Coupled Iterative Refinement for 6D Multi-Object Pose Estimation

Lahav Lipson Zachary Teed Ankit Goyal Jia Deng
Princeton University

Abstract

We address the task of 6D multi-object pose: given a set of known 3D objects and an RGB or RGB-D input image, we detect and estimate the 6D pose of each object. We propose a new approach to 6D object pose estimation which consists of an end-to-end differentiable architecture that makes use of geometric knowledge. Our approach iteratively refines both pose and correspondence in a tightly coupled manner, allowing us to dynamically remove outliers to improve accuracy. We use a novel differentiable layer to perform pose refinement by solving an optimization problem we refer to as Bidirectional Depth-Augmented Perspective-N-Point (BD-PnP). Our method achieves state-of-the-art accuracy on standard 6D Object Pose benchmarks. Code is available at https://github.com/princeton-vl/Coupled-Iterative-Refinement.

1. Introduction

Given an RGB or RGB-D image containing a set of object instances of known 3D shapes, 6D multi-object pose is the task of detecting and estimating the 6D pose—position and orientation—of each object instance. Accurate poses are important for robotics tasks such as grasping and augmented reality applications involving shape manipulation.

In the standard 6D multi-object pose setup, we are given a set of 3D models of known object instances. Given an RGB or RGB-D input image, the goal is to jointly detect object instances and estimate their 6D object pose. Early work solved this problem by first estimating correspondences between the 3D model and the image [22], producing a set of 2D-3D correspondences, which are then used to obtain 6D object pose using Perspective-n-Point (PnP) solvers [16, 19] or iterative algorithms like Levenberg-Marquardt.

While 2D-3D correspondence is sufficient to solve for 6D pose, it is difficult to obtain accurate correspondence in practice. In many applications, we wish to estimate the pose of poorly textured objects where local feature matching is unreliable. Furthermore, problems such as heavy occlusion, object symmetry, and lighting variation can make detecting and matching local features near impossible. These problems cause classical systems to be too brittle for many use cases which require a greater degree of robustness.

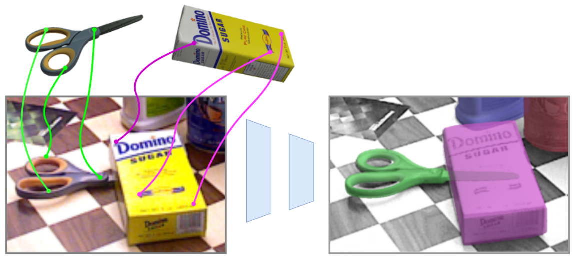


Figure 1. Given an image and collection of 3D models, our method outputs the position and orientation of each object instance.

Recently, many of these issues have been partially addressed using deep learning. A simple approach is to train a network to directly regress 6D poses [18, 20, 36]. Direct pose regression simply learns to map input to output, and makes no use of the fact that the pixels are a perspective projection of a known 3D object. Although direct pose regression can be quite effective in practice, an intriguing question is whether there exist better deep learning methods that take advantage of projective geometry.

Many works on 6D pose have attempted to combine deep learning and projective geometry. One approach is to train a deep network to detect keypoints of a known 3D object [12, 26–29, 34], producing a set of 2D-3D correspondences which can serve as input to a Perspective-n-Point (PnP) solver. Another approach is to impose geometric knowledge in the form of implicit or declarative layers [5, 6]. These works showed that PnP could be implemented as a modular component in end-to-end differentiable architectures. However, both approaches are "one-shot" in the sense that correspondence is predicted once and then used to solve for pose through a PnP solver (differentiable or not); this makes the approaches sensitive to outliers and errors in the correspondences.

We propose a new approach to 6D object pose estimation. Our approach consists of an end-to-end differentiable architecture that makes use of geometric knowledge. The main novelty of our approach over prior work on 6D pose is the use of "coupled iterative refinement": unlike prior work which operates in a single shot setting, we iteratively refine pose and correspondence in a tightly coupled manner, allowing us to dynamically remove outliers to improve accuracy.

Our approach builds on top of the RAFT [32] architecture developed for optical flow (i.e. dense correspondence). The basic idea is to estimate flow between the input image and a set of rendered images of the known 3D object, generating 2D-3D correspondences that are used to solve for pose. Like RAFT, we use a GRU to perform recurrent iterative updates, but at each iteration we update not only flow but also object pose. The flow update and pose update are tightly coupled: the flow update is conditioned on the current pose, and the pose update is conditioned on the flow.

To perform the pose update, we introduce a novel differentiable layer we call "Bidirectional Depth-Augmented PnP (BD-PnP)". This layer is similar to a differentiable PnP solver in that it produces a Gauss-Newton update to object pose by minimizing reprojection error. However, it is novel in two aspects. First, it is bidirectional: it solves for a single pose update to simultaneously satisfy two sets of 2D-3D correspondences, one set defined on the input image, the other set defined on a rendered image. Second, our layer is "depth-augmented": the optimization objective also includes the reprojection error on inverse depth, which we show to be important for improving accuracy.

Our method achieves state-of-the-art accuracy on the YCB-V [4], T-LESS [14] and Linemod (Occluded) [2] RGB-D multi-object BOP [15] pose benchmarks, significantly outperforming prior work. A variant of our method can handle RGB-only input, with performance on par with the current state-of-the-art.

2. Related Work

Classical Approaches Early works on 6D object pose estimation used invariant local features [1, 23] to generate correspondences between 2D image features and 3D model features [24]. Given the set of 2D-3D correspondences, PnP solvers are then used to estimate 6D object pose, that is, the position and orientation of the object in world coordinates [10]. Both closed form [10, 19, 38] and iterative algorithms [25] exist for recovering pose from correspondence. In practice, it is common to use a closed form solution as initialization followed by iterative refinement [19]. Due to the presence of outliers, robust estimation techniques such as RANSAC [9] are typically required. Local features perform well on highly textured objects, but often fail to produce a sufficient number of accurate correspondences on textureless objects.

In this work, we also estimate correspondence between the 3D model and the input image to produce a set of 2D-3D correspondences. However, instead of predicting a set of sparse matches, we predict dense correspondence fields between the input image and rendered views of the 3D model together with per-pixel confidence weights. By predicting dense correspondence, we can ensure a sufficient number of matches allowing us to solve for accurate pose even on textureless objects where classical methods fail.

Learning-based Approaches Several works propose to estimate pose by directly regressing rotation and translation parameters [7, 17, 37].

Other works generate 2D-3D correspondences by detecting or regressing keypoints. One type of keypoint parameterization is object coordinates [3, 6, 26, 29]. Given a canonical pose of an object, the object coordinates represent the position of a 3D point in the coordinate system of the canonical pose. Brachmann et al. [3] showed that a random forest could be used to regress object coordinates from image features. Pix2Pose [26] uses a neural network to regress object coordinates from the image, while BB8 [29] estimates bounding box corners. By regressing object coordinates, these systems produce a dense set of 2D-3D correspondences which can be used to estimate object pose using PnP solvers. BPNP [6] takes this idea a step further and implements the PnP solver as a differentiable network layer. During training, BPNP uses the implicit function theorem to backpropagate gradients through the PnP solver such that the full system can be trained end-to-end. Our work is similar to these approaches in the sense that we also regress 2D-3D correspondences (in the form of optical flow between the input image and rendered views of the 3D model), but we differ by performing coupled iterations where both correspondences and object pose are iteratively refined.

Iterative Refinement It can be challenging to estimate accurate pose in a single-shot setting. This has motivated several works to apply iterative refinement techniques to produce more accurate pose estimates. DeepIM [20] is an iterative "render-and-compare" approach to pose estimation. During each iteration, DeepIM uses the current estimate of object pose to render the 3D model, then uses the render and the image to regress a pose update to better align the image with the render. CosyPose [18] builds on this idea using improved network architectures and rotation paramterizations.

Similar to DeepIM, our approach also includes an outer loop that re-renders the 3D model using the current pose estimate. However, our pose updates are produced not by regression but by our BD-PnP layer that makes use of geometric constraints. In particular, the BD-PnP layer solves for a pose update based on the current estimate of flow.

RAFT-3D [33] applies iterative refinement in the context of scene flow estimation. Like our work, they iterate between optical flow refinement and fitting rigid body transformations. However, RAFT-3D predicts pixelwise transformation fields between pairs of frames using the Dense-SE3 layer, while our work predicts transformations on the object level using our novel BD-PnP layer, which is substantially different from the Dense-SE3 layer.

