\overline{\boldsymbol{n}}_r^{\mathrm{T}}\overline{\boldsymbol{p}}_s$. Following the camera perspective model, we have

$$\lambda_{s,1}\hat{\boldsymbol{q}}_{s,1} = R^\star\overline{\boldsymbol{q}}_{s,1} + t^\star$$
$$\lambda_{s,2}\hat{\boldsymbol{q}}_{s,2} = R^\star R_s\overline{\boldsymbol{q}}_{s,1} + R^\star\boldsymbol{t}_s + t^\star$$

Subtracting these two equations, we have

$$\lambda_{s,2}\hat{\boldsymbol{q}}_{s,2} - \lambda_{s,1}\hat{\boldsymbol{q}}_{s,1} = R^\star(R_s\overline{\boldsymbol{q}}_{s,1} + \boldsymbol{t}_s - \overline{\boldsymbol{q}}_{s,1})$$

Left multiply both sides of the equation by $\hat{\boldsymbol{q}}_{s,2}\times$ yields

$$-\lambda_{s,1}\hat{\boldsymbol{q}}_{s,2} \times \hat{\boldsymbol{q}}_{s,1} = \hat{\boldsymbol{q}}_{s,2} \times [R^\star(R_s\overline{\boldsymbol{q}}_{s,1} + \boldsymbol{t}_s - \overline{\boldsymbol{q}}_{s,1})] \tag{5}$$

Geometrically, (5) reveals that $\hat{\boldsymbol{q}}_{s,2} \times \hat{\boldsymbol{q}}_{s,1}$ is perpendicular to the plane with span of $\{\boldsymbol{q}_{s,2}, R^\star(R_s\overline{\boldsymbol{q}}_{s,1} + \boldsymbol{t}_s - \overline{\boldsymbol{q}}_{s,1})\}$, thus we have

$$(\hat{\boldsymbol{q}}_{s,2} \times \hat{\boldsymbol{q}}_{s,1})^{\mathrm{T}} R^\star(R_s\overline{\boldsymbol{q}}_{s,1} + \boldsymbol{t}_s - \overline{\boldsymbol{q}}_{s,1}) =$$
$$2(\overline{\boldsymbol{n}}_r^{\mathrm{T}}(\overline{\boldsymbol{p}}_s - \overline{\boldsymbol{q}}_{s,1}))(\hat{\boldsymbol{q}}_{s,2} \times \hat{\boldsymbol{q}}_{s,1})^{\mathrm{T}} R^\star\overline{\boldsymbol{n}}_r = 0$$

Since $2(\overline{\boldsymbol{n}}_r^{\mathrm{T}}(\overline{\boldsymbol{p}}_s - \overline{\boldsymbol{q}}_{s,1}))$ is a non-zero scalar, we can delete this term and finally get

$$(\hat{\boldsymbol{q}}_{s,2} \times \hat{\boldsymbol{q}}_{s,1})^{\mathrm{T}} R^\star\overline{\boldsymbol{n}}_r = 0$$

1.2. Pose solution in eigenvector space.

A nice feature shared by (1), (2) and (3) is that all constraints are linear in the elements of R and $\boldsymbol{t}$. This allows us to derive a closed-form solution of R and $\boldsymbol{t}$ in the affine transformation space. Specifically, we can define $\boldsymbol{x} = (\boldsymbol{r}_1^{\mathrm{T}}, \boldsymbol{r}_2^{\mathrm{T}}, \boldsymbol{r}_3^{\mathrm{T}}, \boldsymbol{t}^{\mathrm{T}})_{12\times 1}^{\mathrm{T}}$ as a vector that contains rotation and translation parameters in affine space. Expanding constraint (1) and constraint (2) yields three linear equations for each predicted element respectively for $\boldsymbol{x}$, and expanding constraint (3) yields one linear equation. By concatenating all linear equations of predicted elements together, we can generate a linear system of the form $A\boldsymbol{x} = \boldsymbol{0}$, where A is matrix and its dimension is $(3|\mathcal{K}| + 3|\mathcal{E}| + |\mathcal{S}|) \times 12$.

To model the relative importance among keypoints, edge vectors, and symmetry correspondences, we rescale (2) and (3) by hyper-parameters α_E and α_S, respectively, to generate A. As discussed in the body of this paper, we calculate α_E and α_S by solving an optimization problem using finite-difference and back-track line search.

Then following EPnP [2], we compute $\boldsymbol{x}$ as

$$\boldsymbol{x} = \sum_{i=1}^{N} \gamma_i \boldsymbol{v}_i \tag{6}$$

where $\boldsymbol{v}_i$ is the ith smallest right singular vector of A. Ideally, when predicted elements are noise-free, $N = 1$ with $\boldsymbol{x} = \boldsymbol{v}_1$ is an optimal solution. However, this strategy performs poorly given noisy predictions. Same as EPnP [2], we choose $N = 4$.

