MegaPose: 6D Pose Estimation of Novel Objects via Render & Compare

Yann Labbé [1†] **Lucas Manuelli** [2] **Arsalan Mousavian** [2] **Stephen Tyree** [2]
Stan Birchfield [2] **Jonathan Tremblay** [2] **Justin Carpentier** [1] **Mathieu Aubry** [3]
Dieter Fox [2,4] **Josef Sivic** [5]
[1] ENS/Inria [2] NVIDIA [3] LIGM/ENPC [4] University of Washington [5] CIIRC CTU
megapose6d.github.io

Appendix

This appendix is organized as follows. In section 1, we provide the equations of the pose update predicted by the refiner network, and show it depends on the anchor point. In section 2, we give details on the loss used to train the refiner network. Section 3 explains the normalization strategy we apply to the observed and rendered depth images of the RGB-D refiner. Section 4 details the pose hypotheses used during training and inference of the coarse model. In section 5, we provide examples of training images and give details on the data augmentation and training hardware. In section 6, we perform additional ablations to validate (i) the contributions of our coarse network, (ii) the choice of hyper-parameter M. We also provide details on the robot experiments shown in the supplementary video. In section 7, we illustrate qualitatively that our approach is robust to illumination condition variations. Section 8 illustrates the main failure modes of our approach. Finally, section 9 investigates the robustness of our approach with respect to an incorrect 3D model.

The **supplementary video** shows predictions of our approach on real images. We apply our approach in *tracking mode* on several videos. Tracking consists in running the coarse estimator on the first frame of a video sequence, and then applying one iteration of the refiner on each new image, using the prediction in the previous image as the pose initialization at the input of refinement network. This approach can process 20 images per second. The video notably demonstrates the method is robust to occlusion and can be used to perform visually guided robotic manipulation of novel objects.

1 Pose update and anchor point

Pose update. We use the same pose update as DeepIM [1] and CosyPose [2]. The network predicts 9 values corresponding to one 3-vector $[v_x,v_y,v_z]$ to predict an update of the translation of a 3D anchor point, and two 3-vectors e_1,e_2 that define a rotation update explained below. The pose update consists in updating (i) the position of a 3D reference point $\mathcal{O}$ attached on the object, and (ii) the rotation matrix R_{CO} of the object frame expressed in the camera frame (please note the different notations for the

[1]Inria Paris and Département d'informatique de l'ENS, École normale supérieure, CNRS, PSL Research University, 75005 Paris, France.
[3]LIGM, École des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-vallée, France.
[5]Czech Institute of Informatics, Robotics and Cybernetics at the Czech Technical University in Prague.
[†]Work partially done while the author was an intern at NVIDIA.

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

anchor point $\mathcal{O}$ and the object frame O):

$$x_{\mathcal{O}}^{k+1} = \left(\frac{v_x}{f_x^C} + \frac{x_{\mathcal{O}}^k}{z_{\mathcal{O}}^k}\right) z_{\mathcal{O}}^{k+1}, \tag{1}$$

$$y_{\mathcal{O}}^{k+1} = \left(\frac{v_y}{f_y^C} + \frac{y_{\mathcal{O}}^k}{z_{\mathcal{O}}^k}\right) z_{\mathcal{O}}^{k+1}, \tag{2}$$

$$z_{\mathcal{O}}^{k+1} = v_z z_{\mathcal{O}}^k, \tag{3}$$

$$R_{CO}^{k+1} = R(e_1, e_2) R_{CO}^k, \tag{4}$$

where $[x_{\mathcal{O}}^k, y_{\mathcal{O}}^k, z_{\mathcal{O}}^k]$ is the 3D position of the anchor point expressed in camera frame at iteration k, R_{CO}^k a rotation matrix describing the objects orientation expressed in camera frame, f_x^C and f_y^C are the (known) focal lengths that correspond to the (virtual) camera associated with the cropped observed image, and $R(e_1, e_2)$ is a rotation matrix describing the rotation update recovered from e_1,e_2 using [3] by orthogonalizing the basis defined by the two predicted rotation vectors e_1,e_2 similar to [2]. Finally, $[x_{\mathcal{O}}^{k+1}, y_{\mathcal{O}}^{k+1}, z_{\mathcal{O}}^{k+1}]$ and R_{CO}^{k+1} are, respectively, the translation and rotation after applying the pose update. The 3D translation of the anchor point and the rotation matrix R_{CO} are used to define the pose the object.