3. Approach

Our method operates on a single input image and produces a set of object pose estimates (Fig. 1). For simplicity of exposition, we assume RGB-D input unless otherwise noted. Our method can be decomposed into 3 stages: (1) object detection, (2) pose initialization, and (3) pose refinement. The first two stages (object detection and pose initialization) follow the method proposed by CosyPose [18]. Our primary contribution concerns the pose refinement stage, where we seek to transform the initial coarse pose estimates into refined poses with subpixel reprojection error.

Preliminaries Given a textured 3D mesh of an object, we can render images and depth maps of the object from different viewpoints using PyTorch3D [30], with views parameterized by intrinsic and extrinsic parameters

$$\mathbf{G}_i = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \qquad \mathbf{K}_i = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{1}$$

where $\mathbf{G}_i$ is the object pose in camera coordinates. Letting $\mathbf{G}_0$ be the pose for the image and $\{\mathbf{G}_1, ..., \mathbf{G}_N\}$ be the poses for a set of renders, we can define a function which maps points in a render to points in the image

$$\mathbf{x}'_{i\to 0} = \Pi\left(\mathbf{G}_0\mathbf{G}_i^{-1}\Pi^{-1}(\mathbf{x}_i)\right) \tag{2}$$

or from the image to a render

$$\mathbf{x}'_{0\to i} = \Pi\left(\mathbf{G}_i\mathbf{G}_0^{-1}\Pi^{-1}(\mathbf{x}_0)\right) \tag{3}$$

We use depth-augmented pinhole projection functions Π and Π^{-1} which convert not just image coordinates of a point but also its *inverse depth* between frames

$$\Pi(\mathbf{X}) = \begin{bmatrix} X/Z \\ Y/Z \\ 1/Z \end{bmatrix} \quad \Pi^{-1}(\mathbf{x}) = \begin{bmatrix} x/d \\ y/d \\ 1/d \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ d \end{bmatrix} \tag{4}$$

where it is assumed pixels coordinates are normalized using the camera intrinsics.

The goal is to solve for pose $\mathbf{G}_0$ such that Eqn 2 correctly maps points between the image and renders. We can return the object pose in world coordinates by inverting $\mathbf{G}_0$.

Object Candidate Detection Given an input image, we first apply Mask-RCNN [11] to generate a set of object detections and associated labels. We use the pretrained Mask-RCNN weights from CosyPose [18] which were trained on the BOP [15] object classes. We then use the detected bounding boxes to generate crops from the image, segmentation mask, and depth map (in the RGB-D setting). We resize crops to 320×240 and adjust intrinsics accordingly.

Pose Initialization Following detection, our system operates in parallel for each object candidate. Given an object, we start by generating an initial pose estimate $\mathbf{G}^{(0)}$.

We first compute a translation vector $\mathbf{t}_{\text{bbox}}$ which aligns the bounding box of the 3D model to the detected object mask, such that the diameter of the mesh aligns with the projected bounding box. We then render the 3D model using the estimated translation and concatenate the render with the image crop. This input is fed directly to an Resnet-based architecture which regresses a rotation and translation update $(\mathbf{R}, \Delta\mathbf{t})$ where rotation is predicted using the continuous 6D parameterization proposed by Zhou et al. [39]. The initial pose estimate can be written as a 4×4 matrix

$$\mathbf{G}_0^{(0)} = \begin{pmatrix} \mathbf{R} & \mathbf{t}_{\text{bbox}} + \Delta\mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}. \tag{5}$$

We use the pretrained EfficientNet [31] weights from Cosypose [18] for this pose initialization step.

Feature Extraction and Correlation Given our initial pose estimate, we render several viewpoints at our pose estimate as well as centered around it by adding or subtracting $22.5°$ from either pitch, yaw or roll (7 rendered views in total). For each render, our network estimates bidirectional, dense correspondence between the render and the image crop. The object pose of each of the renders is known; the pose of the object in the image crop needs to be estimated.

For all N renders we extract dense $\frac{H}{4} \times \frac{W}{4}$ feature maps. We also apply the same feature extraction network to the image crop using shared weights.

We then build two correlation volumes for each image-render pair, one from the image to the render and another from the render to the image. The correlation volume is computed by taking the dot product between all pair of feature vectors. Like RAFT [32], we pool the last two dimension of each correlation volume to produce a set of 4-level correlation pyramids. These pyramids contain correlation features useful for matching.

3.1. Coupled Iterative Refinement

We use a GRU-based update operator (Fig. 2) to produce a sequence of updates to our pose estimates. The GRU also has a hidden state which gets updated with each iteration.

Let $\mathbf{G}$ be the set of all poses, including both the renders and the image. The poses of the renders are fixed, while the first pose $\mathbf{G}_0$, the pose of the image, is a variable.

Using Eqn. 2, we compute the dense correspondence field bidirectionally between the image and each render. We compute $\mathbf{x}_{i\to 0}$ using Eqn. 2 and $\mathbf{x}_{0\to i}$ using Eqn. 3. The correspondence field $\mathbf{x}_{i\to 0} \in \mathbb{R}^{H\times W\times 3}$ tells us, for every pixel in render i, its estimated 2D location in the image. It is worth noting that the correspondence field is augmented with inverse depth, that is, $\mathbf{x}_{i\to 0}$ contains not just 2D coordinates but also inverse depth.

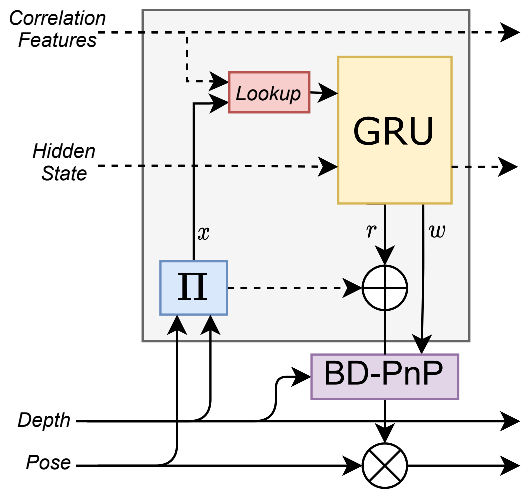


Figure 2. The update operator. A GRU produces revisions $\mathbf{r}$ and confidence weights $\mathbf{w}$. The revisions and confidence weights are used to solve for a pose update.

Correlation Lookup We use $\mathbf{x}_{i\to 0}$ to index from the corresponding correlation pyramid using the lookup operator defined in RAFT [32]. The lookup operator constructs a local grid around each point with radius r and uses the grid to index from each level in the correlation pyramid, producing a total of L correlation features. The result of the lookup operation is a map of correlation features $\mathbf{s}_{i\to 0} \in \mathbb{R}^{H\times W\times L}$. Similarly, we use $\mathbf{x}_{0\to i}$ to produce the correlation features $\mathbf{s}_{0\to i} \in \mathbb{R}^{H\times W\times L}$.

GRU Update For each image-render pair, the correlation features $\mathbf{s}_{i\to 0}$ and the hidden state $\mathbf{h}_{i\to 0}$, together with additional context and depth features described in the appendix, are fed to a 3x3 convolution GRU, which outputs (1) a new hidden state, (2) revisions $\mathbf{r}_{i\to 0} \in \mathbb{R}^{H\times W\times 3}$ to each of the dense correspondence fields, and (3) a dense map of confidence $\mathbf{w}_{i\to 0}$ in the predicted revisions. The revision $\mathbf{r}_{i\to 0}$ represents a new flow estimate in the form of a dense map of corrections that should be applied to the correspondences produced by the current pose estimate. Note that $\mathbf{r}_{i\to 0}$ includes not just revisions for 2D coordinates but also revisions for inverse depth. The revisions for depth are necessary to compensate for the fact that the input sensor depth may be noisy and the corresponding point may be occluded.

We also apply the same GRU for the other direction of the image-render pair. That is, we use the correlation features $\mathbf{s}_{0\to i}$ to produce revisions $\mathbf{r}_{0\to i}$ and confidence map $\mathbf{w}_{0\to i}$. Note that the weights of the GRU are shared across all image-render pairs in both directions.