1.3. Optimize a good linear combination.

To compute the optimal $\boldsymbol{x}$, we optimize latent variables γ_i and the rotation matrix R with following objective function:

$$\min_{R\in\mathbb{R}^{3\times 3}, \gamma_i} \|\sum_{i=1}^{4} \gamma_i R_i - R\|_{\mathcal{F}}^2 \tag{7}$$

where $R_i \in \mathbb{R}^{3\times 3}$ is reshaped from the first 9 elements of $\boldsymbol{v}_i$. We solve this optimization problem with the following alternating procedure:

1. Fix γ_i and solve for R by SVD. i.e. $R = U\mathrm{diag}(1,1,1)V^{\mathrm{T}}$ given $\sum_{i=1}^{4}\gamma_i R_i = U\Sigma V^{\mathrm{T}}$[1],

2. Fix R and solve for γ_i's by optimizing a linear system $\sum_{i=1}^{4}\gamma_i R_i = R$ in an element-wise manner.

To initialize γ_i's for the above optimization problem, we calculate γ_i with $i = 1...3$ by enforcing that $\sum_{i=1}^{3}\gamma_i R_i$ is an orthogonal matrix[2]:

$$(\sum_{i=1}^{3}\gamma_i R_i)^{\mathrm{T}}\sum_{i=1}^{3}\gamma_i R_i = I_3 \tag{8}$$

Since I_3 is a symmetric matrix, expanding (8) yields 6 nonlinear constraints for $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \gamma_3)^{\mathrm{T}}$, which is however uneasy to solve. We then define a new vector $\boldsymbol{y} = (y_1, y_2, y_3, y_4, y_5, y_6)^{\mathrm{T}} = (\gamma_1^2, \gamma_1\gamma_2, \gamma_1\gamma_3, \gamma_2^2, \gamma_2\gamma_3, \gamma_3^2)^{\mathrm{T}}$ and form a linear system $C\boldsymbol{y} = \boldsymbol{z}$ which has the unique solution with $\boldsymbol{z}$ generated from I_3. Afterwards, it is easy to recover γ_i from $\boldsymbol{y}$ and optimize from initialized γ_i alone with $\gamma_4 = 0$.

After optimization, we again apply SVD to project $\sum_{i=1}^{4}\gamma_i R_i$ onto the space of SO(3), i.e., $R^{init} = U\mathrm{diag}(1,1,1)V^{\mathrm{T}}$ and enforce $\det(R^{init}) > 0$ where $R^{init} = U\Sigma V^{\mathrm{T}}$. Leveraging $A\boldsymbol{x} = 0$ defined in section (1.2), the corresponding translation $\boldsymbol{t}^{init}$ is

$$\boldsymbol{t}^{init} = -(A_2^{\mathrm{T}}A_2)^{-1}A_2^{\mathrm{T}}A_1\boldsymbol{r}^{init} \tag{9}$$

where $A_1 = A_{[:,1:9]}$, $A_2 = A_{[:,10:12]}$, $\boldsymbol{r}_{9\times 1}^{init}$ is reshaped from R^{init}.

2. Ablation Study on Linemod Dataset

Table 1 summarizes the performance of HybridPose using different predicted intermediate representations on Linemod dataset. The overall relative performance is similar to that on Occlusion-Linemod. Specifically, adding symmetry correspondences can boost the performance of rotations. Adding edge vectors can significantly boost the performance of both rotations and translations. Moreover, such

---

[1]If $\det(R) < 0$ we enforce $\det(R) > 0$ by defining $R = U\mathrm{diag}(1,1,-1)V^{\mathrm{T}}$.

[2]The reason of initializing 3 γ_i's is that (8) is unable to provide enough linear constraints for 4 γ_i's and this initialization ensures the convergence of optimization.

	keypoints		keypoints + symmetries		full model	
	Rotation	Translation	Rotation	Translation	Rotation	Translation
ape	1.110°	0.096	1.096°	0.097	0.803°	0.054
benchvise	1.196°	0.034	1.077°	0.032	0.941°	0.028
cam	1.537°	0.060	1.365°	0.057	0.817°	0.029
can	1.246°	0.044	1.102°	0.044	0.757°	0.024
cat	1.041°	0.063	0.978°	0.065	0.663°	0.029
driller	1.483°	0.048	1.278°	0.045	0.748°	0.023
duck	1.049°	0.076	0.967°	0.075	0.726°	0.052
eggbox	1.740°	0.070	1.624°	0.068	0.767°	0.029
glue	1.147°	0.057	1.082°	0.057	0.810°	0.039
holepuncher	1.435°	0.067	1.274°	0.067	0.862°	0.036
iron	1.604°	0.066	1.586°	0.067	0.805°	0.054
lamp	1.608°	0.043	1.408°	0.044	0.771°	0.019
phone	1.073°	0.046	1.068°	0.046	0.644°	0.023
mean	1.328°	0.059	1.223°	0.059	0.778°	0.034

Table 1. **Qualitative evaluation with different intermediate representations (Linemod).** We report errors using two metrics: the median of absolute angular error in rotation, and the median of relative error in translation with respect to object diameter.

improvements are consistent when starting from only using keypoints and when starting from combing keypoints and edge vectors.

3. Stability Analysis for Pose Refinement

In this section, we provide a local stability analysis of the pose regression procedure, which amounts to solving the following optimization problem:

$$
\min_{R,\boldsymbol{t}} \sum_{k=1}^{|\mathcal{K}|} \rho(\|\boldsymbol{r}_{R,\boldsymbol{t}}^{\mathcal{K}}(\boldsymbol{p}_k)\|, \beta_{\mathcal{K}})\|\boldsymbol{r}_{R,\boldsymbol{t}}^{\mathcal{K}}(\boldsymbol{p}_k)\|_{\Sigma_k}^2
$$
$$
+ \frac{|\mathcal{K}|}{|\mathcal{E}|} \sum_{e=1}^{|\mathcal{E}|} \rho(\|\boldsymbol{r}_{R,\boldsymbol{t}}^{\mathcal{E}}(\boldsymbol{v}_e)\|, \beta_{\mathcal{E}})\|\boldsymbol{r}_{R,\boldsymbol{t}}^{\mathcal{E}}(\boldsymbol{v}_e)\|_{\Sigma_e}^2
$$
$$
+ \frac{|\mathcal{K}|}{|\mathcal{S}|} \sum_{s=1}^{|\mathcal{S}|} \rho(r_{R,\boldsymbol{t}}^{\mathcal{S}}(\boldsymbol{q}_{s,1}, \boldsymbol{q}_{s,2}), \beta_{\mathcal{S}}) r_{R,\boldsymbol{t}}^{\mathcal{S}}(\boldsymbol{q}_{s,1}, \boldsymbol{q}_{s,2})^2
$$
$$(10)$$

When predictions are accurate, then the optimal solution of the objective function described above should recover the underlying ground-truth. However, when the predictions possess noise, then the optimal object pose can drift from the underlying ground-truth. Our focus is local analysis, which seeks to understand the interplay between different objective terms defined by keypoints, edge vectors, and symmetry correspondences. Therefore, we assume the noise level of the input is small, and the perturbation of the output is well captured by low-order Taylor expansion of the output.