Dependency to the anchor point. We now show that the predictions the network must make to correct a pose error between an initial pose $\mathcal{T}_{CO}^k$ and a target pose $\mathcal{T}_{CO}^{k+1}$ is independent of the choice of the orientation of the objects coordinate frame O but depends on the choice anchor point $\mathcal{O}$. Let us denote $\mathcal{O}^1$,$\mathcal{O}^2$ two different anchor points, and R_{CO^1},R_{CO^2} the rotation matrices of the object (expressed in the fixed camera frame) for two different choices of object coordinate frames O^1 and O^2. We note $t_{\mathcal{O}^1\mathcal{O}^2} = \mathcal{O}^2 - \mathcal{O}^1 = [x_{12}, y_{12}, z_{12}]$ the 3D translation vector between $\mathcal{O}^2$ and $\mathcal{O}^1$; and $R_{O^1O^2} = R_{CO^1}^T R_{CO^2}$ the rotation of coordinate frame O^2 expressed in O^1. For one choice of anchor point and object frame, e.g. $\mathcal{O}_1$ and R_{CO^1}, we derive the predictions the network has to make to correct the error using equations (1),(2),(3),(4):

$$v_x^1 = f_x^c \left(\frac{x_{O^1}^{k+1}}{z_{O^1}^{k+1}} - \frac{x_{O^1}^k}{z_{O^1}^k}\right) \tag{5}$$

$$v_y^1 = f_y^C \left(\frac{y_{O^1}^{k+1}}{z_{O^1}^{k+1}} - \frac{y_{O^k}^k}{z_{O^k}^k}\right) \tag{6}$$

$$v_z^1 = \frac{z_{O^1}^{k+1}}{z_{O^1}^k} \tag{7}$$

$$R^1 = R_{CO^1}^{k+1} \left(R_{CO^1}^k\right)^T, \tag{8}$$

and similar for 2 by replacing the superscript. From these equations, we derive:

$$v_x^1 - v_x^2 = f_x^c \left(\frac{x_{O^1}^{k+1}}{z_{O^1}^{k+1}} - \frac{x_{O^1}^k}{z_{O^1}^k} - \frac{x_{O^2}^{k+1}}{z_{O^2}^{k+1}} + \frac{x_{O^2}^k}{z_{O^2}^k}\right) \tag{9}$$

$$v_y^1 - v_y^2 = f_y^C \left(\frac{y_{O^1}^{k+1}}{z_{O^1}^{k+1}} - \frac{y_{O^1}^k}{z_{O^1}^k} - \frac{y_{O^2}^{k+1}}{z_{O^2}^{k+1}} + \frac{y_{O^2}^k}{z_{O^2}^k}\right) \tag{10}$$

$$v_z^1 - v_z^2 = \frac{z_{O^1}^{k+1}}{z_{O^1}^k} - \frac{z_{O^2}^{k+1}}{z_{O^2}^k} \tag{11}$$

$$R^1 \left(R^2\right)^T = R_{CO^1}^{k+1} \left(R_{CO^1}^k\right)^T R_{CO^2}^k \left(R_{CO^2}^{k+1}\right)^T = R_{CO^1}^{k+1} R_{O^1O^2} R_{O^2C}^{k+1} = Id. \tag{12}$$

From eq. (12), we have $R^1 \left(R^2\right)^T = Id$. In other words, the rotation matrices that the network must predict to correct the errors in scenarios 1 and 2 are the same. The network predictions for the rotation components thus do not depend on the choice of the choice of object coordinate system. However the other components of the translation cannot be simplified further. For example, derivations of eq. (11) leads to $v_z^1 - v_z^2 = \frac{z_{12}\left(z_{\mathcal{O}^1}^{k+1} - z_{\mathcal{O}^1}^k\right)}{z_1^k (z_1^k + z_{12})}$ which is non-zero in the general case where O^1 and O^2 are different

and there is an error between the initial and target poses. This proves that different choices of anchor point leads to different predictions. For the network to generalize to a novel object, the network be able to infer the 3D position of the anchor point on this object. We achieve this by rendering multiple views of the objects in which the anchor point reprojects to the center of each image as explained in Section 3.1 of the main paper.