Bidirectional Depth-Augmented PnP (BD-PnP) The BD-PnP layer converts the predicted revisions $\mathbf{r}$ and confidences $\mathbf{w}$ to a camera pose update $\Delta\mathbf{G}_0$. We first use the revisions to update the correspondence fields

$$\begin{aligned} \mathbf{x}'_{i\to 0} &= \mathbf{x}_{i\to 0} + \mathbf{r}_{i\to 0} \\ \mathbf{x}'_{0\to i} &= \mathbf{x}_{0\to i} + \mathbf{r}_{0\to i} \end{aligned} \tag{6}$$

and define an objective function to minimize the distance between the reprojected coordinates and the revised correspondence

$$\begin{aligned} \mathbf{E}(\mathbf{G}_0) = \sum_{i=1}^{N} ||\mathbf{x}'_{i\to 0} - \Pi(\mathbf{G}_0\mathbf{G}_i^{-1}\Pi^{-1}(\mathbf{x}_i))||^2_{\Sigma_{i\to 0}} + \\ \sum_{i=1}^{N} ||\mathbf{x}'_{0\to i} - \Pi(\mathbf{G}_i\mathbf{G}_0^{-1}\Pi^{-1}(\mathbf{x}_0))||^2_{\Sigma_{0\to i}} \end{aligned} \tag{7}$$

where $||\cdot||_\Sigma$ is the Mahalanobis distance with $\Sigma_{i\to 0} = \text{diag}\ \mathbf{w}_{i\to 0}$. The objective in Eqn. 7 states that we want camera poses $\mathbf{G}_0$ such that the reprojected points match the revised correspondence $\mathbf{x}'_{ij}$. It is important to note that this objective is similar to conventional PnP because it optimizes reprojection error. But unlike conventional PnP, which optimizes a single set of 2D-3D correspondences, our objective is bidirectional because it optimizes two sets of 2D-3D correspondences, one defined on the render and the other defined on the input image. In addition, unlike conventional PnP, our objective also includes reprojection error of inverse depth, which experiments show to be important for improving accuracy.

We linearize Eqn. 7 using the current pose and perform a fixed number of Gauss-Netwon updates (3 during training and 10 during inference). Each Gauss-Newton update produces a pose update $\delta\xi \in \mathfrak{se}(3)$ which is applied to the current pose estimate using retraction on the SE3 manifold

$$\mathbf{G}_0^{(t+1)} = \exp(\delta\xi)\cdot\mathbf{G}_0^{(t)}. \tag{8}$$

Inner and Outer Update Loops For a given set of renders, we run 40 iterations of the update operator. Upon completion, we use the refined pose estimate to re-render a new set of 7 viewpoints and repeat the process. As we show in our experiments, we can trade speed for accuracy by increasing the number of inner and outer iterations.

3.2. RGB Input

To handle RGB input, we can use the current pose $\mathbf{G}_0^{(t)}$ to render the depth from the known 3D model, and proceed as if we have RGB-D input. However, this basic approach is not mathematically sound because the rendered depth is a function of the object pose but is treated as a constant in the optimization. On the other hand, a fully principled treatment is difficult to implement because it requires computing the Jacobian of the rendering function, as well as the derivatives of the Jacobian during backpropagation. As a middle ground, we use the rendered depth to linearize the optimization objective, and introduce depth as a variable in the optimization so that we jointly optimize pose and depth but discard the depth update (full details in the appendix). This revised approach gives better results.

3.3. Training

Each training step, we randomly sample a visible object in the input image and randomly purturb the ground-truth rotation and translation to initialize the pose. Our model is trained to recover the ground truth pose from this initialization. In order to save GPU memory, we use 10 inner-loops and one outer-loop during training, and render only one viewpoint at each training step.

Supervision We supervise on the predicted correspondence revisions and the updated pose estimates from all update iterations in the forward pass, with exponentially increasing loss weights similar to RAFT [32]. Specifically, we supervise on the geodesic L1 distance between the estimated pose and the ground truth pose. The flow is supervised using an L1 endpoint error loss, as is standard for optical flow problems. All ground truth poses in the BOP benchmark [15], which we use for experiments, have a set of discretized symmetries which are considered equivalent with regard to the MSSD and MSPD error metrics. In order to align the loss with the error metrics, we compute the loss using all discretized symmetries and backpropagate the minimum.

4. Experiments

Evaluation Metrics In keeping with the BOP benchmark evaluation, we report Maximum Symmetry-Aware Surface Distance (MSSD) Recall, Maximum Symmetry-Aware Projection Distance (MSPD) Recall, Visible Surface Discrepancy (VSD) Recall, as well as the average over all three metrics. MSSD is the maximum euclidean distance between associated mesh vertices in the predicted and ground truth poses. MSPD is the maximum reprojection error between all associated vertices from the predicted and ground truth poses. Both MSSD and MSPD assume the minimum value across all symmetrically equivalent ground truth poses. VSD is the depth discrepancy between the mesh rendered at the predicted and ground truth poses, measured over pixels where the model is visible. All three metrics are reported as recall percentages over a set of thresholds defined in the BOP benchmark [15], scaled between 0 and 1. Higher is better for all three.

Datasets We evaluate our method on the *varying number of instances of a varying number of objects* in a single RGB-D image (the ViVo task) from the official BOP benchmark [15]. Specifically, we evaluate our method on the YCB-V [4], T-LESS [14] and LM-O (Linemod-Occluded) [2] datasets from the BOP benchmark [15]. Each dataset

Method	Avg.	MSPD	VSD	MSSD
YCB-V [4]				
Ours	**0.893**	**0.885**	**0.871**	**0.924**
CosyPose [18]	0.861	0.849	0.831	0.903
W-PoseNet w/ICP [17]	0.779	0.734	0.779	0.824
Pix2Pose-BOP20 (w/ ICP) [26]	0.780	0.758	0.766	0.817
Koenig-Hybrid-DL-PointPairs [8]	0.701	0.635	0.778	0.690
CDPNv2-BOP20 (w/ ICP) [21]	0.619	0.565	0.590	0.701
EPOS [13]	0.696	0.783	0.626	0.677
Vidal-Sensors18 [35]	0.450	0.347	0.623	0.380
T-LESS [14]				
Ours	**0.776**	0.795	**0.760**	**0.773**
CosyPose [18]	0.728	**0.821**	0.669	0.695
Pix2Pose-BOP20 (w/ ICP) [26]	0.512	0.549	0.438	0.548
Koenig-Hybrid-DL-PointPairs [8]	0.655	0.696	0.580	0.689
CDPNv2-BOP19 (w/ ICP) [21]	0.490	0.674	0.377	0.418
EPOS [13]	0.476	0.635	0.369	0.423
Vidal-Sensors18 [35]	0.538	0.574	0.464	0.575
LINEMOD-Occluded [2]				
Ours	**0.734**	0.824	**0.601**	**0.778**
CosyPose [18]	0.714	**0.826**	0.567	0.748
W-PoseNet w/ICP [17]	0.707	0.793	**0.601**	0.726
Pix2Pose-BOP20 (w/ ICP) [26]	0.588	0.659	0.473	0.631
Koenig-Hybrid-DL-PointPairs [8]	0.631	0.703	0.517	0.675
CDPNv2-BOP20 (w/ ICP) [21]	0.630	0.731	0.469	0.689
EPOS [13]	0.547	0.750	0.389	0.501
Vidal-Sensors18 [35]	0.582	0.647	0.473	0.625

Table 1. Top performing methods on the BOP Benchmark [15]. The MSPD, VSD and MSSD columns are their recall across a range of thresholds (sec. 4). We use the same detector as cosypose [18]

Method	Avg.	MSPD	VSD	MSSD
YCB-V [4]				
Ours	**0.824**	**0.852**	**0.783**	0.835
CosyPose [18]	0.821	0.850	0.772	**0.842**
EPOS [13]	0.696	0.783	0.626	0.677
CDPN [21]	0.532	0.631	0.396	0.570
T-LESS [14]				
Ours	0.715	0.798	0.663	0.684
CosyPose [18]	**0.728**	**0.821**	**0.669**	**0.695**
EPOS [13]	0.476	0.635	0.369	0.423
CDPN [21]	0.490	0.674	0.377	0.418
LINEMOD-Occluded [2]				
Ours	**0.655**	**0.831**	**0.501**	**0.633**
CosyPose [18]	0.633	0.812	0.480	0.606
EPOS [13]	0.547	0.750	0.389	0.501
CDPN [21]	0.624	0.731	0.469	0.612

Table 2. Results on the BOP Benchmark [15] excluding methods that use depth. The MSPD, VSD and MSSD columns are their recall across a range of thresholds (sec. 4). We use the same detector as cosypose [18].

consists of a unique set of objects designed to evaluate a method's accuracy in different real-world settings. The YCB-V dataset consists of 21 household objects with texture and color, the T-LESS dataset consists of 30 highly-similar industry-relevant objects with no texture or color, and the Linemod (Occluded) dataset consists of 15 textureless colored household objects. For each image in the test sets, we must classify and predict the rotation and translation of all visible objects. There are 900 YCB-V test images, 1000 T-LESS test images and the 200 Linemod-Occluded test images. Each image contains 3 to 8 objects.