Our goal is to characterize the relation between the variance of the input noise and the variance of the output pose. We show that incorporating edge vectors and symmetry correspondences generally help to reduce the variance of the output.

The remainder of this section is organized as follows. In Section 3.1, we provide a local stability analysis framework for regression problems. In Section 3.2, we describe the structure of the pose regression and apply this framework to provide a preliminary analysis of the stability of pose regression. In Section 3.3, we provide further analysis on a specific example, which indicates the interactions among keypoints, edge vectors, and symmetry correspondences. Finally, Section 3.4 provide proofs of the propositions in this analysis.

3.1. Local Stability Analysis Framework

We begin with a general result regarding an optimization problem of the following form

$$
\boldsymbol{x}^{\star}(\boldsymbol{y}) := \operatorname*{argmin}_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y}). \tag{11}
$$

In the context of this paper, $\boldsymbol{y}$ encodes the noise associated with the predictions, i.e., keypoints, edge vectors, and symmetry correspondences. $\boldsymbol{x} \in \mathbb{R}^6$ provides a local parameterization of the output, i.e., the object pose. The specific expressions of $\boldsymbol{y}$ and $\boldsymbol{x}$ will be described in Section 3.2.

Without losing generality, we further assume that f satisfies the following assumptions (which are valid in the context of this paper):

- $f(\boldsymbol{x}, \boldsymbol{y}) \geq 0$. Moreover, $f(\boldsymbol{x}, \boldsymbol{y}) = 0$ if and only if $\boldsymbol{x} = 0$ and $\boldsymbol{y} = 0$. This means $\boldsymbol{x}^{\star}(\boldsymbol{0}) = \boldsymbol{0}$, and $(\boldsymbol{0}, \boldsymbol{0})$ is the strict global optimal solution.

- f is smooth and at least C^3 continuous.

- The following Hessian matrix is positive definite in some local neighborhood of $(\boldsymbol{0}, \boldsymbol{0})$:

$$
\begin{bmatrix}
\frac{\partial^2 f}{\partial^2 \boldsymbol{x}} & \frac{\partial^2 f}{\partial \boldsymbol{x} \partial \boldsymbol{y}} \\
\frac{\partial^2 f}{\partial \boldsymbol{x} \partial \boldsymbol{y}} & \frac{\partial^2 f}{\partial^2 \boldsymbol{y}}
\end{bmatrix}.
$$

Our analysis will utilize the following partial derivative of $\boldsymbol{x}^\star$ with respect to $\boldsymbol{y}$.

Proposition 2 *Under the assumptions described above, $\boldsymbol{x}^\star(\boldsymbol{y})$ is unique in the local neighborhood of $\boldsymbol{0}$, and*

$$\frac{\partial \boldsymbol{x}^\star}{\partial \boldsymbol{y}}(\boldsymbol{y}) := -\Big(\frac{\partial^2 f}{\partial^2 \boldsymbol{x}}(\boldsymbol{x}^\star(\boldsymbol{y}), \boldsymbol{y})\Big)^{-1} \frac{\partial^2 f}{\partial \boldsymbol{x} \partial \boldsymbol{y}}(\boldsymbol{x}^\star(\boldsymbol{y}), \boldsymbol{y}). \tag{12}$$

Proof. See Section 3.4.2. $\square$

Since we are interested in local stability analysis, we assume the magnitude of $\boldsymbol{y}$ is small. Thus,

$$\boldsymbol{x}^\star(\boldsymbol{y}) \approx \frac{\partial \boldsymbol{x}^\star}{\partial \boldsymbol{y}}(\boldsymbol{0}) \cdot \boldsymbol{y}. \tag{13}$$

If we further assume $\boldsymbol{y}$ follows some random distribution whose variance matrix if $\mathrm{Var}(\boldsymbol{y})$. Then the variance of the output $\boldsymbol{x}^\star$ is given by

$$\mathrm{Var}(\boldsymbol{x}^\star(\boldsymbol{y}))$$
$$\approx \Big(\frac{\partial^2 f}{\partial^2 \boldsymbol{x}}\Big)^{-1} \cdot \frac{\partial^2 f}{\partial \boldsymbol{x} \partial \boldsymbol{y}} \cdot \mathrm{Var}(\boldsymbol{y}) \Big(\frac{\partial^2 f}{\partial \boldsymbol{x} \partial \boldsymbol{y}}\Big)^T \Big(\frac{\partial^2 f}{\partial^2 \boldsymbol{x}}\Big)^{-1}. \tag{14}$$

Note that in our problem, f consists of non-linear least squares, i.e.,

$$f = \sum \frac{\beta_{i,1}^2 \cdot \|\boldsymbol{r}_i\|_{\Sigma_i}^2}{\beta_{i,2}^2 + \|\boldsymbol{r}_i\|^2}. \tag{15}$$

The following proposition characterizes how to compute $\frac{\partial^2 f}{\partial^2 \boldsymbol{x}}$ and $\frac{\partial^2 f}{\partial \boldsymbol{x} \partial \boldsymbol{y}}$.