2 Refiner loss

Our refiner network is trained using the same loss as in CosyPose [2], but without using symmetry information on the objects because it is not typically not available for large-scale datasets of CAD models like ShapeNet or GoogleScannedObjects. We first define the distance $D_O(\mathcal{T}_1,\mathcal{T}_2)$ to measure the distance between two 6D poses represented by transformations $\mathcal{T}_1$ and $\mathcal{T}_2$ using the 3D points $\mathcal{X}_O$ of an object O:

$$D_O(\mathcal{T}_1,\mathcal{T}_2) = \frac{1}{|\mathcal{X}_O|} \sum_{x \in \mathcal{X}_O} |\mathcal{T}_1 x - \mathcal{T}_2 x|, \tag{13}$$

where $|\cdot|$ is the L_1 norm. In practice, we uniformly sample 2000 points on the surface of an object's CAD model to compute this distance. We also define the pose update function F which takes as input the initial estimate of the pose $\mathcal{T}^k_{CO}$, the predictions of the neural network $[v_x,v_y,v_z]$ and R, and outputs the updated pose:

$$\mathcal{T}^{k+1}_{CO} = F(\mathcal{T}^k_{CO},[v_x,v_y,v_z],R), \tag{14}$$

where the closed form of function F is expressed in equations (1) (2) (3) (4). We also write $[v^\star_x,v^\star_y,v^\star_z]$ and $R^\star$ as the target predictions, i.e. the predictions such that $\mathcal{T}^\star_{CO} = F(\mathcal{T}^k_{CO},[v^\star_x,v^\star_y,v^\star_z],R^\star)$, where $\mathcal{T}^\star_{CO}$ is the ground truth camera-object pose. The loss used to train the refiner is the following:

$$\mathcal{L} = \sum_{k=1}^{K} D_O(F(\mathcal{T}^k_{CO},[v_x,v_y,v^\star_z],R^\star),\mathcal{T}^\star_{CO}) \tag{15}$$

$$+ D_O(F(\mathcal{T}^k_{CO},[v^\star_x,v^\star_y,v_z],R^\star),T^\star_{CO}) \tag{16}$$

$$+ D_O(F(\mathcal{T}^k_{CO},[v^\star_x,v^\star_y,v^\star_z],R),\mathcal{T}^\star_{CO}), \tag{17}$$

where D_O is the distance defined in eq. (13) and K is the number of training iterations. The different terms of this loss separate the influence of: xy translation (15), relative depth (16) and rotation (17). We sum the loss over $K=3$ refinement iterations to imitate how the refinement algorithm is applied at test time but the error gradients are not backpropagated through rendering and iterations. For simplicity, we write the loss for a single training sample (i.e. a single object in an image), but we sum it over all the samples in the training set.

3 Depth normalization

When depth measurements are available, the observed depth image and depth images of the renderings are concatenated with the images, as mentioned in Section 3.1 of the main paper. At test time, the objects may be observed at different depth outside of the training distribution. In order for the network to become invariant to the absolute depth values of the inputs, we normalize both observed and rendered depth. Let us denote D a depth image (rendered or observed are treated similarly). We apply the following operations to D. (i) Clipping of the metric depth values of D to lie between 0 and $z^k_{\mathcal{O}}+1$, where $z^k_{\mathcal{O}}$ is the depth of the anchor point on the object in the input pose at iteration k:

$$D \leftarrow \text{clip}(D^k,0,z^k_{\mathcal{O}}+1), \tag{18}$$

and (ii) centering of the depth values:

$$D \leftarrow \frac{D}{z^k_{\mathcal{O}}} - 1. \tag{19}$$

4 Pose hypotheses in the coarse model

Training hypotheses. Given the ground truth object pose $\mathcal{T}^{\star}_{CO}$, we generate a perturbed pose $\mathcal{T}'_{CO}$ by applying random translation and rotation to $\mathcal{T}'_{CO}$. The parameters of this (small) perturbation are sampled from the same distribution as the distribution used to sample the perturbed poses the refiner network is trained to correct. The translation is sampled from a normal distribution with a standard deviations of $(0.02, 0.02, 0.05)$ centimeters and rotation is sampled as random Euler angles with a standard deviation of 15 degrees in each axis.

