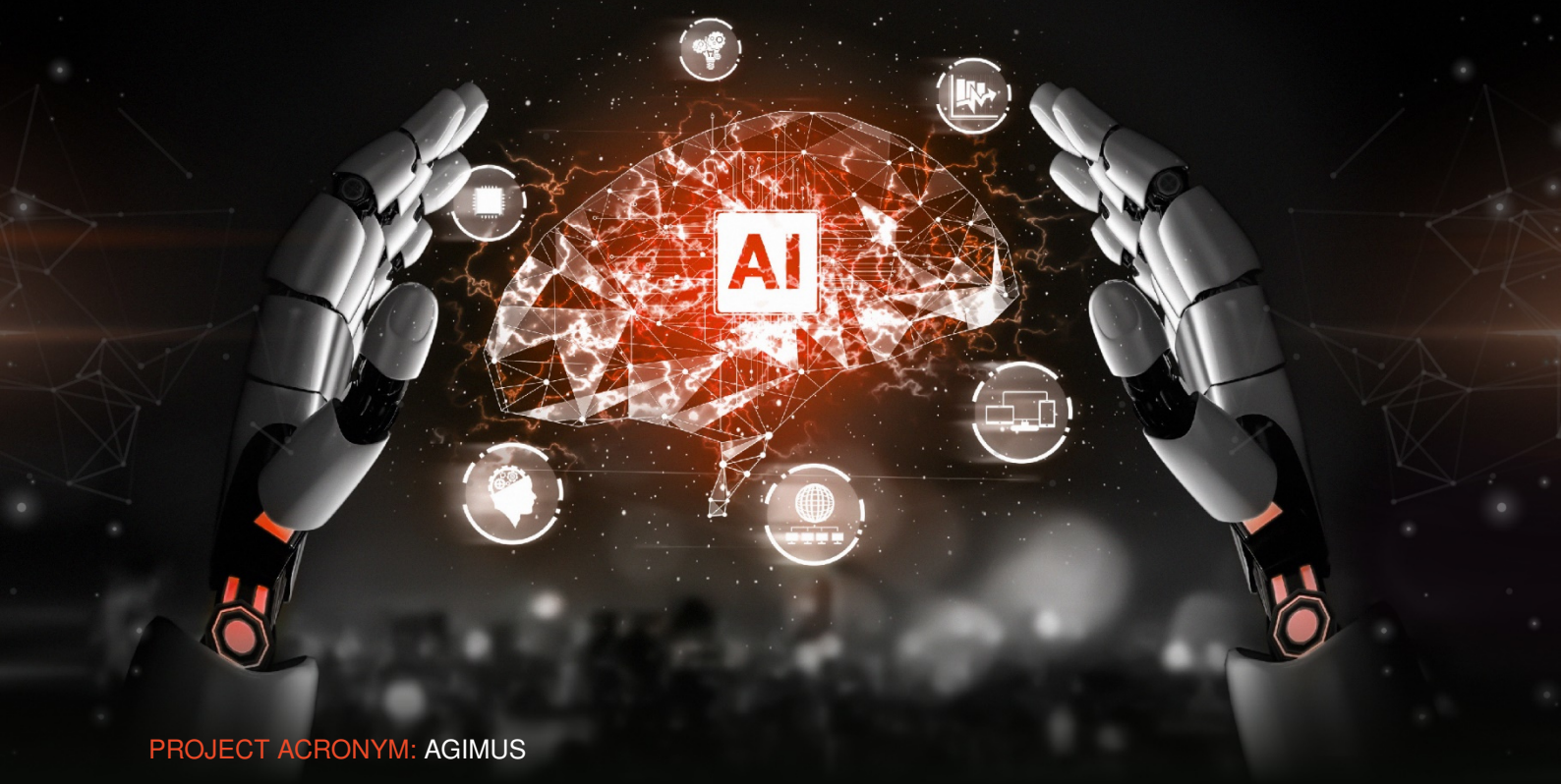




INNOVATIVE ROBOTICS FOR AGILE PRODUCTION

## D4.1: 6D object pose estimation



**PROJECT ACRONYM:** AGIMUS

**PROGRAMME:** Horizon Europe

**GRANT AGREEMENT:** No 101070165

**TYPE OF ACTION:** Horizon Research & Innovation Actions

**START DATE:** 1 October 2022

**DURATION:** 48 months



Funded by  
the European Union

## Document information

Issued by:	CTU
Issue date:	15/09/2022
Due date:	31/09/2022
Work package leader:	CTU
Start date:	1 April 2023
Dissemination level:	PUB (Public)

## Document History

Version	Date	Modifications made by
1.0	15/09/2023	First version released - CTU

## Authors

First name	Last name	Beneficiary
Vladimir	Petrik	CTU
Mederic	Fourmy	CTU
Josef	Sivic	CTU

*In case you want any additional information, or you want to consult with the authors of this document, please send your inquiries to: [vladimir.petrik@cvut.cz](mailto:vladimir.petrik@cvut.cz).*

## Quality reviewers

First name	Last name	Beneficiary
Narcis Miguel	Baños	PAL
Florent	Lamiroux	CNRS
Nicolas	Mansard	CNRS

### **Disclaimer**

*Funded by the European Union under GA no. 101070165. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union. The granting authority cannot be held responsible for them.*

**© AGIMUS Consortium, 2022**

*Reproduction is authorised provided the source is acknowledged.*

### Executive summary

This report was developed in the context of the AGIMUS project, funded by the Horizon Europe Framework Program for Research and Innovation of the European Union for 2021-2027. Its objective is to present the progress mainly towards the specific objective ***SpO1.2: Develop advanced perception combining data-driven training with physics-based models.*** that is part of the ***Objective 1: Significantly accelerate the introduction and deployment of a robotic system to a new agile production environment through advanced perception and understanding of human-centric feedback.***



## Table of Contents

<b>1 Introduction</b>	<b>7</b>
<b>2 Object Pose Estimation</b>	<b>7</b>
2.1 Objects seen during training	8
2.2 Objects unseen during training	8
<b>3 Continuous Tracking</b>	<b>11</b>
3.1 Object localization and tracking (OLT)	13
3.2 Time delay corrector	14
3.3 Evaluation	15
<b>4 Conclusion</b>	<b>15</b>
<b>Reference List</b>	<b>15</b>
<b>A MegaPose paper [8]</b>	<b>17</b>
<b>B Visual MPC paper [9]</b>	<b>38</b>
<b>C Documentation of HappyPose</b>	<b>47</b>

## List of Figures

1	CNOS overview. . . . .	9
2	MegaPose overview. . . . .	10
3	BOP Challenge 2023: 6D localization of unseen objects. . . . .	11
4	BOP Challenge 2023 Award in the category of 6D localization of unseen objects. . . . .	12
5	Overview of the fast continuous perception module. . . . .	13
6	Perception module timeline . . . . .	13
7	Evaluation of the proposed perception module . . . . .	14

## 1 Introduction

Consistent and accurate perception is one of the key needs for robotic manipulation as it drives robot control. This work summarizes our effort towards accurate image-based perception module that is fast enough to be used in closed-loop robot control. The report is based on the following Agimus-related publications:

- **MegaPose: 6D Pose Estimation of Novel Objects via Render & Compare** [8] presented at the Conference on Robot Learning (CoRL) 2022, and
- **Visually guided model predictive robot control via 6D object pose detection and tracking** [9] submitted to IEEE International Conference on Robotics and Automation (ICRA) 2024 conference.