Proposition 3 *Under the expression described in (15), the second-order derivatives $\frac{\partial^2 f}{\partial^2 \boldsymbol{x}}$ and $\frac{\partial^2 f}{\partial \boldsymbol{x} \partial \boldsymbol{y}}$ at $(\boldsymbol{0}, \boldsymbol{0})$ are given by*

$$\frac{\partial^2 f}{\partial^2 \boldsymbol{x}} = \sum_i \frac{\beta_{i,1}^2}{\beta_{i,2}^2} \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{x}} \Sigma_i \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{x}}^T \tag{16}$$

$$\frac{\partial^2 f}{\partial \boldsymbol{x} \partial \boldsymbol{y}} = \sum_i \frac{\beta_{i,1}^2}{\beta_{i,2}^2} \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{x}} \Sigma_i \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{y}}^T \tag{17}$$

Proof. See Section 3.4.3. $\square$

3.2. Structure of Pose Stability

We begin by rephrasing the pose-regression problem described in the main paper.

Ground-truth setup. We use the same definition of variables as that in full paper. Recall that $\overline{\boldsymbol{p}}_k$ is coordinates of keypoint in canonical system. Let R^{gt} and $\boldsymbol{t}^{gt}$ be the ground-truth pose. Then the ground-truth 3D location of $\overline{\boldsymbol{p}}_k$ in the camera coordinate system is

$$\overline{\boldsymbol{p}}_k^{gt} = R^{gt}\overline{\boldsymbol{p}}_k + \boldsymbol{t}^{gt}, \qquad 1 \le k \le |\mathcal{K}|.$$

Let $\overline{\boldsymbol{p}}_k^{gt} = (p_{k,x}^{gt,3D}, p_{k,y}^{gt,3D}, p_{k,z}^{gt,3D})^T$. Then the ground-truth image coordinates of the projected keypoint $\boldsymbol{p}_k^{gt} \in \mathbb{R}^2$ is given by

$$\boldsymbol{p}_k^{gt} = \Big(\frac{p_{k,x}^{gt,3D}}{p_{k,z}^{gt,3D}}, \frac{p_{k,y}^{gt,3D}}{p_{k,z}^{gt,3D}}\Big)^T = (p_{k,x}^{gt}, p_{k,y}^{gt})^T.$$

Likewise, recall $(\overline{\boldsymbol{q}}_{s1}, \overline{\boldsymbol{q}}_{s2})$ are symmetry correspondence in the world coordinate system, and let

$$\overline{\boldsymbol{q}}_{s1}^{gt} := R^{gt}\overline{\boldsymbol{q}}_{s1} + \boldsymbol{t}^{gt}$$
$$\overline{\boldsymbol{q}}_{s2}^{gt} := R^{gt}\overline{\boldsymbol{q}}_{s2} + \boldsymbol{t}^{gt}$$

denote the transformed points in the camera coordinate system, where $\overline{\boldsymbol{q}}_{si}^{gt} = (q_{si,x}^{gt,3D}, q_{si,y}^{gt,3D}, q_{si,z}^{gt,3D})^T$. So the image coordinates of each symmetry correspondence are given by

$$\boldsymbol{q}_{s1}^{gt} = \Big(\frac{q_{s1,x}^{gt,3D}}{q_{s1,z}^{gt,3D}}, \frac{q_{s1,y}^{gt,3D}}{q_{s1,z}^{gt,3D}}\Big)^T, \quad \boldsymbol{q}_{s2}^{gt} = \Big(\frac{q_{s2,x}^{gt,3D}}{q_{s2,z}^{gt,3D}}, \frac{q_{s2,y}^{gt,3D}}{q_{s2,z}^{gt,3D}}\Big)^T.$$

Noise model. we proceed to describe the noise model used in the stability analysis. In this analysis, we assume each input keypoint is perturbed from the ground-truth location by $\boldsymbol{y}_k = (y_{k,x}, y_{k,y})^T$, i.e.,

$$\boldsymbol{p}_k = \boldsymbol{p}_k^{gt} + \boldsymbol{y}_k.$$

Likewise, we assume each input edge vector is perturbed from the ground-truth edge vector by $\boldsymbol{y}_e = (y_{e,x}, y_{e,y})^T$, i.e.,

$$\boldsymbol{v}_e = \boldsymbol{p}_{e_s}^{gt} - \boldsymbol{p}_{e_t}^{gt} + \boldsymbol{y}_e.$$

Finally, for symmetry correspondences, we assume that $\boldsymbol{q}_{s1}$ is not perturbed), and $\boldsymbol{q}_{s2}$ is perturbed by $\boldsymbol{y}_s = (y_{s,x}, y_{s,y})^T$, i.e.,

$$\boldsymbol{q}_{s1} = \boldsymbol{q}_{s1}^{gt}, \quad \boldsymbol{q}_{s2} = \boldsymbol{q}_{s2}^{gt} + \boldsymbol{y}_s.$$

Local parameterization. We parameterize the 6D object pose locally using exponential map with coefficients ($\boldsymbol{c} \in \mathbb{R}^3, \overline{\boldsymbol{c}} \in \mathbb{R}^3$), i.e.,

$$R = \exp(\boldsymbol{c}\times) \cdot R^{gt}, \quad \boldsymbol{t} = \boldsymbol{t}^{gt} + \overline{\boldsymbol{c}}.$$

Note this parameterization is quite standard for rigid transformations.