Training Data On the YCB-V Objects dataset [4], we use the 80K synthetic and 113K real training images provided in the BOP challenge [15]. On the T-LESS [14] dataset, we train using the 50K synthetic and 38K real training images. On the Linemod dataset, we train exclusively using the 50K synthetic training images provided. Please see the appendix for additional implementation details.

4.1. BOP Benchmark Results

RGB-D Results Our approach significantly outperforms all other methods on YCB-V, T-LESS and LM-O for RGB-D input (see Tab. 1). Inline with the BOP benchmark guidelines, all of our RGB-D methods use the exact same hyper parameter settings across all datasets. For each prior work against which we compare, we report their best single-image method per-dataset, with or without ICP, whichever is better. Our reported results do not use ICP.

RGB-Only Results Using our adaptation to RGB-only input, we compare our method to all prior work on the BOP benchmark. Our method performs competitively with the state-of-the-art on the BOP benchmark (See Tab. 2), outperforming on Linemod (Occluded) and YCB-V, and underperforming on T-LESS. Just as with our RGB-D results, all our RGB results were obtained using the same hyper parameter settings.

4.2. Ablation Experiments

All ablation experiments were conducted on held-out scenes from the training data in YCB-V, T-LESS, and LM-O. We mainly report ablation results for the RGB-D setting (Tab. 6). We perform the same ablations for the RGB setting and report the results in the appendix. The better design choices based on RGB-D ablations also perform well for RGB input, and are used to obtain our final results for both settings.

Bidirectional Depth-Augmented PnP Our method benefits from being bidirectional between an image-render pair. Using correspondence from only a single direction in the PnP solver yields less accurate results. In addition, depth augmentation improves accuracy.

Predicting Confidence Weights The predicted confidence weights allow our model to down-weight correspondences

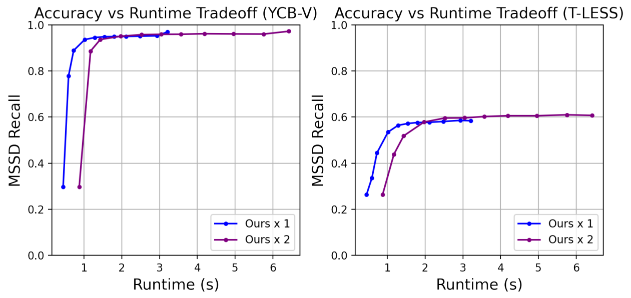


Figure 3. The accuracy / speed tradeoff of our method. Results of one outer update loop (Ours 1x) and two (Ours 2x). **Left:** Accuracy on a held-out split of YCB-V training data. **Right:** Accuracy on a held-out split of T-LESS training data. Our method converges quickly, meaning few inner loops and one outer loop gives good results. Timings are measured on single objects with random rotation and translation perturbation.

which are outliers. This behavior is critical to performance, as uniform confidence over all pixels within the masks is much less accurate (see Tab. 6).

Multi-view Renders In the forward pass, we can arbitrarily add more viewpoints by rendering the object at additional perturbations of the input pose. The memory usage scales linearly with the number of viewpoints, therefore during inference this is tractable. Even without explicitly training using more than one render, rendering seven rotationally-perturbed viewpoints leads to better results across the board during inference (see Tab. 6). Viewpoints generated from excessively large rotation perturbations share too few correspondences with the input image to be useful, while too small perturbations add little novel information. $22.5°$ perturbations work well in all situations.

Coupled Iterative Refinement Tightly coupled iterative refinement of both correspondence and pose performs substantially better than the single-shot approach that solves for pose after predicting correspondence.

Flow and Pose Loss Both the pose and flow loss functions are critical, suggesting that there is strong coupling between the pose updates and flow updates.

Outer Loop It is beneficial to have an outer loop that regenerates the renders with the latest pose estimate.

Handling RGB-only Input Tab. 5 compares the basic approach and the revised approach for handling RGB-only input (see Sec. 3.2). The revised approach performs better.

4.3. Speed vs Accuracy Tradeoff

One can trade off accuracy for speed by varying the number of outer or inner update loops (sec. 3.1). In Fig. 3, we report the accuracy of our method on a held-out portion of the YCB-V and T-LESS training datasets as a function of runtime, for both one and two outer-loops. For our final

	YCB-V [4]		T-LESS [14]		LM-O [2]	
	MSPD Recall	MSSD Recall	MSPD Recall	MSSD Recall	MSPD Recall	MSSD Recall
Bidirectional PnP	**0.924**	**0.955**	**0.685**	**0.582**	**0.828**	**0.788**
Unidirectional PnP (render to image)	0.905	0.941	0.677	0.546	0.605	0.465
Unidirectional PnP (image to render)	0.890	0.917	0.337	0.200	0.811	0.773
Depth-Augmented PnP (predicting depth revisions)	**0.924**	**0.955**	**0.685**	**0.582**	**0.828**	**0.788**
No depth augmentation (no depth revisions)	0.909	0.940	0.678	0.573	0.819	0.784
Predicting per-pixel confidence weights	**0.924**	**0.955**	**0.685**	**0.582**	**0.828**	**0.788**
Uniform confidence	0.721	0.832	0.587	0.424	0.812	0.760
Multiview renders	**0.924**	**0.955**	**0.685**	**0.582**	**0.828**	**0.788**
Single render	0.902	0.941	0.663	0.545	0.744	0.641
Coupled iterative refinement	**0.924**	**0.955**	**0.685**	**0.582**	**0.828**	**0.788**
One-shot (flow by RAFT followed by PnP)	0.562	0.643	0.483	0.275	0.569	0.054
Pose + Flow Loss	**0.924**	**0.955**	**0.685**	**0.582**	**0.828**	**0.788**
Flow Loss Only	0.740	0.733	0.558	0.386	0.804	0.735
Pose Loss Only	0.866	0.919	0.261	0.169	0.615	0.303
4 Outer Loops	**0.933**	**0.958**	**0.694**	**0.601**	**0.831**	0.787
1 Outer Loop	0.924	0.955	0.685	0.582	0.828	**0.788**
No refinement of initial pose	0.194	0.298	0.263	0.167	0.475	0.316

Table 3. Ablation experiments using our method for RGB-D input. We evaluate our method on held-out training images. Initial poses are generated by randomly perturbing the ground truth pose. Options used in our full method are bolded.

	YCB-V [4]		T-LESS [14]		LM-O [2]	
	MSPD Recall	MSSD Recall	MSPD Recall	MSSD Recall	MSPD Recall	MSSD Recall
Revised approach (depth as variable)	**0.833**	**0.751**	**0.649**	**0.474**	0.793	**0.609**
Basic approach (depth as constant)	0.814	0.678	0.637	0.343	**0.808**	0.570

Table 4. Ablation experiments using our method for RGB-Only input. We evaluate our method on a held-out subset of training images. Initial poses are generated by randomly perturbing the ground truth pose. Options used in our full RGB method are bolded. Additional ablations on our RGB method are in the supplemenary material.

results in Tab. 1 and 2, we use 4 outer loops and 40 inner loops, which takes 10.80s per batch of detections. However, Fig. 3 shows that our method converges quickly with few inner loops and one outer loop.

4.4. Robustness

We evaluate the robustness of our method to inaccurate initial pose estimates on the YCB-V test set. In addition to a coarse pose estimation model, Cosypose [18] introduces a regression-based refinement model. In Fig. 6, we plot the accuracy of our model as a function of the rotation error of the initial input pose. For comparison, we also include the refinement model introduced in [18]. Both methods were trained using the same random perturbations of the training data. A limitation of our method is that its ability to refine the poses diminishes for larger initial rotation errors.

5. Qualitative Results

Confidence Weights In the forward pass, our model generates a dense field of confidence weights for all predicted correspondences between an image and rendered pose esti-

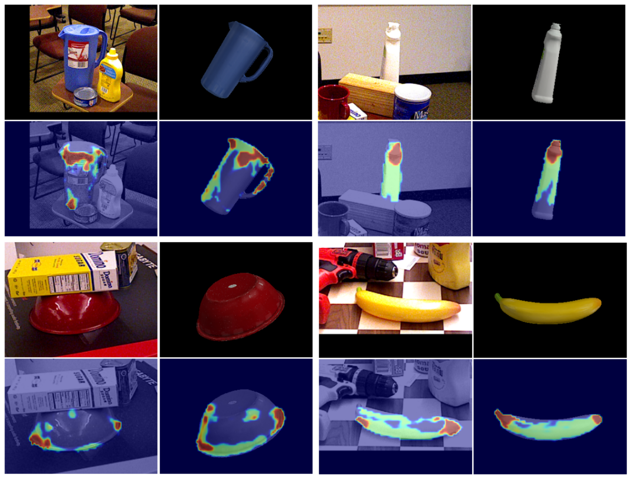


Figure 4. The predicted confidence weights on the YCB-V dataset. The heatmaps provide insight into what surface features are most helpful for pose optimization algorithms. Specifically, our method has low confidence over textureless regions, and high confidence over textured ones, over thin structures, and on edges.