We then define several poses that depend on $\mathcal{T}'_{CO}$ and cover a large variety of viewing angles of the object. We define a cube of size $2z'_{\mathcal{O}}$, where $z'_{\mathcal{O}}$ is the z component of the 3D translation in the pose $\mathcal{T}'_{CO}$. The CAD model of the object observed under orientation R'_{CO} is placed at the center of the cube. We then place 26 cameras at the locations of each corner, half-side and face centers of the cube. By construction, one of these cameras, which we denote C^0, has the same camera-object orientation as $\mathcal{T}'_{CO}$, and all others $\{C^i\}_{i=1..25}$ correspond to cameras observing the object under viewpoints which are sufficiently far from R_{C^0O} and outside the basin of attraction of the refiner by construction. In addition, we apply inplane rotations of $90°$, $180°$ and $270°$ to each camera, which leads to a total of $26*4=104$ cameras with one positive and 103 negatives.

We mark $\mathcal{T}_{C^0O}$ as a *positive* for the coarse model because the error between $\mathcal{T}_{C^0O}$ and $\mathcal{T}^{\star}_{CO}$ lies within the basin of attraction of the refiner. All other cameras are marked as negatives. During training, the positives account for around 30% of the total numbers of images in a mini-batch.

Test hypotheses. At test time, a 2D detection of the object is available. Let $u_{det}=(u_{det,x}, u_{det,y})$ and $(\Delta u_{det}=\Delta u_{det,x}, \Delta u_{det,y})$ define the center and the size of the approximate 2D bounding box of the object in the image. We start by defining a random camera-object orientation R^p. The anchor point on the object is set to match the center of the bounding box u_{det}. We make a first hypothesis of the depth of the anchor by setting $z^{guess}_{\mathcal{O}^p}=1\mathrm{m}$ and use this initial value to estimate the coordinates $x_{\mathcal{O}^p}$ and $y_{\mathcal{O}^p}$ of the anchor point in the camera frame:

$$x^{guess}_{\mathcal{O}^p} = u_{det,x} \frac{z^{guess}_{\mathcal{O}^p}}{f_x} \tag{20}$$

$$y^{guess}_{\mathcal{O}^p} = u_{det,y} \frac{z^{guess}_{\mathcal{O}^p}}{f_y}, \tag{21}$$

where f_x and f_y are the (known) focal lengths of the camera. We then update the depth estimate $z^{guess}_{\mathcal{O}^p}$ using the following simple strategy. We project the points of the object 3D model using R^p and the initial guess of the 3D position of the anchor point we have just defined. These points define a bounding box with dimensions $\Delta u_{guess,x}=(\Delta u_{guess,x}, \Delta u_{guess,y})$ and the center remains unchanged $u_{guess}=u_{det}$ by construction. We compute an updated depth of the anchor point such that its width and height approximately match the size of the 2D detection:

$$z_{\mathcal{O}^p} = z^{guess}_{\mathcal{O}^p} \frac{1}{2} \left(f_x \frac{\Delta u_{guess,x}}{\Delta u_{det,x}} + f_y \frac{\Delta u_{guess,y}}{\Delta u_{det,y}} \right) \tag{22}$$

and use this new depth to compute $x_{\mathcal{O}^p}$ and $y_{\mathcal{O}^p}$ using equations (20) and (21) that were used to define $x^{guess}_{\mathcal{O}^p}$ and $y^{guess}_{\mathcal{O}^p}$. The rotation R^p and 3-vector $[x_{\mathcal{O}^p}, y_{\mathcal{O}^p}, z_{\mathcal{O}^p}]$ define the pose of hypothesis p. We then use the same strategy used to define the training hypotheses (described above) in order to define 103 additional viewpoints depending on p. We repeat the operation $P=5$ times, for a total of $5*104=520$ pose hypotheses.

5 Training details

Training images. We generate 2 million photorealistic images using BlenderProc [4] as explained in Section 3.2 of the main paper. Randomly sampled images from the training set are shown in Figure 1.