The full version of both papers is attached at the end of the deliverable.

The deliverable is structured as follows:

The object 6D pose estimation methods from images are described in Sec. 2. It covers the pose estimation method from RGB(D) data for (i) objects known at the train time *i.e.* CosyPose [7] as well as for (ii) objects unseen during training *i.e.* MegaPose [8]. Both methods were partially reimplemented in the Agimus project into a single open-source framework called **HappyPose** that is available at <https://github.com/agimus-project/happypose>. The CosyPose and MegaPose documentation has been combined and updated into the HappyPose documentation, which is attached at the end of the deliverable.

In Sec. 3 we describe our effort towards speeding up the perception module so that it can be used in closed-loop robot control. Our approach combines *slower* CosyPose [7] pose detection with *faster* image-based 6D pose tracker, called ICG [11]. Our Python wrapper around ICG is publicly available at <https://github.com/MedericFourmy/pyicg>. The presented perception module provides fast and temporally consistent predictions that are required for developing **T4.3: Feedback from multimodal perception (haptic+vision)**.

## 2 Object Pose Estimation

The primary objective of the object pose estimation method is to predict the relative transformation between the camera frame and the frame attached to the object of interest *i.e.*  $T_{co} \in SE(3)$ . Input to the method is an RGB(D) image of the scene, the camera intrinsic calibration, and the database of 3D models of interest. The approach adapted in the project is split into three stages:

- object detection,
- coarse pose estimation, and
- pose refinement.

The implementation of these stages differs for objects seen and unseen during training.

## 2.1 Objects seen during training

For estimating pose of a known object, *i.e.* object seen during training, we use CosyPose [7], the winning method in the BOP Challenge 2020 (5 awards in total, including the best overall method and the best RGB only method).

**Object detection.** The CosyPose uses the MaskRCNN [5] detector that is trained for each dataset of objects; this is possible because the objects are known during training. The output of the detection model is a list of bounding boxes and labels of objects that are shown in the image patches defined by the bounding boxes.

**Coarse pose estimation.** The bounding box of the previous stage is used to heuristically compute the first guess of the pose of the object. The model that corresponds to the detected label is selected from the database and used for rendering in the render-and-compare strategy. The coarse pose estimator first renders the image based on the model and the guessed pose. The rendering is concatenated with the input image patch and used as input to a neural network that is trained to predict update of the pose. The predicted update is applied to the first guess to create a new estimate of the object pose. The coarse model is evaluated only once.

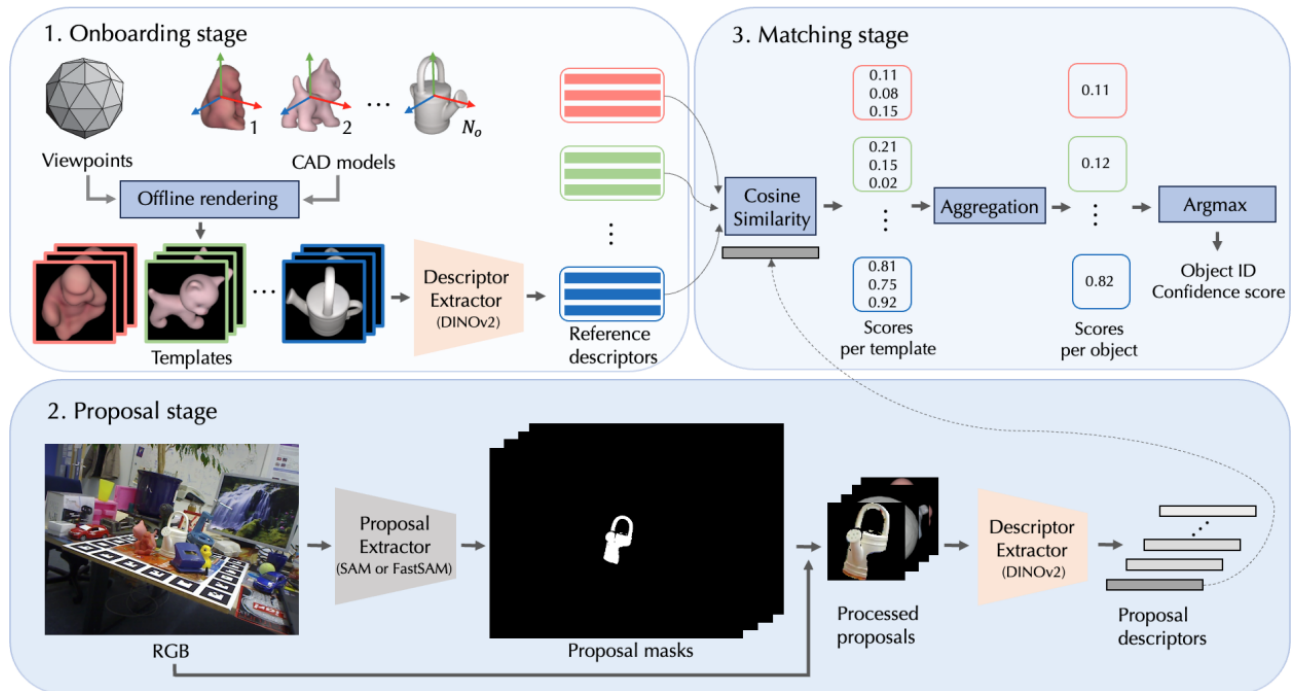
**Pose refinement.** The same render-and-compare strategy used in coarse pose estimation is also used for pose refinement. The current estimate of the object pose is used to render the image which is concatenated with the input image patch and used as input to neural network. The goal of the refiner is to predict an update to the pose that iteratively reduces the error to zero. The refiner is evaluated several times in a sequence.

**Training.** Both coarse and refiner models are trained for a specific dataset of objects. Therefore, costly (about 4 hours on 40 GPUs) retraining is needed for each new object, which is impractical for real-world industrial applications. This limitation motivates us to design unseen object pose estimator as described next.