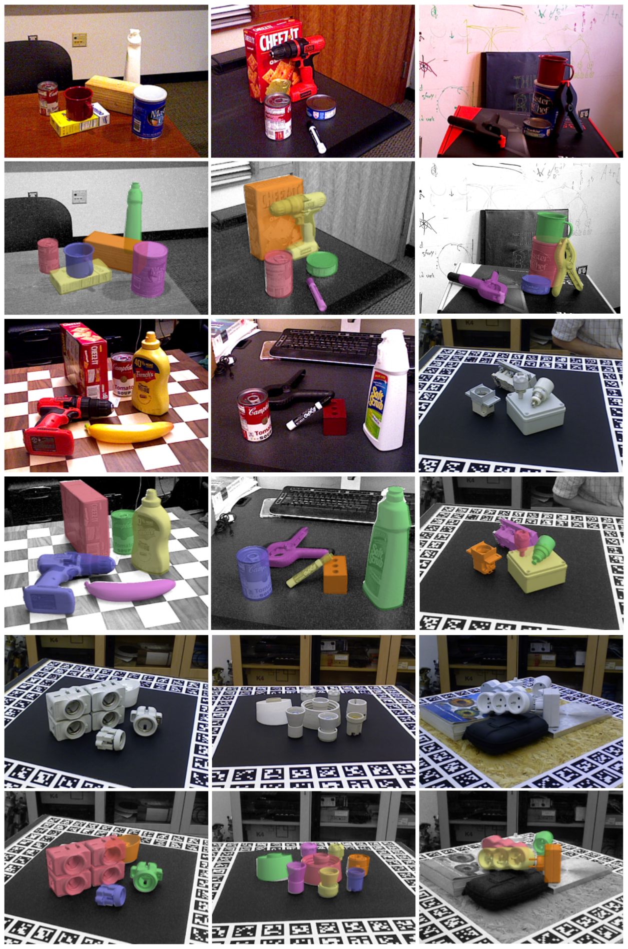


Figure 5. Predictions on the YCB-V and T-LESS test datasets. Known object models are rendered at the predicted poses.

mate. In Fig. 4, we visualize these confidence weights as heatmaps over the images and renders. The heatmaps indicate which parts of the images are most useful for predicting the object's pose. In Fig. 7, we show the correspondences with highest confidence within 5-pixel radii.

Full Image Predictions In Fig. 5, we show the results of our end-to-end method for multi-object pose prediction on the T-LESS and YCB-V test datasets. Additional qualitative results on the YCB-V, T-LESS and LM-O test datasets are included in the appendix.

6. Conclusion

We have introduced a new approach to 6D multi-object pose estimation. Our approach iteratively refines both pose and dense correspondence together using a novel differentiable solver layer. We also introduce a variant of our method for RGB-only input. Our method achieves state-of-the-art accuracy on standard benchmarks.

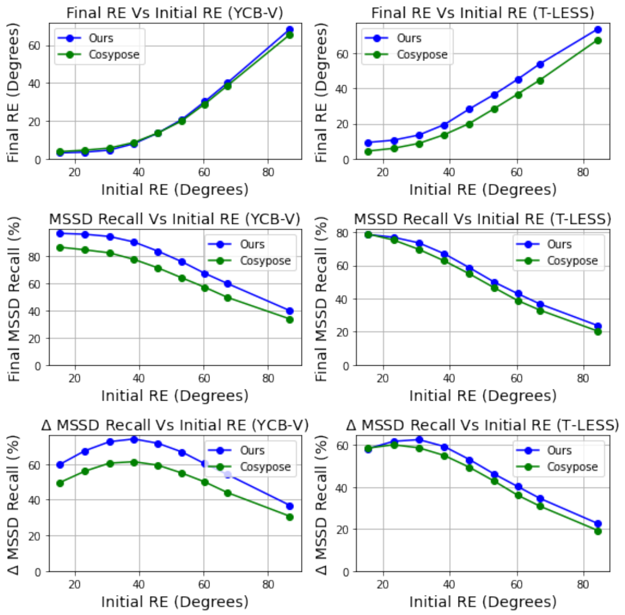


Figure 6. Our model is robust to partially incorrect initial input poses. We plot the accuracy (MSSD recall) and rotation error of the output pose as a function of the rotation error (SO3 geodesic distance, denoted "RE") of the initial input pose. Input poses were randomly rotated from the ground-truth. **Top:** Output rotation error. Our RE is slightly higher on T-LESS. **Middle:** Output MSSD Recall. Our method is more accurate. **Bottom:** Improvement to MSSD Recall over the MSSD Recall of the initial pose.

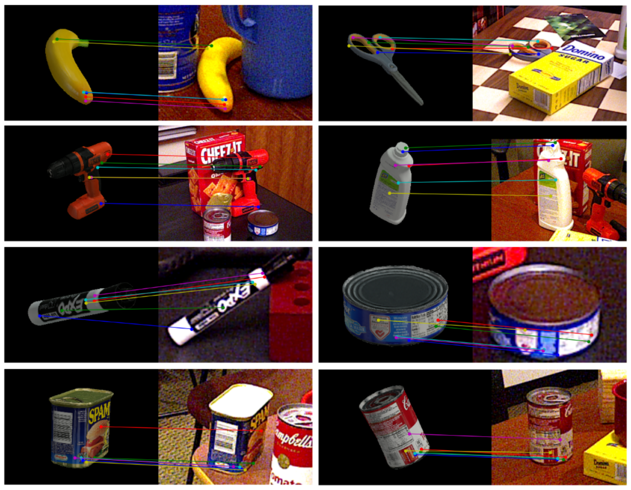


Figure 7. The predicted high-confidence matches on the YCB-V dataset between the input image and the rendered input pose. We apply non-max suppression with 5-pixel radius to the confidence weights and show the most confident predicted correspondences. Our method learns to predict high confidence for matches that are useful for solving for pose.

Acknowledgements This work is partially supported by the National Science Foundation under Award IIS-1942981.

References

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008. 2

[2] 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. 2, 5, 7, 13

[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. 2

[4] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015. 2, 5, 6, 7, 13

[5] Dylan Campbell, Liu Liu, and Stephen Gould. Solving the blind perspective-n-point problem end-to-end with robust differentiable geometric optimization. In *European Conference on Computer Vision*, pages 244–261. Springer, 2020. 1

[6] Bo Chen, Alvaro Parra, Jiewei Cao, Nan Li, and Tat-Jun Chin. End-to-end learnable geometric vision by backpropagating pnp optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8100–8109, 2020. 1, 2

[7] T Do, Trung Pham, Ming Cai, and Ian Reid. Real-time monocular object instance 6d pose estimation. 2019. 2

[8] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 998–1005. Ieee, 2010. 5

[9] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 2

[10] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 25(8):930–943, 2003. 2

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 3

[12] Yisheng He, Wei Sun, Haibin Huang, Jianran Liu, Haoqiang Fan, and Jian Sun. Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11632–11641, 2020. 1

[13] Tomas Hodan, Daniel Barath, and Jiri Matas. Epos: Estimating 6d pose of objects with symmetries. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11703–11712, 2020. 5

[14] Tomáš Hodan, Pavel Haluza, Štepán Obdržálek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis. T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 880–888. IEEE, 2017. 2, 5, 6, 7, 13

[15] Tomas Hodan, Frank Michel, Eric Brachmann, Wadim Kehl, Anders GlentBuch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, et al. Bop: Benchmark for 6d object pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018. 2, 3, 5, 6, 12

[16] Radu Horaud, Bernard Conio, Olivier Leboulleux, and Bernard Lacolle. An analytic solution for the perspective 4-point problem. *Computer Vision, Graphics, and Image Processing*, 47(1):33–44, 1989. 1

[17] 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. 2, 5

[18] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *European Conference on Computer Vision*, pages 574–591. Springer, 2020. 1, 2, 3, 5, 7, 12