Now consider the three terms used in pose regression[3]:

$$\boldsymbol{r}_k^{\mathcal{K}} := \boldsymbol{p}_k - \mathcal{P}_{R,\boldsymbol{t}}(\overline{\boldsymbol{p}}_k),$$
$$\boldsymbol{r}_e^{\mathcal{E}} := \boldsymbol{v}_e - (\mathcal{P}_{R,\boldsymbol{t}}(\overline{\boldsymbol{p}}_{e_s}) - \mathcal{P}_{R,\boldsymbol{t}}(\overline{\boldsymbol{p}}_{e_t})),$$
$$r_s^{\mathcal{S}} := (\hat{\boldsymbol{q}}_{s,1} \times \hat{\boldsymbol{q}}_{s,2})^T R\overline{\boldsymbol{n}}_r.$$

The following proposition characterizes the derivatives between each term and the parameters of the noise model and the parameters of the local parameterization.

---

[3]For convenience, we negate both $\boldsymbol{r}_k^{\mathcal{K}}$ and $\boldsymbol{r}_e^{\mathcal{E}}$ defined in the body of this paper.

Proposition 4 *Define*

$$J_{k,\boldsymbol{c}} = \begin{pmatrix} -p_{k,x}^{gt}p_{k,y}^{gt} & 1+{p_{k,x}^{gt}}^{2} & -p_{k,y}^{gt} \\ -1-{p_{k,y}^{gt}}^{2} & p_{k,x}^{gt}p_{k,y}^{gt} & p_{k,x}^{gt} \end{pmatrix}$$

$$J_{k,\overline{\boldsymbol{c}}} = \frac{1}{p_{k,z}^{gt,3D}} \begin{pmatrix} 1 & 0 & -p_{k,x}^{gt} \\ 0 & 1 & -p_{k,y}^{gt} \end{pmatrix}$$

The derivatives of $\boldsymbol{r}_{R,\boldsymbol{t}}^{\mathcal{K}}(\boldsymbol{p}_k) = \boldsymbol{r}_k^{\mathcal{K}}$ are given by

$$\left(\frac{\partial \boldsymbol{r}_k^{\mathcal{K}}}{\partial \boldsymbol{y}_k}, \frac{\partial \boldsymbol{r}_k^{\mathcal{K}}}{\partial \boldsymbol{c}}, \frac{\partial \boldsymbol{r}_k^{\mathcal{K}}}{\partial \overline{\boldsymbol{c}}} \right) = (I_2, -J_{k,\boldsymbol{c}}, -J_{k,\overline{\boldsymbol{c}}})$$

The derivatives of $\boldsymbol{r}_{R,\boldsymbol{t}}^{\mathcal{E}}(\boldsymbol{v}_e) = \boldsymbol{r}_e^{\mathcal{E}}$ are given by

$$\left(\frac{\partial \boldsymbol{r}_e^{\mathcal{E}}}{\partial \boldsymbol{y}_e}, \frac{\partial \boldsymbol{r}_e^{\mathcal{E}}}{\partial \boldsymbol{c}}, \frac{\partial \boldsymbol{r}_e^{\mathcal{E}}}{\partial \overline{\boldsymbol{c}}} \right) = (I_2, J_{e_t,\boldsymbol{c}} - J_{e_s,\boldsymbol{c}}, J_{e_t,\overline{\boldsymbol{c}}} - J_{e_s,\overline{\boldsymbol{c}}}).$$

Moreover, the derivatives of r_s^S are given by

$$\frac{\partial r_s^S}{\partial \boldsymbol{y}_s} = \begin{pmatrix} n_y^{gt} - n_z^{gt}q_{s1,y}^{gt} \\ -(n_x^{gt} - n_z^{gt}q_{s1,x}^{gt}) \end{pmatrix}^T, \qquad (18)$$

$$\frac{\partial r_s^S}{\partial \boldsymbol{c}} = \left(\overline{\boldsymbol{n}}_r \times (\hat{\boldsymbol{q}}_{s1}^{gt} \times \hat{\boldsymbol{q}}_{s2}^{gt}) \right)^T. \qquad (19)$$

where $\overline{\boldsymbol{n}}^{gt} = R\overline{\boldsymbol{n}}_r = (n_x^{gt}, n_y^{gt}, n_z^{gt})^T$, $\hat{\boldsymbol{q}}_{si}^{gt}$ is homogeneous coordinate of $\boldsymbol{q}_{si}^{gt}$ normalized by camera intrinsic matrix.

Proof. See Section 3.4.1. □

Let $\boldsymbol{y}$ collect all the random variables in a vector. Let $J_{\mathcal{K}}$, $J_{\mathcal{E}}$, and $J_{\mathcal{S}}$ collect the Jacobi matrices for the predicted elements under each type in its column. Note that the size of $J_{\mathcal{S}}$ is 3×3 according to the derivations above. To facilitate the definition below, we reshape $J_{\mathcal{S}}$ as a 6×6 matrix by placing original elements to the upper-left corner, and zeros to elsewhere. Denote $\beta_{\mathcal{E}}$ and $\beta_{\mathcal{S}}$ as the weight in front of each term (without loss of generality, we set $\beta_{\mathcal{K}} = 1$). Then the variance matrix $\mathrm{Var}(\boldsymbol{c}, \overline{\boldsymbol{c}})$ can be approximated by

$$\mathrm{Var}(\boldsymbol{c}, \overline{\boldsymbol{c}}) \approx A^{-1}B\mathrm{Var}(\boldsymbol{y})B^T A^{-1} \qquad (20)$$

where

$$A := J_{\mathcal{K}}J_{\mathcal{K}}^T + \beta_{\mathcal{E}}J_{\mathcal{E}}J_{\mathcal{E}}^T + \beta_{\mathcal{S}}J_{\mathcal{S}}J_{\mathcal{S}}^T$$
$$B := (J_{\mathcal{K}}, \beta_{\mathcal{E}}J_{\mathcal{E}}, \beta_{\mathcal{S}}J_{\mathcal{S}})$$