## 2.2 Objects unseen during training

Unseen object pose estimation is based on two works: the state-of-the-art unseen object detection module called CNOS [10] and the Agimus co-developed pose estimation method MegaPose [8]. Note that the database of objects of interest is known only at runtime but not at the training time.

**Object detection.** For detecting the bounding boxes and object labels without training, we used CNOS: CAD-based Novel Object Segmentation module. The method uses SAM [6] or FastSAM [13] as a proposal mechanism that predicts a set of segmentation masks. These masks are combined with the input image to segment-out the background, *i.e.* only the object of interest is visible for each proposal. Dino features are extracted from the proposal and the



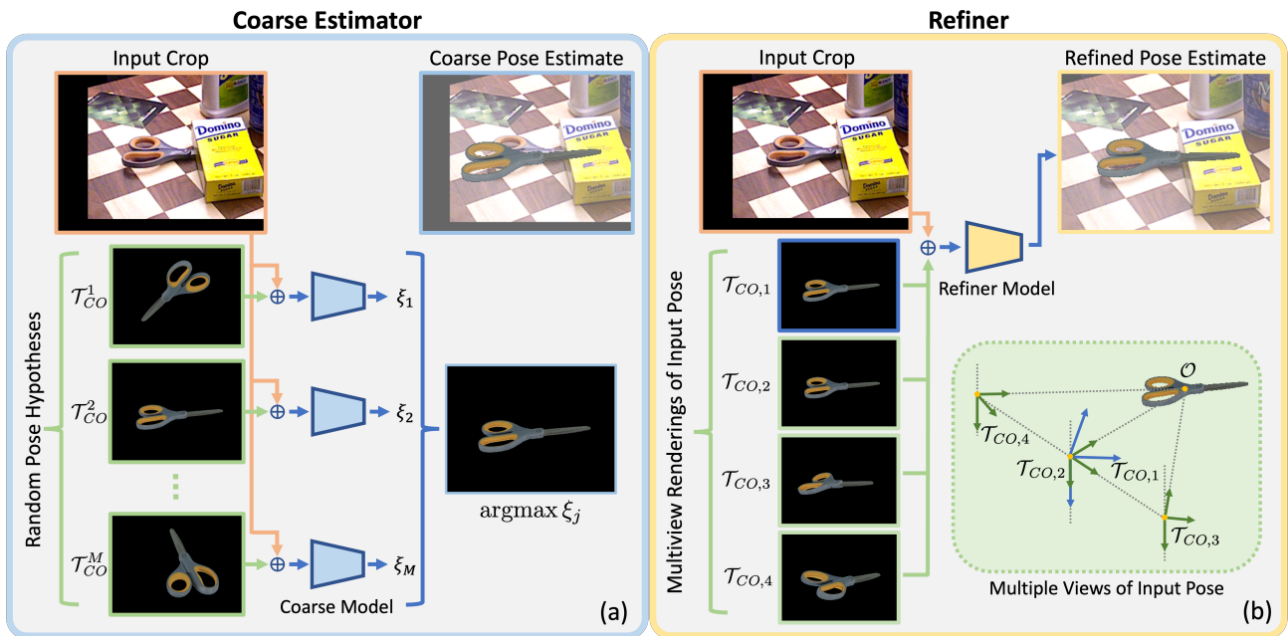
**Figure 1: CNOS overview.** Given CAD models of the target objects, the objects are onboarded by (i) rendering a set of templates showing the models from different viewpoints, and (ii) describing the templates by the DINO features. At inference time, segmentation proposals are generated from the input RGB image using SAM or FastSAM, and the proposals are matched against the templates by comparing their DINO features. The image is from [10].

closest feature is selected from a database that is computed on a synthetically pre-rendered images from the database of objects. An overview of the CNOS approach is shown in Fig. 1. Note that at the time of writing this report, CNOS is the only method competing in BOP Challenge 2023, in the unseen-object detection category.

**Coarse pose estimation.** The cropped input image and the object label are used to run MegaPose, which is, similarly to CosyPose, divided into the coarse pose estimator and the refiner. Contrary to CosyPose, the coarse pose estimation is framed as a classification problem. The input to the classification problem is a rendering of the object concatenated with the input crop. The neural network is used to score similarity between these two images, and the one with the best score is selected for refinement. The coarse estimator is shown in Fig. 2.

**Pose refinement.** The pose refinement follows the render-and-compare strategy of CosyPose, however, with three more rendered viewpoints. The refiner predicts an update of the 6D pose and is run iteratively several times. The single run of the refiner is illustrated in Fig. 2.

**Training.** For training, both coarse and refiner models require RGB(D) images with ground-truth 6D object pose annotations, along with 3D models for these objects. To allow MegaPose to generalize to novel objects, a large dataset containing diverse objects is required. Therefore, MegaPose is trained purely on synthetic data generated using BlenderProc [3]. A dataset of 2 million images was used for training using a combination of ShapeNet [2]



**Figure 2: MegaPose overview.**  $\oplus$  denotes concatenation. (a) Coarse Estimator: Given a cropped input image the coarse module renders the object in multiple input poses. The coarse network then classifies which rendered image best matches the observed image. (b) Refiner: Given an initial pose estimate the refiner renders the objects at the estimated pose (blue axes) along with 3 additional viewpoints (green axes) defined such that the camera z-axis intersects the anchor point. The refiner network consumes the concatenation of the observed and rendered images and predicts an updated pose estimate. The image is from [8].