[19] 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. 1, 2

[20] 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. 1, 2

[21] Zhigang Li, Gu Wang, and Xiangyang Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7678–7687, 2019. 5

[22] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999. 1

[23] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999. 2

[24] David G Lowe. Local feature view clustering for 3d object recognition. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001. 2

[25] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963. 2

[26] Kiru Park, Timothy Patten, and Markus Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7668–7677, 2019. 1, 2, 5

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

[28] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4561–4570, 2019. 1

[29] 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. 1, 2

[30] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020. 3

[31] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 3

[32] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020. 2, 3, 4, 5, 11, 13

[33] Zachary Teed and Jia Deng. Raft-3d: Scene flow using rigid-motion embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8375–8384, 2021. 2

[34] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 292–301, 2018. 1

[35] Joel Vidal, Chyi-Yeu Lin, Xavier Lladó, and Robert Martí. A method for 6d pose estimation of free-form rigid objects using point pair features on range data. *Sensors*, 18(8):2678, 2018. 5

[36] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017. 1

[37] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017. 2

[38] Yinqiang Zheng, Yubin Kuang, Shigeki Sugimoto, Kalle Astrom, and Masatoshi Okutomi. Revisiting the pnp problem: A fast, general and optimal solution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2344–2351, 2013. 2

[39] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019. 3

Coupled Iterative Refinement for 6D Multi-Object Pose Estimation Appendix

The correlation features and hidden state input to the GRU are described in the main paper. The additional inputs to the GRU are described in this section.

Context Features Following the procedure described in RAFT [32], we use a Resnet [**?**]-based feature extractor to construct context features and an initial hidden state for every image (Fig. **??**). The hidden state is updated with every GRU iteration, while the context features remain unchanged.

Depth Features For each image pair $(i, 0)$, we have inverse-depth maps $\mathbf{z}_i^{-1}$ and $\mathbf{z}_0^{-1}$. We use $\mathbf{x}_{i\to 0}$ to index $\mathbf{z}_0^{-1}$, producing a new inverse-depth map $\mathbf{z}_{i\to 0}^{-1}$. The depth residuals $(\mathbf{z}_0^{-1} - \mathbf{z}_{i\to 0}^{-1})$ implied by the induced correspondences $\mathbf{x}_{i\to 0}$ are provided as input to the GRU before each update. This gives the GRU more information on how well depth has been aligned. Since $\mathbf{x}_{i\to 0}$ are real numbers, we use bilinear interpolation. This procedure is identical for correspondences from image 0 to i.

Solver Residuals The solver residuals from the previous solver iteration are fed to the GRU. These residuals are calculated by taking the difference between the current induced correspondences $\mathbf{x}_{i\to 0}^{(t)}$ and the previous revised correspondences $\mathbf{x}_{i\to 0}'^{(t-1)}$. This allows our model to detect outliers easily as the pixels with unusually high residuals, and thus prune them in the next update.

A. RGB-Only Optimization

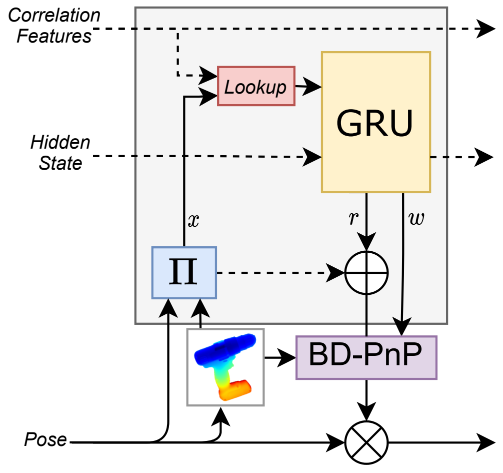
In the RGB-D setting, depth for the input image is given and stays fixed for each pixel regardless of the current pose estimate. In the RGB-Only setting, depth is obtained by rendering the object at the current pose estimate $\mathbf{G}_0$. See Fig. **??**.

$$\mathbf{x}_0^{RGBD} = \begin{bmatrix} x_0 \\ y_0 \\ d_0 \end{bmatrix} \quad \mathbf{x}_0^{RGB} = \begin{bmatrix} x_0 \\ y_0 \\ \mathcal{R}(\mathbf{G}_0) \end{bmatrix} \tag{9}$$

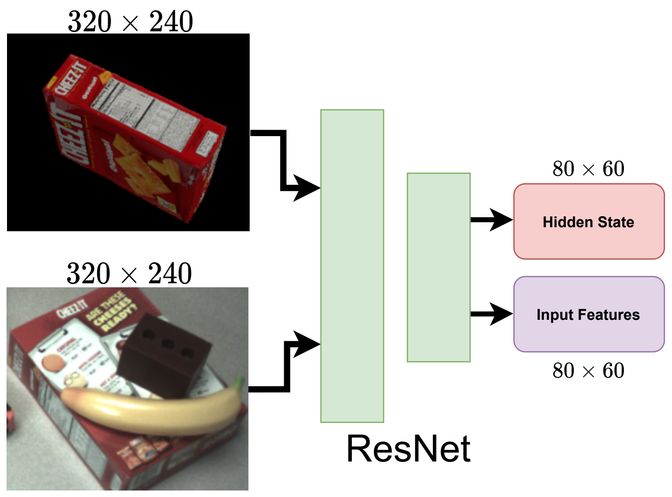
In our objective function, $\mathbf{G}_0$ maps the points between images. In the RGB setting, it also produces the depth for points $\mathbf{x}_0^{RGB}$ in the input image.

$$\textbf{RGBD}: \quad \mathbf{E}(\mathbf{G}_0) = \sum_{i=1}^{N} ||\mathbf{x}'_{i\to 0} - \Pi(\mathbf{G}_0\mathbf{G}_i^{-1}\Pi^{-1}(\mathbf{x}_i))||^2_{\Sigma_{i\to 0}} + \sum_{i=1}^{N} ||\mathbf{x}'_{0\to i} - \Pi(\mathbf{G}_i\mathbf{G}_0^{-1}\Pi^{-1}(\mathbf{x}_0^{RGBD}))||^2_{\Sigma_{0\to i}} \tag{10}$$

$$\textbf{RGB}: \quad \mathbf{E}(\mathbf{G}_0) = \sum_{i=1}^{N} ||\mathbf{x}'_{i\to 0} - \Pi(\mathbf{G}_0\mathbf{G}_i^{-1}\Pi^{-1}(\mathbf{x}_i))||^2_{\Sigma_{i\to 0}} + \sum_{i=1}^{N} ||\mathbf{x}'_{0\to i} - \Pi(\mathbf{G}_i\mathbf{G}_0^{-1}\Pi^{-1}(\mathbf{x}_0^{RGB}))||^2_{\Sigma_{0\to i}} \tag{11}$$



(a) RGB Pose Update Module



(b) Context features and hidden state initialization

In the RGB setting, the depth of $\mathbf{x}_0^{RGB}$ is a function of the pose $\mathbf{G}_0$ in Eq. 11. However, it is difficult to treat it as such in the optimization since our differentiable solver requires calculating the derivative of the Jacobian for the rendering function $\mathcal{R}$. Instead, we treat the depth in the image as a variable to be optimized over jointly with the pose $\mathbf{G}_0$. The resulting optimized pose and depth may be inconsistent with one another since they were treated as separate variables in the optimization. Therefore, we discard the depth update produced by the solver and produce a new depth map by rendering the updated pose. This ensures that the depth is a function of the pose.

B. Implementation Details:

Training Schedule: Our method is implemented in Pytorch [**?**]. All models are initialized from scratch with random weights and trained for 200K steps. During training, we use the AdamW [**?**] optimizer. Final models are trained with a batch size of 12 on two RTX-3090s. Ablation experiments are trained with a batch size of 4. We use an exponential learning rate schedule with a linear increase to 3×10^{-4} over 10000 steps and a 50% drop every subsequent 20000 steps. We use 10^{-5} weight decay. We use the full resolution images for training: 640×480 for YCB-V, 720×540 for T-LESS, and 640×480 for LM-O. In the inner update loop, the correspondence field, confidence weights, depth, hidden state, etc. are maintained at 80×60 spatial resolution, which is $\frac{1}{4}$ of the resolution of the 320×240 input crop. We downsample the input depth by subsampling and use strided convolutions for the image features.

Image Augmentation: We use the same image augmentation as [18] on all three datasets, specifically contrast, hue, sharpness, gaussian blur, and brightness. The depth images in the training data are sparse, so we fill in the gaps using bilinear interpolation. When training our RGB-Only variant, we also swap the ground truth segmentation mask with one predicted by a Mask R-CNN in order to prevent overfitting to the mask boundary.