It we consider $A^{-1}B\mathrm{Var}(\boldsymbol{y})B^T A^{-1}$ as a function of $\beta_{\mathcal{E}}$ and $\beta_{\mathcal{S}}$ and compute its derivatives at $\beta_{\mathcal{E}} = 0$ and $\beta_{\mathcal{S}} = 0$, we obtain

$$\frac{\partial A^{-1}B\mathrm{Var}(\boldsymbol{y})B^T A^{-1}}{\partial \beta_{\mathcal{E}}}$$
$$:= A^{-1}\left(J_{\mathcal{E}}\Sigma_{\mathcal{E}\mathcal{K}}J_{\mathcal{K}}^T + J_{\mathcal{K}}\Sigma_{\mathcal{K}\mathcal{E}}J_{\mathcal{E}}^T \right.$$
$$\left. - J_{\mathcal{E}}J_{\mathcal{E}}^T A^{-1}J_{\mathcal{K}}\Sigma_{\mathcal{K}\mathcal{K}}J_{\mathcal{K}}^T - J_{\mathcal{K}}\Sigma_{\mathcal{K}\mathcal{K}}J_{\mathcal{K}}^T A^{-1}J_{\mathcal{E}}J_{\mathcal{E}}^T \right)A^{-1}.$$

where $\Sigma_{\mathcal{K}\mathcal{K}}$ and $\Sigma_{\mathcal{E}\mathcal{K}}$ are the corresponding components in $\mathrm{Var}(\boldsymbol{y})$. This means whenever

$$J_{\mathcal{E}}\Sigma_{\mathcal{E}\mathcal{K}}J_{\mathcal{K}}^T + J_{\mathcal{K}}\Sigma_{\mathcal{K}\mathcal{E}}J_{\mathcal{E}}^T$$
$$\prec J_{\mathcal{E}}J_{\mathcal{E}}^T A^{-1}J_{\mathcal{K}}\Sigma_{\mathcal{K}\mathcal{K}}J_{\mathcal{K}}^T + J_{\mathcal{K}}\Sigma_{\mathcal{K}\mathcal{K}}J_{\mathcal{K}}^T A^{-1}J_{\mathcal{E}}J_{\mathcal{E}}^T, \quad (21)$$

increasing the value of $\beta_{\mathcal{E}}$ from zero is guaranteed to obtain a positive reduction in the variance matrix (in terms of both the trace-norm and the spectral-norm).

(21) is satisfied when $\Sigma_{\mathcal{K}\mathcal{K}} = I$ and $\Sigma_{\mathcal{K}\mathcal{E}} = 0$. In general, when $\boldsymbol{y}_k$ and $\boldsymbol{y}_e$ are uncorrelated, then it is likely that increasing its value can lead to reduction in the output variance matrix.

A very similar argument can be applied to $\beta_{\mathcal{S}}$, and we omit the details for brevity.

3.3. An Example

We proceed to provide an example that explicitly shows how the variance of $\mathrm{Var}([\boldsymbol{c}, \overline{\boldsymbol{c}}])$ is reduced by incorporating edge vectors and symmetry correspondences. To this end, we consider a simple object that is given by a square, whose normal direction is along the z-axis in the camera coordinate system. We assume this square object has eight keypoints, whose z coordinates are all 1, i.e., $p_{k,z}^{gt,3D} = 1, 1 \leq k \leq 8$. Their x and y image coordinates are:

$$\boldsymbol{p}_1^{gt} = (\delta, \delta), \quad \boldsymbol{p}_2^{gt} = (\delta, 0), \quad \boldsymbol{p}_3^{gt} = (\delta, -\delta),$$
$$\boldsymbol{p}_4^{gt} = (0, \delta), \quad \boldsymbol{p}_5^{gt} = (0, -\delta), \quad \boldsymbol{p}_6^{gt} = (-\delta, \delta),$$
$$\boldsymbol{p}_7^{gt} = (-\delta, 0), \quad \boldsymbol{p}_8^{gt} = (-\delta, -\delta)$$

Moreover, assume that the normal to the reflection plane is $(1, 0, 0)$. The ground-truth symmetry correspondences are dense, and they are in the form of (x, y) and $(-x, y)$, where $0 \leq x \leq 1, -1 \leq y \leq 1$.

With this setup and after simple calculations, we have

$$H_{\mathcal{K}} := \frac{1}{8}\sum_{k=1}^{8} J_k^T J_k$$

$$= \begin{pmatrix} c_1(\delta) & 0 & 0 & 0 & -c_2(\delta) & 0 \\ 0 & c_1(\delta) & 0 & c_2(\delta) & 0 & 0 \\ 0 & 0 & \frac{12\delta^2}{8} & 0 & 0 & 0 \\ 0 & c_2(\delta) & 0 & 1 & 0 & 0 \\ -c_2(\delta) & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{12\delta^2}{8} \end{pmatrix}$$

where $c_1(\delta) = \frac{8+12\delta^2+10\delta^4}{8}$, and $c_2(\delta) = \frac{8+6\delta^2}{8}$.

Likewise, we have

$$H_{\mathcal{E}} := \frac{1}{28} \sum_{e=1}^{28} J_e^T J_e$$

$$= \begin{pmatrix} \frac{11}{7}\delta^4 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{11}{7}\delta^4 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{24}{7}\delta^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{24}{7}\delta^2 \end{pmatrix}$$

Finally, we have

$$H_S := [\int_0^1 (\int_{-1}^1 J_s^T J_s dy) dx, 0; 0, 0]$$

$$= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{4}{3}\delta^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{4}{9}\delta^4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

We proceed to assume the following noise model for the input:

$$\text{Var}(\boldsymbol{y}_k) = \sigma_{\mathcal{K}}^2 I_2, \quad \text{Var}(\boldsymbol{y}_e) = \sigma_{\mathcal{E}}^2 I_2, \quad \text{Var}(y_s) = \sigma_{\mathcal{S}}^2.$$

In other words, noises in different predictions are independent.