	Date (UTC)	Method	Test image	AR <sub>Core</sub>	AR <sub>LM-O</sub>	AR <sub>T-LESS</sub>	AR <sub>TUD-L</sub>	AR <sub>IC-BIN</sub>	AR <sub>ITODD</sub>	AR <sub>HB</sub>	AR <sub>YCB-V</sub>	Time (s)
1	2023-09-28	<a href="#">GenFlow-MultiHypo16</a>	RGB-D	0.674	0.635	0.521	0.862	0.534	0.554	0.779	0.833	34.578
2	2023-09-25	<a href="#">GenFlow-MultiHypo</a>	RGB-D	0.662	0.622	0.509	0.849	0.524	0.544	0.770	0.818	21.457
3	2023-09-27	<a href="#">Megapose-CNOS_fastSAM+Multihyp_Te...</a>	RGB-D	0.628	0.626	0.487	0.851	0.467	0.468	0.730	0.764	141.965
4	2023-09-27	<a href="#">Megapose-CNOS_fastSAM+Multihyp_Te...</a>	RGB-D	0.623	0.620	0.485	0.846	0.462	0.460	0.725	0.764	116.564
5	2023-09-28	<a href="#">SAM6D-CNOSmask</a>	RGB-D	0.616	0.648	0.483	0.794	0.504	0.351	0.727	0.804	3.872
6	2023-09-26	<a href="#">PoZe (CNOS)</a>	RGB-D	0.616	0.644	0.494	0.924	0.409	0.516	0.712	0.611	159.425
7	2023-09-27	<a href="#">ZeroPose-Multi-Hypo-Refinement-De...</a>	RGB-D	0.570	0.538	0.400	0.835	0.392	0.521	0.653	0.653	16.168
8	2023-09-26	<a href="#">GenFlow-MultiHypo-RGB</a>	RGB	0.570	0.563	0.523	0.684	0.453	0.395	0.739	0.633	20.890
9	2023-09-26	<a href="#">Megapose-CNOS_fastSAM+MultiHyp-10</a>	RGB	0.549	0.560	0.508	0.687	0.419	0.346	0.706	0.620	53.878
10	2023-09-26	<a href="#">Megapose-CNOS_fastSAM+MultiHyp</a>	RGB	0.547	0.560	0.507	0.684	0.414	0.338	0.704	0.621	47.386
11	2023-09-27	<a href="#">ZeroPose-Multi-Hypo-Refinement</a>	RGB-D	0.534	0.493	0.342	0.790	0.396	0.465	0.629	0.623	18.971
12	2023-09-04	<a href="#">MegaPose-CNOS_fastSAM</a>	RGB	0.509	0.499	0.477	0.653	0.367	0.315	0.654	0.601	31.724
13	2023-08-22	<a href="#">ZeroPose-One-Hypo</a>	RGB-D	0.348	0.272	0.156	0.536	0.307	0.362	0.462	0.341	9.756
14	2023-09-26	<a href="#">GenFlow-coarse</a>	RGB	0.235	0.250	0.215	0.300	0.168	0.154	0.283	0.277	3.839

**Figure 3: BOP Challenge 2023: 6D localization of unseen objects.** Our submissions are labeled *Megapose-CNOS\_fastSAM+Multihyp\_Te...*, *CNOS\_fastSAM+MegaPose MultiHyp* and *CNOS\_fastSAM+MegaPose*.

and Google-Scanned-Objects [4]. By training on a large diverse dataset, the MegaPose generalizes well to unseen objects even without fine-tuning.

**HappyPose at BOP Challenge.** We implemented the pose estimation of unseen objects in the HappyPose software <https://github.com/agimus-project/happypose>. We used this implementation to compete in the BOP Challenge 2023. Our results in the category of 6D localization of unseen objects are shown in Fig. 3. In this category, we were awarded the **BOP Challenge 2023 Award for The Best Open-Source Method** as shown in Fig. 4, and ranked 3rd among all competitors.

### 3 Continuous Tracking

Object pose estimation from previous sections are accurate but slow to be used in closed-loop control. On a recent GPU, the evaluation of CosyPose takes approximately 0.25 second while the evaluation of MegaPose takes approximately 30 seconds. To achieve higher frequency we combine CosyPose pose estimation (here called localizer) with an image-based 6D pose tracker, called ICG [11].

Although being fast, this local tracker is unable to detect the appearance of a new object or that the object is no longer visible. We combined the benefits of the object 6D pose estimator and object 6D pose tracker into a single perception module, that is accurate and fast at the same time.

We present below a brief overview of the proposed perception module and highlight of the results. We refer the reader to the attached paper for more information and discussion of the



## BOP Challenge 2023 Award

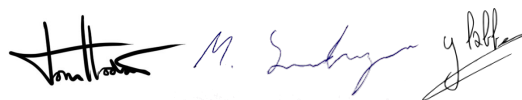
**The Best Open-Source Method**

Task 4: Model-based 6D localization of unseen objects

**MegaPose**

Elliot Maître, Mederic Fourmy, Lucas Manuelli, Yann Labbé

8th International Workshop on Recovering 6D Object Pose, ICCV 2023

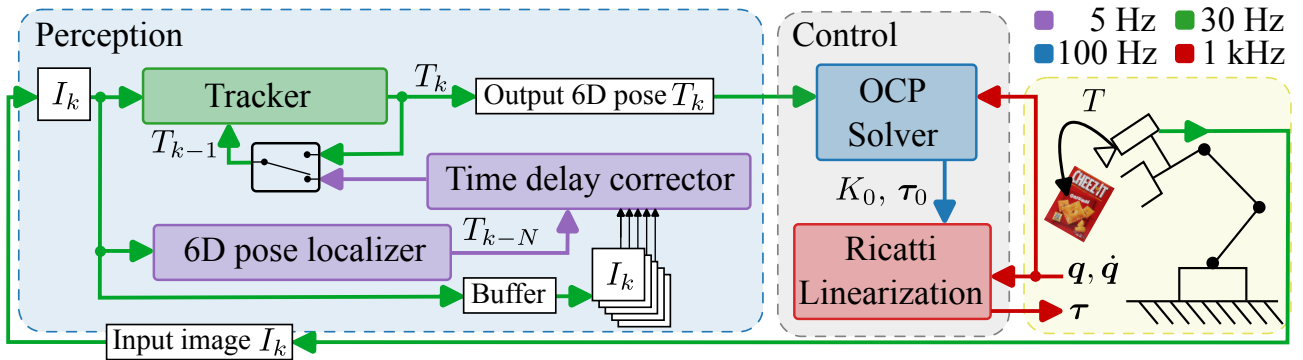
Three handwritten signatures in blue ink. From left to right: the first signature is "Elliot Maître", the second is "M. Fourmy", and the third is "Y. Labbé".

**Figure 4: BOP Challenge 2023 Award in the category of 6D localization of unseen objects.**

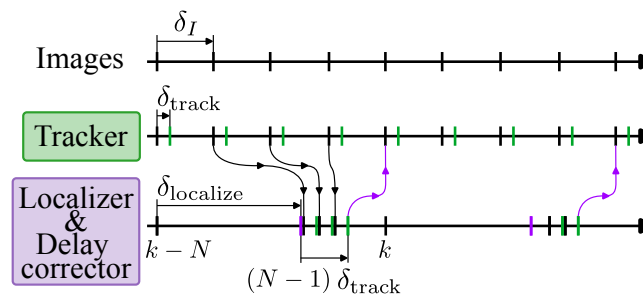
results.