Multi-view Renders: At test time, our RGB-D method renders 6 additional views at $22.5°$ perturbations around the current pose estimate in the beginning of each outer-loop, for a total of 7. Our RGB-Only method renders 6 *more* views at $45°$ perturbations, for a total of 13.

C. Accuracy Metrics

Our ablation experiments and main results follow the evaluation protocol used in the BOP Challenge [15]. Here, we formally define the error metrics used.

Visible Surface Discrepancy (VSD)

$$e_{\text{VSD}} = \underset{p \in \hat{V} \cup \bar{V}}{\text{avg}} \begin{cases} 0 & \text{if } p \in \hat{V} \cap \bar{V} \, \wedge \, |\hat{D}(p) - \bar{D}(p)| < \tau \\ 1 & \text{otherwise,} \end{cases}$$

where $\hat{D}(p)$ and $\bar{D}(p)$ are the depth maps obtained by rendering the object at the predicted pose and ground-truth pose, respectively. $\hat{V}$ and $\bar{V}$ are visibility masks obtained by comparing each depth map with the sensor depth. VSD treats indistinguishable poses as identical. VSD Recall is the percent of VSD scores less than 10 thresholds ranging from 0.05 to 0.5, with the misalignment tolerance τ ranging from 5% to 50% of the object's diameter.

Maximum Symmetry-Aware Surface Distance (MSSD)

$$e_{\text{MSSD}} = \min_{\mathbf{S} \in S_O} \max_{\mathbf{x} \in V_O} \|\hat{\mathbf{P}}\mathbf{x} - \bar{\mathbf{P}}\mathbf{S}\mathbf{x}\|_2,$$

where S_O are the rotation symmetries of object O and V_O are its vertices. $\hat{\mathbf{P}}$ and $\bar{\mathbf{P}}$ are the predicted and ground truth poses. MSSD is useful for robotic manipulation where the maximum surface deviation is related to the chance of a successful grasp. Compared to the average distance used in ADD/ADI, the maximum distance is less dependant on the sampling density of vertices. MSSD Recall is the percent of MSSD scores less than 10 thresholds ranging from 5% to 50% of the object's diameter.

Maximum Symmetry-Aware Projection Distance (MSPD)

$$e_{\text{MSPD}} = \min_{\mathbf{S} \in S_O} \max_{\mathbf{x} \in V_O} \|\text{proj}(\hat{\mathbf{P}}\mathbf{x}) - \text{proj}(\bar{\mathbf{P}}\mathbf{S}\mathbf{x})\|_2,$$

where V_O, S_O, $\hat{\mathbf{P}}$, $\bar{\mathbf{P}}$ are defined above. MSPD evaluates the perceivable discrepancy, which is important for augmented reality applications. Like MSSD, MSPD also measures the maximum distance instead of the average in order to be robust to the sampling density of vertices. MSPD Recall is the percent of MSPD scores less than 10 thresholds ranging from 5 to 50 (measured in pixels).

	YCB-V [4]		T-LESS [14]		LM-O [2]	
	MSPD Recall	MSSD Recall	MSPD Recall	MSSD Recall	MSPD Recall	MSSD Recall
Bidirectional (depth as variable) PnP	0.833	**0.751**	**0.649**	**0.474**	**0.793**	**0.609**
Unidirectional (depth as variable) PnP [render to image]	**0.834**	0.750	0.639	0.456	0.792	0.593
Unidirectional (depth as variable) PnP [image to render]	0.750	0.630	0.480	0.234	0.645	0.379
Multiview renders	**0.833**	**0.751**	**0.649**	**0.474**	**0.793**	**0.609**
Single render	0.814	0.727	0.619	0.429	0.772	0.569
Predicting per-pixel confidence weights	**0.833**	**0.751**	**0.649**	**0.474**	**0.793**	**0.609**
Uniform confidence	0.694	0.598	0.568	0.377	0.765	0.568
Revised approach (depth as variable)	**0.833**	**0.751**	**0.649**	**0.474**	**0.793**	**0.609**
Depth as variable but do not discard depth update	0.744	0.674	0.520	0.326	0.725	0.497
Use Gradient Clipping	0.833	0.751	0.649	0.474	0.793	0.609
No Gradient Clipping	Training diverges causing NaNs					
Bidirectional context features	0.833	0.751	**0.649**	**0.474**	0.793	**0.609**
Using unidirectional context features	**0.848**	**0.764**	0.647	0.452	**0.807**	0.608
Pose + Flow Loss	**0.833**	**0.751**	**0.649**	**0.474**	**0.793**	**0.609**
Flow Loss Only	0.760	0.666	0.589	0.378	0.741	0.521
Pose Loss Only	0.246	0.190	0.263	0.166	0.271	0.093
Depth-augmented PnP (predicting depth revisions)	**0.842**	**0.765**	0.647	0.465	**0.803**	**0.616**
No depth augmentation (no depth revisions)	0.833	0.751	**0.649**	**0.474**	0.793	0.609

Table 5. Additional ablation experiments using our method for RGB-Only input. We evaluate our method on a held-out subset of training images. Initial poses are generated by randomly perturbing the ground truth pose. Options used in our full RGB method are bolded.

	YCB-V [4]		T-LESS [14]		LM-O [2]	
	MSPD Recall	MSSD Recall	MSPD Recall	MSSD Recall	MSPD Recall	MSSD Recall
Bidirectional context features	**0.924**	**0.955**	**0.685**	**0.582**	**0.828**	**0.788**
Unidirectional context features	0.910	0.944	**0.685**	**0.582**	0.826	0.784

Table 6. Additional ablation experiments using our method for RGB-D input. We evaluate our method on held-out training images. Initial poses are generated by randomly perturbing the ground truth pose. Options used in our full method are bolded.

D. Additional Ablations

Gradient Clipping Treating depth as a variable in the optimization leads to unstable behavior after several thousand training steps. We solve this problem by clipping the gradients of the input to the GRU to a maximum of 0.01 in the middle of the backward pass.

Discarding Depth Update In the RGB setting, we jointly optimize the pose and depth together. We then discard the depth update and generate a new depth map by rendering the updated pose (see Sec A). This works better than jointly optimizing pose and depth together but applying the depth update to the previous estimate instead of discarding it.

RGB-Only Bidirectional (depth as variable) PnP In the RGB setting, bidirectional PnP is helpful on T-LESS while benign on LM-O and YCB-V. On YCB-V, our method converges to an accurate pose even when only using correspondences from the input image to the rendered image (Tab. 5). This indicates that the induced flow is sufficiently accurate even when using a depth *approximation* in the mapping function Π .

RGB-Only Depth-augmented PnP In the RGB setting, depth residuals are helpful on YCB-V and LM-O but not on T-LESS (Tab. 5). The inconsistent benefit could arise from the fact that the depth in the image is generated from the current pose estimate, and therefore predicting the appropriate depth residuals in the GRU is more challenging.

Bidirectional Context Features There is a significant domain gap between the input images and the rendered images. This is due to, among other things, innaccurate meshes resulting from imperfect scans, significant lighting differences and different reflectance properties. We evaluate the importance of extracting contextual features from both images, contrary to RAFT [32] which operates in a single image domain. Tabs. 5 & 6 show that using contextual information from only the source image is sufficient to establish accurate correspondences.