Applying Prop. 3.4.3, we have that

$$\text{Var}([\boldsymbol{c}, \overline{\boldsymbol{c}}])$$
$$\approx (H_{\mathcal{K}} + \lambda H_{\mathcal{E}} + \mu H_{\mathcal{S}})^{-1}.$$
$$(\sigma_{\mathcal{K}}^2 H_{\mathcal{K}} + \lambda^2 \sigma_{\mathcal{E}}^2 H_{\mathcal{E}} + \mu^2 \sigma_{\mathcal{S}}^2 H_{\mathcal{S}}).$$
$$(H_{\mathcal{K}} + \lambda H_{\mathcal{E}} + \mu H_{\mathcal{S}})^{-1}$$

$$= \begin{pmatrix} a_1 & 0 & 0 & 0 & a_2 & 0 \\ 0 & a_3 & 0 & a_4 & 0 & 0 \\ 0 & 0 & a_5 & 0 & 0 & 0 \\ 0 & a_4 & 0 & a_6 & 0 & 0 \\ a_2 & 0 & 0 & 0 & a_7 & 0 \\ 0 & 0 & 0 & 0 & 0 & a_8 \end{pmatrix} \quad (22)$$

$a_i, 1 \le i \le 8$ are functions of $\delta, \beta_{\mathcal{E}}, \beta_{\mathcal{S}}$ and $\sigma_{\mathcal{K}}^2, \sigma_{\mathcal{E}}^2, \sigma_{\mathcal{S}}^2$. For simplify, we only analyze a_8, which is

$$a_8 = \frac{\sigma_{\mathcal{K}}^2 \frac{12}{8}\delta^2 + \beta_{\mathcal{E}}^2 \sigma_{\mathcal{E}}^2 \frac{24}{7}\delta^2}{\left(\frac{12}{8}\delta^2 + \frac{24}{7}\beta_{\mathcal{E}}\delta^2\right)^2}$$

It is easy to check that to minimize a_8, the optimal value for $\beta_{\mathcal{E}}$ is given by

$$\beta_{\mathcal{E}} = \frac{\sigma_{\mathcal{K}}^2}{\sigma_{\mathcal{E}}^2}.$$

In other words, incorporating edge vectors is helpful for reducing the velocity of the third dimension of the rotational component.

Similar analysis can be done for other a_i. As the rationale is similar, we omit them for brevity.

Contributions of keypoints, edge vectors, and symmetry correspondences. It is very interesting to study the structure of (22). First of all, all elements are relevant to keypoints. Edge vectors provide full constraints on the underlying rotation. Symmetry correspondences also provide constraints on two dimensions of the underlying rotation. However, by analyzing the structure of $H_{\mathcal{E}}$ and $H_{\mathcal{K}}$, one can see that they do not provide constraints on two dimensions of the underlying translation (albeit on this simple model). This explains why only using edge vectors and symmetry correspondences leads to poor results on object translations.

3.4. Proof of Propositions

3.4.1 Proof of Proposition 4

Derivatives of $r_k^{\mathcal{K}}$ and $r_e^{\mathcal{E}}$. It is straightforward to compute the derivatives of $r_k^{\mathcal{K}}$ and $r_e^{\mathcal{E}}$ with respect to $\boldsymbol{y}_k$ and $\boldsymbol{y}_e$, respectively. In the following, we focus on the derivatives of $r_k^{\mathcal{K}}$ with respect to $(\boldsymbol{c}, \overline{\boldsymbol{c}})$. The derivatives of $r_e^{\mathcal{E}}$ can be obtained by subtracting those of $r_{e_s}^{\mathcal{K}}$ and those of $r_{e_t}^{\mathcal{K}}$.

Recall the local parameterization $R = \exp(\boldsymbol{c}\times)R^{gt}$ and $\boldsymbol{t} = \boldsymbol{t}^{gt} + \overline{\boldsymbol{c}}$. We have

$$\frac{\partial \overline{\boldsymbol{p}}_k}{\partial(\boldsymbol{c}, \overline{\boldsymbol{c}})} = \frac{\partial(\overline{\boldsymbol{p}}_k^{gt} + \boldsymbol{c} \times \overline{\boldsymbol{p}}_k^{gt} + \overline{\boldsymbol{c}})}{\partial(\boldsymbol{c}, \overline{\boldsymbol{c}})}$$
$$= (-\overline{\boldsymbol{p}}_k^{gt}\times, I_3).$$

Using chain rule, we have

$$\frac{\partial \boldsymbol{r}_k^{\mathcal{K}}}{\partial(\boldsymbol{c}, \overline{\boldsymbol{c}})} = -\frac{1}{p_{k,z}^{gt,3D}}\left(\begin{pmatrix} \frac{\partial p_{k,x}^{3D}}{\partial(\boldsymbol{c}, \overline{\boldsymbol{c}})} \\ \frac{\partial p_{k,y}^{3D}}{\partial(\boldsymbol{c}, \overline{\boldsymbol{c}})} \end{pmatrix} - \begin{pmatrix} p_{k,x} \\ p_{k,y} \end{pmatrix} \cdot \frac{\partial p_{k,z}^{3D}}{\partial(\boldsymbol{c}, \overline{\boldsymbol{c}})}\right)$$

$$= -\begin{pmatrix} 0 & 1 & -p_{k,y}^{gt} & \frac{1}{p_{k,z}^{gt,3D}} & 0 & 0 \\ -1 & 0 & p_{k,x}^{gt} & 0 & \frac{1}{p_{k,z}^{gt,3D}} & 0 \end{pmatrix}$$

$$+ \begin{pmatrix} p_{k,x}^{gt} \\ p_{k,y}^{gt} \end{pmatrix} \cdot \begin{pmatrix} p_{k,y}^{gt} & -p_{k,x}^{gt} & 0 & 0 & 0 & \frac{1}{p_{k,z}^{gt,3D}} \end{pmatrix}$$