### 3.1 Object localization and tracking (OLT)

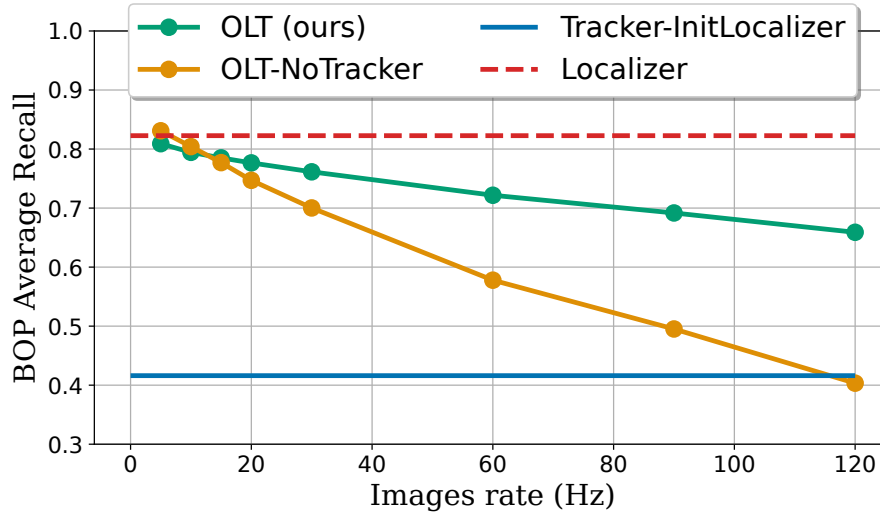
We combine the CosyPose localizer and the ICG tracker into a single perception module that computes fast feedback at the frequency of the tracker while running the localizer in parallel for object (re-)discovery and more accurate pose estimation. Our architecture, shown in the perception plate of Fig. 5, runs the tracker on the current image  $I_k$  with the initial pose selected either from (i) the previous iteration of the tracker, *i.e.*  $T_{k-1}$  or (ii) the separate



**Figure 5: Overview of the fast continuous perception module.** The objective of the feedback control is to track the 6D pose of an object seen by a camera, as illustrated on the right by a robot and red cheez-it box. To achieve that, we designed a perception module that runs a fast local **Tracker** on an input image  $I_k$  with the initial pose  $T_{k-1}$  selected either from the previous run of the tracker or from the **6D pose localizer & Time delay corrector** modules, if that information is available. The **6D pose localizer** is slow and the objective of the **Time delay corrector** is to *catch-up* in time by quickly tracking through images stored in the buffer while the 6D pose localizer was computing. The output of the tracker, the pose  $T_k$ , can be used to control robot, *e.g.* by running an Optimal Control Problem solver with Ricatti Linearization.



**Figure 6: Perception module timeline.** The first row illustrates the stream of images with typical delay between images  $\delta_I$  being 33 ms. The second row illustrates the delay caused by the **tracker** module, denoted by  $\delta_{\text{track}}$  that corresponds to a few milliseconds and therefore output poses (green ticks) are produced at the frequency of the input image stream. The tracker needs initial pose that is taken either from previous run of the tracker or from the **6D pose localizer & time delay corrector** modules if possible as indicated by purple arrows. The 6D pose localizer runs with typical  $\delta_{\text{localize}}$  being a few hundreds of milliseconds and is followed by the tracker applied  $N - 1$  times on the buffered images (green ticks in the third row).



**Figure 7: Evaluation of the proposed perception module** Average recall (higher is better) of BOP metrics [12] measuring the accuracy of 6D pose estimation of different implementations of object localization and tracking. The comparison was run on the YCBV video dataset replayed at different frequencies on the same hardware. The *Localizer* runs CosyPose [7] on every image, it is slow and cannot be used in closed-loop control but it is accurate (high recall). *Tracker-InitLocalizer* runs ICG [11] after initialized with the Localizer on the first frame only, it is fast but the recall is low. *OLT (ours)* combines the benefits of both, as it is as fast as ICG tracker while being as accurate as the localizer for low frequencies of the input image stream. The performance drop of *OLT-NoTracker* illustrates the importance of ICG tracker in the loop by replacing the tracker with identity mapping (*i.e.* use the last estimate of the localizer).

process that localizes the object if that information has already been computed by the *time delay corrector* for the previous image  $I_{k-1}$ , *i.e.* end of the image buffer. The main tracker is initialized once the first frame to enter the system has been processed by the localizer and time delay corrector.

### 3.2 Time delay corrector

In a parallel process, a single instance of the localizer is run all the time the resources are available. Let us assume that the localizer started processing input image  $I_{k-N}$  at time  $k - N$ . It takes some time to get the output of the localizer during which new images arrive and are stacked inside a buffer. Once the pose  $T_{k-N}$  is computed by the localizer, a second instance of the tracker is run on all images inside the buffer, while providing the final pose computed at the time  $k - 1$  to the main tracker process. The timeline of the perception module is illustrated in Fig. 6.

### 3.3 Evaluation

We quantitatively evaluate the new perception module on the YCBV dataset [1] using the 6D object pose (BOP benchmark) evaluation metrics [12]. The YCBV dataset consists of several videos of a moving camera showing a subset of 22 objects available in the dataset. Every frame of the video is annotated with the ground truth poses for all objects visible in the scene. We use the YCBV dataset because of the availability of the real objects for real-world experiments and of the pre-trained models for the CosyPose [7] object pose estimator. We use the BOP toolkit [12] to compute standard 6D pose error metrics to assess the quality of pose estimates. The evaluation procedure feeds the images of the input video sequence in order and with a given frequency to the perception module. The output poses are compared with the ground truth by evaluating *BOP Average Recall* score defined in [12]. The results of the evaluation procedure are shown in Fig. 7.

## 4 Conclusion

In this report, we present an approach to estimate the 6D pose of the object from the input image. We first present accurate learning-based methods to estimate the 6D pose of objects for two categories: (i) objects seen during training and (ii) objects unseen at the training time. Both methods have been shown to be accurate but slow to be used in closed-loop control. Therefore, we combined the learning-based pose estimation method with a local 6D pose tracker that is fast to evaluate. Our solution combines the accuracy of the learning-based pose-estimation method with the speed of a local tracker.

The proposed perception module can be used directly in the closed-loop robot. We made an initial step towards this objective in the attached paper. In the context of the AGIMUS project, the proposed perception module will be used in **T4.3: Feedback from multimodal perception (haptic+vision)** for robust vision-based robot control.

## Reference List

- [1] Berk Calli et al. “Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols”. In: *arXiv preprint arXiv:1502.03143* (2015).
- [2] Angel X Chang et al. “Shapenet: An information-rich 3d model repository”. In: *arXiv preprint arXiv:1512.03012* (2015).
- [3] Maximilian Denninger et al. “Blenderproc”. In: *arXiv preprint arXiv:1911.01911* (2019).
- [4] Laura Downs et al. “Google scanned objects: A high-quality dataset of 3d scanned household items”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 2553–2560.