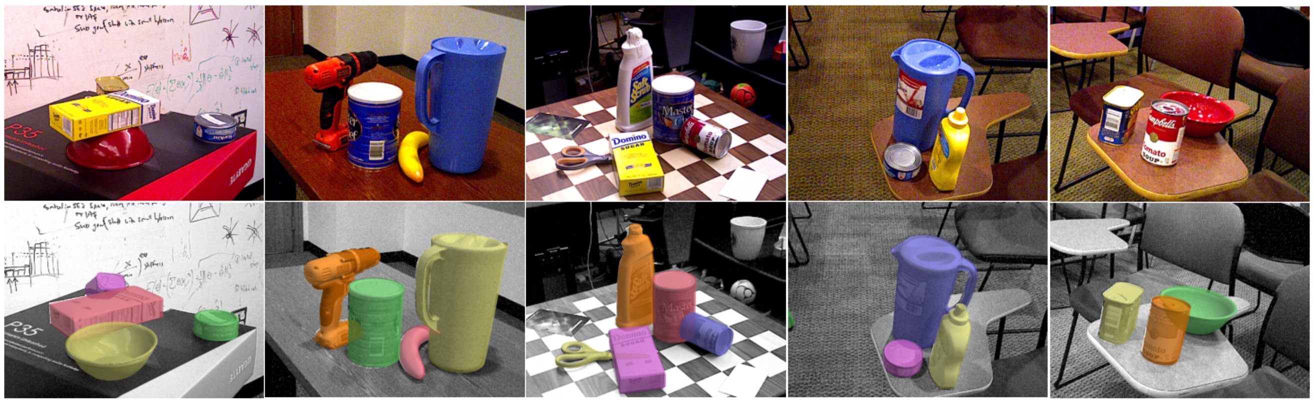
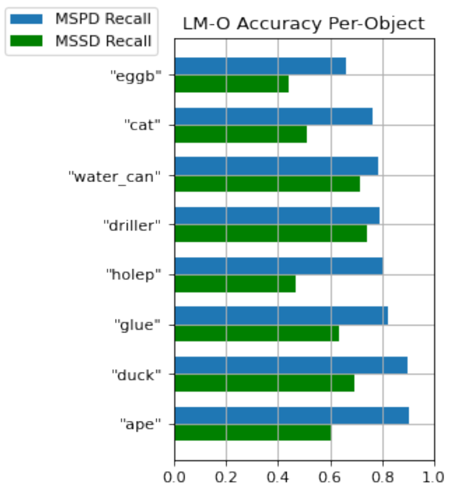
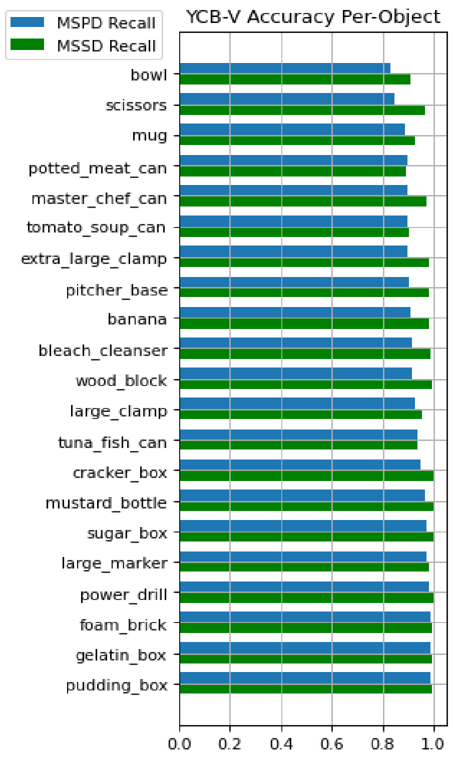


Figure 9. Additional qualitative results on the YCB-V Dataset. Our method incorrectly orients the bowl and spam in the far left image.



(a) Linemod (Occluded) Accuracy



(b) YCB-V Accuracy

Figure 10. Accuracy Per-Object on the Linemod (Occluded) and YCB-V Datasets

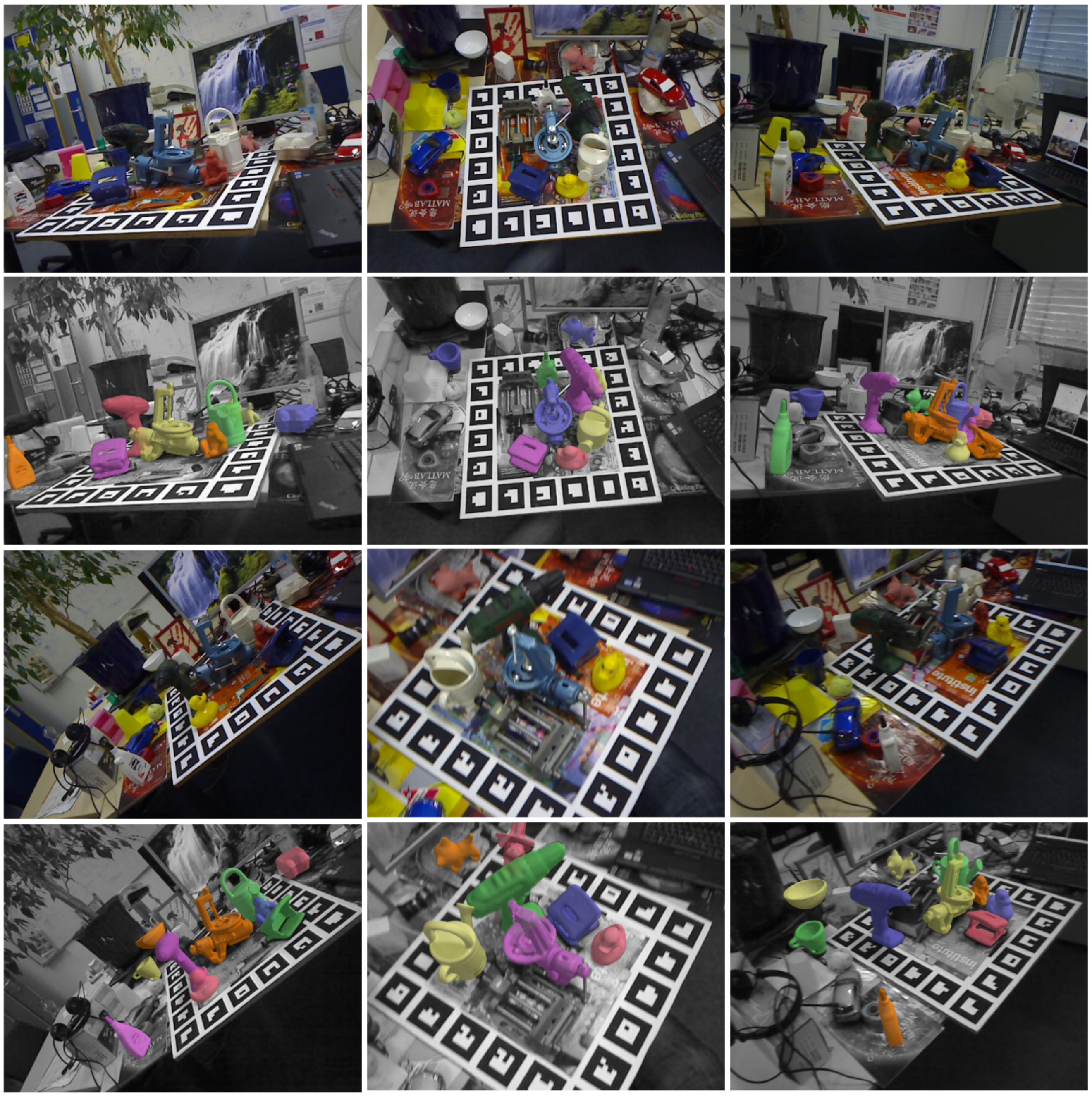


Figure 11. Qualitative results on the Linemod (Occluded) Dataset

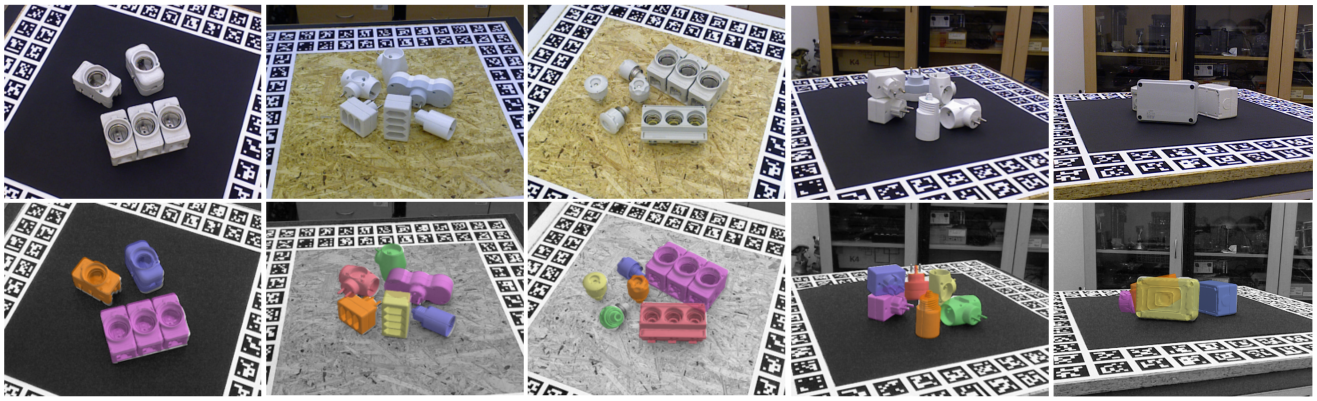


Figure 12. Additional qualitative results on the T-LESS Dataset. Our method incorrectly positions the purple object in the far right image.