Data augmentation. We apply data augmentation to the synthetic images during training. We use the same data augmentation as CosyPose [2] for the RGB images. It includes Gaussian blur, contrast,

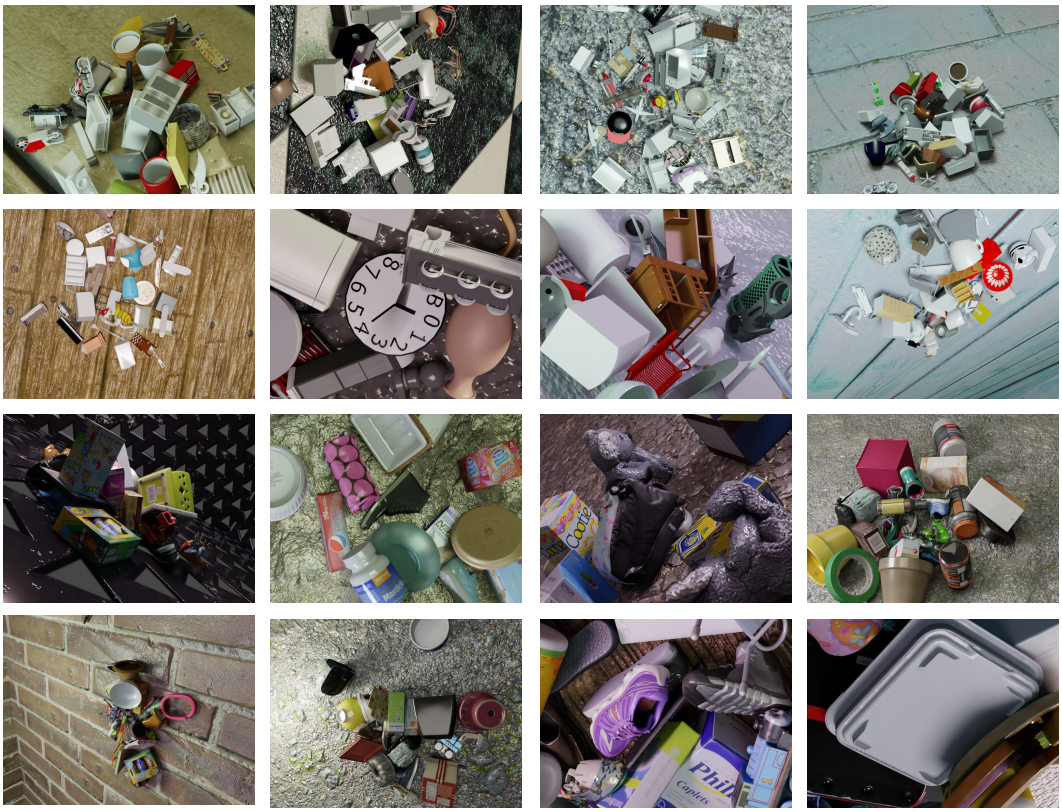

Figure 1: Example of training images. Randomly sampled images from our large-scale synthetic dataset generated with BlenderProc [4]. CAD models from the ShapeNet [5] and GoogleScannedObjects [6] datasets are used.

Pose hypotheses	YCB-V	LM-O
PPF	52.7	34.4
PPF+Zephyr [11]	59.8	45.8
PPF+Our coarse	**61.6**	**52.1**

Table 1: Performance of the coarse model. We compare the performance of our coarse network with Zephyr [11].

brightness, colors and shaprness filters from the Pillow library [7]. For the depth images, we take inspiration from the augmentations used in [8, 9, 10]. Augmentations include blur, ellipse dropout, correlated and uncorrelated noise.

GPU hardware and training time. Training time is respectively 32 and 48 hours for the coarse and refiner models using 32 V-100 GPUs. This training is performed once, and estimating the pose of novel objects does not require any fine-tuning on the target objects.

6 Additional experiments.

Coarse network. In order to evaluate the validate the contributions of our coarse scoring network, we use a set of pose hypotheses generated for novel objects by the commercial Halcon 20.05 Progress software which implements the PPF algorithm described in [22]. Note that these are the same pose hypotheses used in Zephyr [11]. We then find the best hypotheses using the scores of PPF, the scoring network of Zephyr or our coarse network, and report AR results for the LM-O and YCB-V datasets in the table 1. On both datasets, our coarse network is better than the two baselines (PPF and Zephyr) for selecting the best poses among a given set of hypotheses.