- [5] Kaiming He et al. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.
- [6] Alexander Kirillov et al. “Segment anything”. In: *arXiv preprint arXiv:2304.02643* (2023).
- [7] Yann Labbé et al. “Cosypose: Consistent multi-view multi-object 6d pose estimation”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII* 16. Springer. 2020, pp. 574–591.
- [8] Yann Labbé et al. “Megapose: 6d pose estimation of novel objects via render & compare”. In: *arXiv preprint arXiv:2212.06870* (2022).
- [9] Fourmy Mederic et al. “Visually guided model predictive robot control via 6D object pose detection and tracking”. In: *Submitted to ICRA 2024-International Conference on Robotics and Automation*.
- [10] Van Nguyen Nguyen et al. “CNOS: A Strong Baseline for CAD-based Novel Object Segmentation”. In: *arXiv preprint arXiv:2307.11067* (2023).
- [11] Manuel Stoiber, Martin Sundermeyer, and Rudolph Triebel. “Iterative corresponding geometry: Fusing region and depth for highly efficient 3d tracking of textureless objects”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 6855–6865.
- [12] Martin Sundermeyer et al. “Bop challenge 2022 on detection, segmentation and pose estimation of specific rigid objects”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 2784–2793.
- [13] Xu Zhao et al. “Fast Segment Anything”. In: *arXiv preprint arXiv:2306.12156* (2023).



## A MegaPose paper [8]

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

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

**Abstract:** We introduce MegaPose, a method to estimate the 6D pose of novel objects, that is, objects unseen during training. At inference time, the method only assumes knowledge of (i) a region of interest displaying the object in the image and (ii) a CAD model of the observed object. The contributions of this work are threefold. First, we present a 6D pose refiner based on a render & compare strategy which can be applied to novel objects. The shape and coordinate system of the novel object are provided as inputs to the network by rendering multiple synthetic views of the object’s CAD model. Second, we introduce a novel approach for coarse pose estimation which leverages a network trained to classify whether the pose error between a synthetic rendering and an observed image of the same object can be corrected by the refiner. Third, we introduce a large scale synthetic dataset of photorealistic images of thousands of objects with diverse visual and shape properties, and show that this diversity is crucial to obtain good generalization performance on novel objects. We train our approach on this large synthetic dataset and apply it *without retraining* to hundreds of novel objects in real images from several pose estimation benchmarks. Our approach achieves state-of-the-art performance on the ModelNet and YCB-Video datasets. An extensive evaluation on the 7 core datasets of the BOP challenge demonstrates that our approach achieves performance competitive with existing approaches that require access to the target objects during training. Code, dataset and trained models are available on the project page [1].

## 1 Introduction

Accurate 6D object pose estimation is essential for many robotic and augmented reality applications. Current state-of-the-art methods are learning-based [2, 3, 4, 5, 6] and require 3D models of the objects of interest at both training and test time. These methods require hours (or days) to generate synthetic data for each object and train the pose estimation model. They thus cannot be used in the context of robotic applications where the objects are only known during inference (e.g. CAD models are provided by a manufacturer or reconstructed [7]), and where rapid deployment to novel scenes and objects is key.

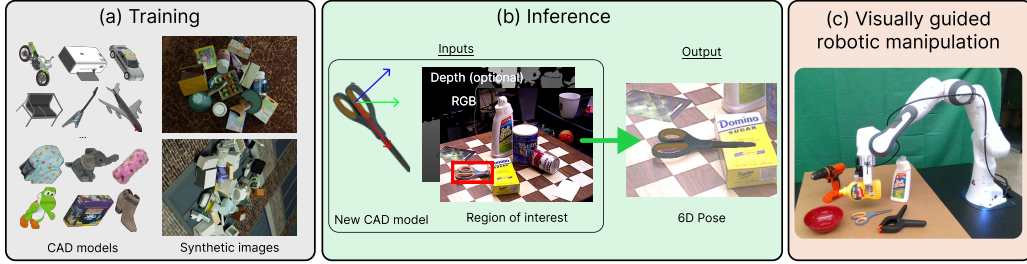
The goal of this work is to estimate the 6D pose of novel objects, *i.e.*, objects that are only available at *inference time* and are not known in advance during training. This problem presents the challenge of generalizing to the large variability in shape, texture, lighting conditions, and severe occlusions that can be encountered in real-world applications. Some prior works [8, 9, 10, 11, 12, 13, 14] have considered category-level pose estimation to partially address the challenge of novel objects by developing methods that can generalize to novel object instances of a known class (e.g. mugs or shoes). These methods however do not generalize to object instances outside of training categories. Other methods aim at generalizing to any novel instances regardless of their category [15, 16, 17, 18, 19, 20, 21, 22]. These works present important technical limitations. They rely on non-learning based components for generating pose hypotheses [21] (e.g. PPF [23]), for pose refinement [17] (e.g.

<sup>1</sup>Inria Paris and Département d’informatique de l’ENS, École normale supérieure, CNRS, PSL Research University, 75005 Paris, France.

<sup>3</sup>LIGM, École des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-vallée, France.

<sup>5</sup>Czech Institute of Informatics, Robotics and Cybernetics at the Czech Technical University in Prague.

<sup>†</sup>Work partially done while the author was an intern at NVIDIA.



**Figure 1: MegaPose** is a 6D pose estimation approach (a) that is trained on millions of synthetic scenes with thousands of different objects and (b) can be applied *without re-training* to estimate the pose of any novel object, given a CAD model and a region of interest displaying the object. It can thus be used to rapidly deploy visually guided robotic manipulation systems in novel scenes containing novel objects (c).

PnP [24] and ICP [25, 26]), for computing photometric errors in pixel space [15], or for estimating the object depth [18, 16] (e.g. using only the size of a 2D detection [27]). These components however inherently cannot benefit from being trained on large amount of data to gain robustness with respect to noise, occlusions, or object variability. Learning-based methods also have the potential to improve as the quality and size of the datasets improve.

Pipelines for 6D pose estimation of known (not novel) objects that consist of multiple learned stages [4, 5] have shown excellent performance on several benchmarks [2] with various illumination conditions, textureless objects, cluttered scenes and high levels of occlusions. We take inspiration from [4, 5] which split the problem into three parts: (i) 2D object detection, (ii) coarse pose estimation, and (iii) iterative refinement via render & compare. We aim at extending this approach to novel objects unseen at training time. The detection of novel objects has been addressed by prior works [17, 28, 29, 30] and is outside the scope of this paper. In this work, we focus on the coarse and refinement networks for 6D pose estimation. Extending the paradigm from [4] presents three major challenges. First, the pose of an object depends heavily on both its visual appearance and choice of coordinate system (defined in the CAD model of the object). In existing refinement networks based on render & compare [20, 4], this information is encoded in the network weights during training, leading to poor generalization results when tested on novel objects. Second, direct regression methods for coarse pose estimation are trained with specific losses for symmetric objects [4], requiring that object symmetries be known in advance. Finally, the diversity of shape and visual properties of the objects that can be encountered in real-world applications is immense. Generalizing to novel objects requires robustness to properties such as object symmetries, variability of object shape, and object textures (or absence of).