Derivatives of r_s. Again using chain rule, we have

$$\frac{\partial r_s^S}{\partial \boldsymbol{y}_s} = \begin{pmatrix} (\hat{\boldsymbol{q}}_{s1}^{gt} \times \boldsymbol{e}_1)^T \boldsymbol{n}^{gt} \\ (\hat{\boldsymbol{q}}_{s1}^{gt} \times \boldsymbol{e}_2)^T \boldsymbol{n}^{gt} \end{pmatrix} = \begin{pmatrix} n_y^{gt} - n_z^{gt}q_{s1,y}^{gt} \\ -(n_x^{gt} - n_z^{gt}q_{s1,x}^{gt}) \end{pmatrix}$$

Moreover,

$$\frac{\partial r_s^S}{\partial \boldsymbol{c}} = \frac{\partial \det((\hat{\boldsymbol{q}}_{s1}^{gt} \times \hat{\boldsymbol{q}}_{s2}^{gt}, \boldsymbol{c}, \boldsymbol{n}))}{\partial \boldsymbol{c}} = \boldsymbol{n} \times (\hat{\boldsymbol{q}}_{s1}^{gt} \times \hat{\boldsymbol{q}}_{s2}^{gt}).$$

3.4.2 Proof of Proposition 2

Proof: First of all, any optimal solution $\boldsymbol{x}^\star(\boldsymbol{y})$ is a critical point of f. Therefore, it shall satisfy:

$$\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}^\star, \boldsymbol{y}) = 0. \tag{23}$$

Consider a neighborhood, where $\|\boldsymbol{y}\| \leq \epsilon_1$, and $\|\boldsymbol{x}\| \leq \epsilon_2$. ϵ_2 is chosen so that it contains for each $\boldsymbol{y}$, the critical point with the smallest norm. Assume that $\frac{\partial^2 f}{\partial^2 \boldsymbol{x}}$ is positive semidefinite in this neighborhood.

By contradiction, suppose there exists two distinctive local minimums $\boldsymbol{x}_1(\boldsymbol{y})$ and $\boldsymbol{x}_1(\boldsymbol{y})$ for a given $\boldsymbol{y}$, i.e.,

$$\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_1(\boldsymbol{y}), \boldsymbol{y}) = \frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_2(\boldsymbol{y}), \boldsymbol{y}) = 0. \tag{24}$$

Through integration, (24) yields

$$0 = \int_0^1 \frac{\partial^2 f}{\partial^2 \boldsymbol{x}}(\boldsymbol{x}_1 + t(\boldsymbol{x}_2 - \boldsymbol{x}_1), \boldsymbol{y})(\boldsymbol{x}_2 - \boldsymbol{x}_1)dt$$

$$= \left(\int_0^1 \frac{\partial^2 f}{\partial^2 \boldsymbol{x}}(\boldsymbol{x}_1 + t(\boldsymbol{x}_2 - \boldsymbol{x}_1), \boldsymbol{y})dt \right) \cdot (\boldsymbol{x}_2 - \boldsymbol{x}_1)$$

Since the weighted sum of positive definite matrices is also positive definite. It follows that

$$\int_0^1 \frac{\partial^2 f}{\partial^2 \boldsymbol{x}}(\boldsymbol{x}_1 + t(\boldsymbol{x}_2 - \boldsymbol{x}_1), \boldsymbol{y})dt \succ 0.$$

In other words, it cannot have a zero eigenvalue, with non-zero eigenvector $\boldsymbol{x}_2 - \boldsymbol{x}_1$. In other words, the critical point is unique. Since the second order derivatives are positive definite, then each critical point is also a local minimum.

Computing the derivatives of (23) with respect to $\boldsymbol{y}$, we obtain

$$\frac{\partial^2 f}{\partial^2 \boldsymbol{x}} \cdot \frac{\partial \boldsymbol{x}^\star}{\partial \boldsymbol{y}} + \frac{\partial^2 f}{\partial \boldsymbol{x} \partial \boldsymbol{y}} = 0 \tag{25}$$

3.4.3 Proof of Proposition 3

The proof is straight-forward as $\boldsymbol{r}_i(\boldsymbol{0}, \boldsymbol{0}) = 0$, and

$$\frac{\partial(\frac{\beta_{i,1}\boldsymbol{r}_i}{\sqrt{\beta_{i,2}^2 + \|\boldsymbol{r}_i\|^2}})}{\partial \boldsymbol{x}} = \frac{\beta_{i,1}}{\beta_{i,2}} \cdot \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{x}},$$

$$\frac{\partial(\frac{\beta_{i,1}\boldsymbol{r}_i}{\sqrt{\beta_{i,2}^2 + \|\boldsymbol{r}_i\|^2}})}{\partial \boldsymbol{y}} = \frac{\beta_{i,1}}{\beta_{i,2}} \cdot \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{y}}.$$

References

[1] 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 *Proceedings of the 11th Asian Conference on Computer Vision - Volume Part I*, ACCV'12, pages 548–562, Berlin, Heidelberg, 2013. Springer-Verlag. 1

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