Classification-based coarse network. To validate our classification-based coarse model, we consider a regression-based alternative. We trained a regression-based network similar to the coarse model of CosyPose [2] which takes as input six views of the objects covering viewpoints at the poles of a sphere centered on the object. The network collapsed during training, leading to large errors that cannot be recovered by the refiner and a performance close to zero on the BOP datasets. We hypothesize this failure is due to the presence of symmetric objects in our training set which leads to ambiguous gradients during training. This failure could also be attributed to other factors, such as the difficulty to interpret the full 3D geometry of an object with a CNN given six views of its 3D model captured under distant viewpoints.

Number of coarse pose hypotheses. M is an important parameter of our method, which can be used to choose a trade-off between running time and accuracy. The performance significantly improves from M=104 to M=520 (+11.4 AR on BOP5) while keeping the running-time of the coarse model reasonable (1.6 seconds for M=520 compared to 0.3 seconds for M=120). Above M=520, the performance improvement is marginal, e.g. (+0.9 AR) for 4608 hypotheses. Please note that we are still making improvements to our code and have lower runtimes than reported in the paper (1.6s for M=520 compared to the 4s mentioned in line 276).

Robotic grasping experiments. We performed a qualitative real-robot grasping experiment. For multiple YCB-V objects, we manually annotated one grasp with respect to the object's coordinate frame. We then placed the considered object (e.g. the drill in the supplementary video) in a scene among other objects representing visual distractors. The object may be placed on the table or on another object. We then take a single RGB image of the scene using a RealSense D415 camera mounted on the gripper of a Franka Emika Panda robot. We detect the object in 2D using the Mask-RCNN detector from CosyPose [2], and run our Megapose approach composed of coarse and refiner modules for estimating the 6D pose of the object with respect to the camera. We then express the 6D pose of the object and grasp with respect to the robot using the known camera-to-robot extrinsic calibration. We then use a motion planner to generate a robot motion that reaches the estimated grasp pose with the gripper and lift the object. This experiment shown in the supplementary video shows that the pose estimates are of sufficiently high quality to be useful for a robotic manipulation task.

7 Robustness to illumination conditions

In Figure 2, we show qualitative predictions of our approach for the watering can on the TUD-L dataset. Please notice the high accuracy of our approach despite challenging illumination conditions.

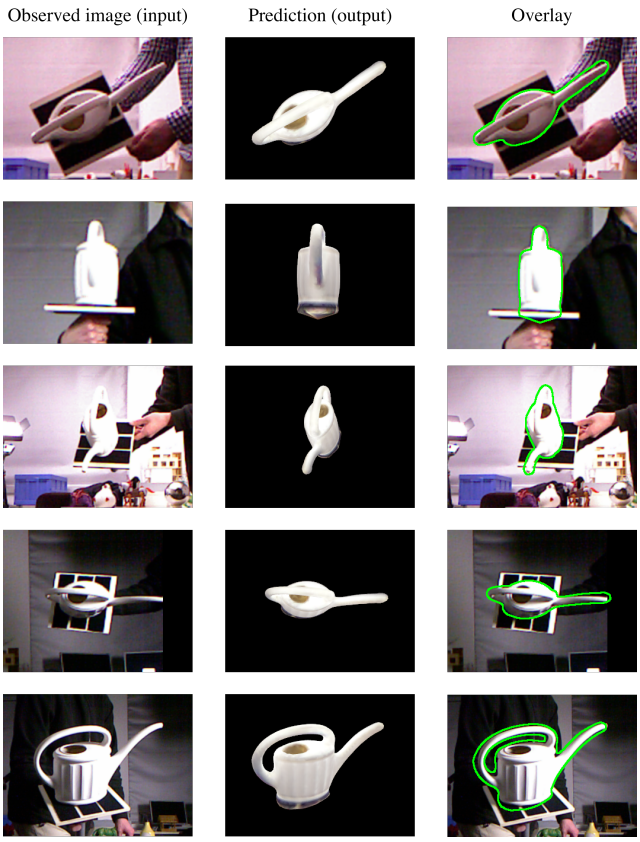

Figure 2: Qualitative examples on the TUD-L dataset. Each row presents one example prediction on a real image. The first column is the real observed image, the second column is the prediction of our approach here illustrated using a rendering of the object's CAD model in the predicted pose, and an overlay of the prediction and output is shown on the right.