**Contributions.** We address these challenges and propose a method for estimating the pose of any novel object in a single RGB or RGB-D image, as illustrated in Figure 1. First, we propose a novel approach for 6D pose refinement based on render & compare which enables generalization to novel objects. The shape and coordinate system of the novel object are provided as inputs to the network by rendering multiple synthetic views of the object’s CAD model. Second, we propose a novel method for coarse pose estimation which does not require knowledge of the object symmetries during training. The coarse pose estimation is formulated as a classification problem where we compare renderings of random pose hypotheses with an observed image, and predict whether the pose can be corrected by the refiner. Finally, we leverage the availability of large-scale 3D model datasets to generate a highly diverse synthetic dataset consisting of 2 million photorealistic [31] images depicting over 20K models in physically plausible configurations. The code, dataset and trained models are available on the project page [1].

We show that our novel-object pose estimation method trained on our large-scale synthetic dataset achieves state-of-the-art performance on ModelNet [32, 20]. We also perform an extensive evaluation of the approach on hundreds of novel objects from all 7 core datasets of the BOP challenge [2] and demonstrate that our approach achieves performance competitive with existing approaches that require access to the target objects during training.

## 2 Related work

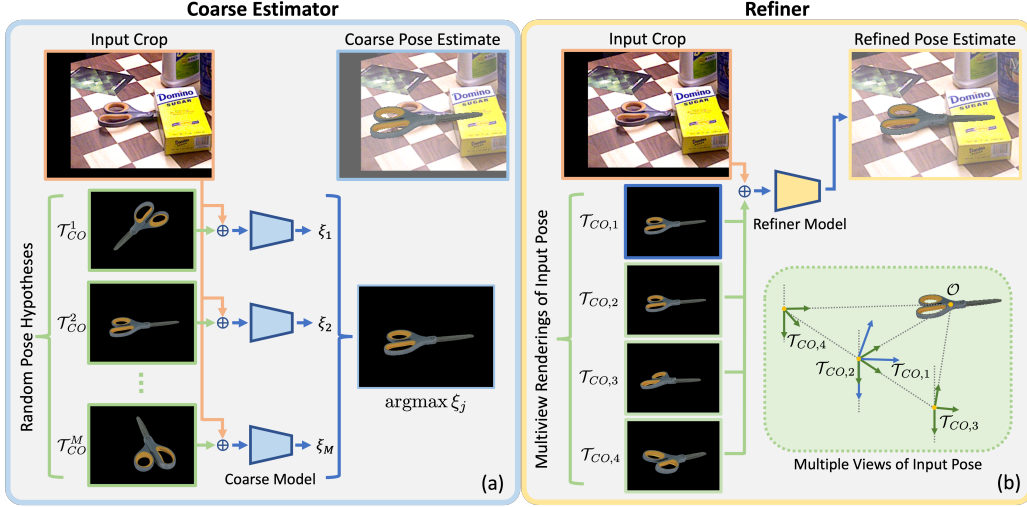
In this section, we first review the literature on 6D pose estimation of known rigid objects. We then focus on the practical scenario similar to ours where the objects are not known prior to training.

**6D pose estimation of known objects.** Estimating the 6D pose of rigid objects is a fundamental computer vision problem [33, 34, 35] that was first addressed using correspondences established with locally invariant features [35, 36, 37, 38, 23] or template matching [39, 40]. These have been replaced by learning-based methods with convolutional neural networks that directly regress sets of sparse [41, 42, 27, 43, 44, 45, 46] or dense [47, 48, 49, 50, 3, 44] features. All these approaches use non-learning stages relying on PnP+Ransac [51, 24] to recover the pose from correspondences in RGB images, or variants of the iterative closest point algorithm, ICP [25, 26], when depth is available. The best performing methods rely on trainable refinement networks [52, 20, 4, 20, 5] based on render & compare [53, 54, 55, 20]. These methods render a single image of the object, which is not sufficient to provide complete information on the shape and coordinate system of a 3D model to the network. This information is thus encoded in the networks weights when training, which leads to poor generalization when tested on novel objects unseen at training. Our approach renders multiple views of an object to provide this 3D information, making the trained network independent of these object-specific properties.

**6D pose estimation of novel objects.** Other works consider a practical scenario where the objects are not known in advance. Category-level 6D pose estimation is a popular problem [8, 9, 10, 11, 12, 13, 14] in which CAD models of test objects are not known, but the objects are assumed to belong to a known category. These methods rely on object properties that are common within categories (*e.g.* handle of a mug) to define and estimate the object pose, and thus cannot generalize to novel categories. Our method requires the 3D model of the novel object instance to be known during inference, but does not rely on any category-level information. Other works address a scenario similar to ours. [56, 19, 57, 18, 16, 30] only estimate the 3D orientation of novel objects by comparing rendered pose hypotheses with the observed image using features extracted by a network. They rely on handcrafted [18, 16] or learning-based DeepIM [19] refiners to recover accurate 6D poses. We instead propose a method that estimates the full 6D pose of the object and show our refinement network significantly outperforms DeepIM [20] when tested on novel object instances. The closest works to ours are OSOP [17] and ZePHyR [21]. OSOP focuses on the coarse estimation by explicitly predicting 2D-2D correspondences between a single rendered view of the object and the observed image, and solves for the pose using PnP or Kabsch [25] which makes inference slower and less robust compared to directly predicting refinement transforms with a network as done in our solution. ZePHyR [21] strongly relies on the depth modality, whereas our approach can also be used in RGB-only images. Finally, [15, 58, 22, 59] investigate using a set of real reference views of the novel object instead of using a CAD model. These approaches have only reported results on datasets with limited or no occlusions. Our use of a deep render & compare network trained on a large-scale synthetic dataset displaying highly occluded object instances enables us to handle highly cluttered scenes with high occlusions like in the LineMOD Occlusion, HomebrewedDB or T-LESS datasets.

## 3 Method

In this section we present our framework for pose estimation of novel objects. Our goal is to detect the pose  $\mathcal{T}_{CO}$  (the pose of object frame  $O$  expressed in camera frame  $C$  composed of 3D rotation and 3D translation) of a novel object given an input RGB (or RGBD) image,  $I_o$ , and a 3D model of the object. Similar to DeepIM [20] and CosyPose [4], our method consists of three components (1) object detection, (2) coarse pose estimation and (3) pose refinement. Compared to these works, our proposed method enables generalization to novel objects not seen during training, requiring novel approaches for the coarse model, the refiner and the training data. Our approach can accept either RGB or RGBD inputs, if depth is available the RGB and D images are concatenated before being passed into the network. Detection of novel-objects in an image is an interesting problem that has been addressed in prior work [28, 60, 17, 22, 30] but lies outside the scope of this paper. Thus for our experiments we assume access to an object detector, but emphasize that our method can be coupled with any object detector, including zero-shot methods such as those in [28, 60].