8 Failure modes and performance on specific types of objects

We carry out a per-object analysis of the performance of our approach on the YCB-V dataset. For each of the 21 objects of the dataset, we report the percentage of predictions for which the error with the ground truth is within a threshold of 15° in rotation and 5cm in translation. Results are reported in Figure 3.

Next, we illustrate the main failure modes of our approach using a set of objects which have a performance below average on this dataset. Examples of failure cases are presented in Figure 4. We observed three main failure modes to our approach. First, we observe the orientation of a novel object may be incorrectly predicted if the object has a similar visual appearance under different viewpoint. We observed this failure mode in particular for textureless objects such as a red bowl that appears similar whether it is standing upside or it is flipped. Second, we observe that our approach may fail to disambiguate the pose of objects that are asymmetric but for which it is necessary to look at fine details on the objects to disambiguate multiple possible poses. An example is a pair of scissors which have left and right handles with slightly different dimensions. In both of these failure modes, we observed that our refiner gets stuck into a local minimal due to an inaccurate coarse estimate outside of the basin of attraction of our refiner model. Finally, using a CAD model with incorrect scale leads to an incorrect estimation of the depth of the object due to the object scale/depth ambiguity in RGB images. We observe for example that the translation estimates of the wooden block of YCB-V have systematically large error despite the rendering of our prediction correctly matching the contours of the object in the observed image. This is because the scale of the CAD model of the wooden block publicly available does not match the correct dimensions of the real object which was used for annotating the ground truth.

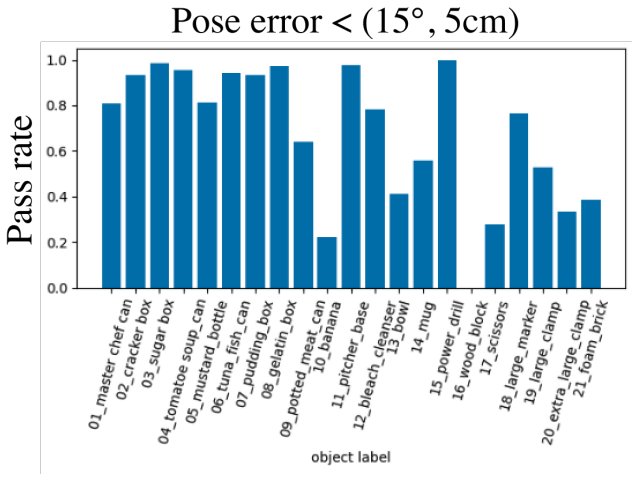

Figure 3: Per-object analysis on the YCB-V dataset. For each object, we report the percentage of estimates for which the error between our pose prediction and the ground truth is within 5 centimeters in translation and 15 degrees in rotation.

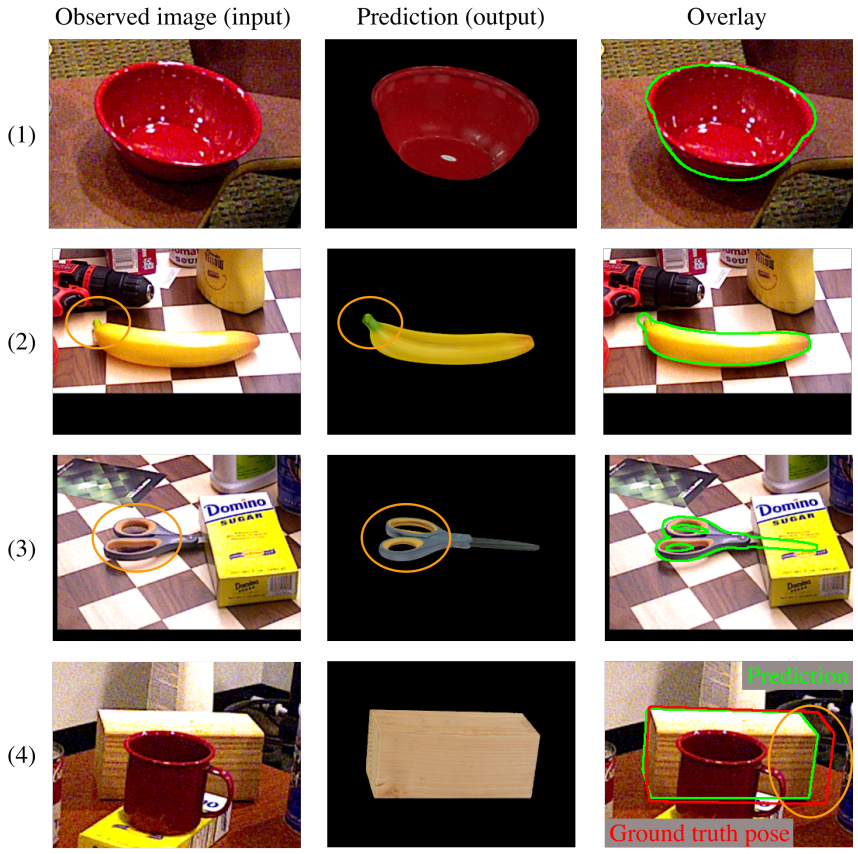

Figure 4: Illustration of the main failure modes of our approach. In (1) and (2), the contours of the object in the predicted poses correctly overlay the observed image, but the pose is incorrect because these objects have a similar appearance under different viewpoints. In (3), our approach fails to correctly distinguish the left and right handles with different dimensions in order to disambiguate the orientation of the asymmetric pair of scissors. In (4), our pose prediction does not match the ground truth annotation, because the CAD model of the wooden block we use for pose estimation has different dimensions that do not match the dimensions of the real objects which was used for annotating the ground truth. Please notice in all examples how the contours of the object in the predicted pose are closely aligned with the contours of the object in the input image.

Figure 5: Predictions using low-fidelity CAD models. In (a) we show the result of our approach on LineMOD Occlusion for three different objects which have only low-fidelity CAD models available. In (1) and (2), the quality of the mesh and textures is poor as illustrated in (b). Notice for example how the annotations on the glue box or the brand of the drill are not readable on the CAD models. In (3), the hole of the watering can does not appear in the CAD model. Despite these discrepancies between the real object and the CAD model, our approach correctly estimates the pose of each object.

9 3D model quality

Our approach can be applied even if the 3D model of the object does not exactly matches the real object. In figure 5, we show examples of correctly estimated poses using low-fidelity CAD models with low-quality textures or geometric discrepancies between the real object and its 3D model.

References

[1] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. DeepIM: Deep iterative matching for 6D pose estimation. In *ECCV*, 2018.

[2] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic. CosyPose: Consistent multi-view multi-object 6D pose estimation. In *ECCV*, 2020.

[3] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. In *CVPR*, 2019.

[4] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam. Blenderproc. *arXiv preprint arXiv:1911.01911*, 2019.

[5] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[6] L. Downs, A. Francis, N. Koenig, B. Kinman, R. Hickman, K. Reymann, T. B. McHugh, and V. Vanhoucke. Google scanned objects: A high-quality dataset of 3D scanned household items. In *ICRA*, 2022.

[7] The pillow imaging library. https://github.com/python-pillow/pillow.

[8] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg. Dex-net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning. In *2018 IEEE International Conference on robotics and automation (ICRA)*, pages 5620–5627. IEEE, 2018.

[9] C. Xie, Y. Xiang, A. Mousavian, and D. Fox. Unseen object instance segmentation for robotic environments. *IEEE Transactions on Robotics*, 37(5):1343–1359, 2021.

[10] B. Wen, C. Mitash, B. Ren, and K. E. Bekris. se (3)-tracknet: Data-driven 6d pose tracking by calibrating image residuals in synthetic domains. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10367–10373. IEEE, 2020.

[11] B. Okorn, Q. Gu, M. Hebert, and D. Held. ZePHyR: Zero-shot pose hypothesis rating. In *ICRA*, 2021.