**Figure 2:**  $\oplus$  denotes concatenation. **(a) Coarse Estimator:** Given a cropped input image the coarse module renders the object in multiple input poses  $\{\mathcal{T}_{CO}^j\}$ . The coarse network then classifies which rendered image best matches the observed image. **(b) Refiner:** Given an initial pose estimate  $\mathcal{T}_{CO}^k$  the refiner renders the objects at the estimated pose  $\mathcal{T}_{CO,1} := \mathcal{T}_{CO}^k$  (blue axes) along with 3 additional viewpoints  $\{\mathcal{T}_{CO,i}\}_{i=2}^4$  (green axes) defined such that the camera  $z$ -axis intersects the anchor point  $\mathcal{O}$ . The refiner network consumes the concatenation of the observed and rendered images and predicts an updated pose estimate  $\mathcal{T}_{CO}^{k+1}$ .

### 3.1 Technical Approach

**Coarse pose estimation.** Given an object detection, shown in Figure 1(b), the goal of the coarse pose estimator is to provide an initial pose estimate  $\mathcal{T}_{CO,coarse}$  which is sufficiently accurate that it can then be further improved by the refiner. In order to generalize to novel-objects we propose a novel classification based approach that compares observed and rendered images of the object in a variety of poses and selects the rendered image whose object pose best matches the observed object pose.

Figure 2(a) gives an overview of the coarse model. At inference time the network consumes the observed image  $I_o$  along with rendered images  $\{I_r(\mathcal{T}_{CO}^j)\}_{j=1}^M$  of the object in many different poses  $\{\mathcal{T}_{CO}^j\}_{j=1}^M$ . For each pose  $\mathcal{T}_{CO}^j$  the model predicts a score  $(I_o, I_r(\mathcal{T}_{CO}^j)) \rightarrow \xi_j$  that classifies whether the pose hypothesis is within the basin of attraction of the refiner. The highest scoring pose  $\mathcal{T}_{CO}^{j*} = \arg\max_j \xi_j$  is used as the initial pose for the refinement step. Since we are performing classification, our method can implicitly handle object symmetries, as multiple poses can be classified as correct.

**Pose refinement model.** Given an input image and an estimated pose, the refiner predicts an updated pose estimate. Starting from a coarse initial pose estimate  $\mathcal{T}_{CO,coarse}$  we can iteratively apply the refiner to produce an improved pose estimate. Similar to [4, 20] our refiner takes as input observed  $I_o$  and rendered images  $I_r(\mathcal{T}_{CO}^k)$  and predicts an updated pose estimate  $\mathcal{T}_{CO}^{k+1}$ , see Figure 2 (b), where  $k$  refers to the  $k^{th}$  iteration of the refiner. Our pose update uses the same parameterization as DeepIM [20] and CosyPose [4] which disentangles rotation and translation prediction. Crucially this pose update  $\Delta\mathcal{T}$  depends on the choice of an *anchor point*  $\mathcal{O}$ , see Appendix for more details. In prior work [4, 20] which trains and tests on the same set of objects, the network can effectively learn the position of the anchor point  $\mathcal{O}$  for each object. However in order to generalize to novel objects we must enable the network to infer the anchor point  $\mathcal{O}$  at inference time.

In order to provide information about the anchor point to the network we always render images  $I_r(\mathcal{T}_{CO}^k)$  such that the anchor point  $\mathcal{O}$  projects to the image center. Using rendered images from multiple distinct viewpoints  $\{\mathcal{T}_{CO,i}\}_{i=1}^N$  the network can infer the location of the anchor point  $\mathcal{O}$  as the intersection point of camera rays that pass through the image center, see Figure 2(b).

Additional information about object shape and geometry can be provided to the network by rendering depth and surface normal channels in the rendered image  $I_r$ . We normalize both input depth (if

available) and rendered depth images using the currently estimated pose  $\mathcal{T}_{CO}^k$  to assist the network in generalizing across object scales, see Appendix for more details.

**Network architecture.** Both the coarse and refiner networks consists of a ResNet-34 backbone followed by spatial average pooling. The coarse model has a single fully-connected layer that consumes the backbone feature and outputs a classification logit. The refiner network has a single fully-connected layer that consumes the backbone feature and outputs 9 values that specify the translation and rotation for the pose update.

### 3.2 Training Procedure

**Training data.** For training, both the coarse and refiner models require RGB(-D)<sup>1</sup> images with ground-truth 6D object pose annotations, along with 3D models for these objects. In order for our approach to generalize to novel-objects we require a large dataset containing diverse objects. All of our methods are trained purely on synthetic data generated using BlenderProc [31]. We generate a dataset of 2 million images using a combination of ShapeNet [61] (abbreviated as SN) and Google-Scanned-Objects (abbreviated as GSO) [7]. Similar to the BOP [62] synthetic data, we randomly sampled objects from our dataset and dropped them on a plane using a physics simulator. Materials, background textures, lighting and camera positions are randomized. Example images can be seen in Figure 1(a) and in the Appendix. Some of our ablations also use the synthetic training datasets provided by the BOP challenge [62]. We add data augmentation similar to CosyPose [4] to the RGB images which was shown to be a key to successful sim-to-real transfer. We also apply data augmentation to the depth images as explained in the appendix.

**Refiner model.** The refiner model is trained similarly to [4]. Given an image with an object  $\mathcal{M}$  at ground-truth pose  $\mathcal{T}_{CO}^*$  we generate a perturbed pose  $\mathcal{T}_{CO}'$  by applying a random translation and rotation to  $\mathcal{T}_{CO}^*$ . 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. The network is trained to predict the relative transformation between the initial and target pose. Following [4, 20] we use a loss that disentangles the prediction of depth,  $x$ - $y$  translation, and rotation. See the appendix for more details.

**Coarse model.** Given an input image  $I_o$  of an object  $\mathcal{M}$  and a pose  $\mathcal{T}_{CO}'$  the coarse model is trained to classify whether pose  $\mathcal{T}_{CO}'$  is within the basin of attraction of the refiner. In other words, if the refiner were started with the initial pose estimate  $\mathcal{T}_{CO}'$  would it be able to estimate the ground-truth pose via iterative refinement? Given a ground-truth pose-annotation  $\mathcal{T}_{CO}^*$  we randomly sample poses  $\mathcal{T}_{CO}'$  by adding random translation and rotation to  $\mathcal{T}_{CO}^*$ . The positives are sampled from the same distribution used to generate the perturbed poses the refiner network is trained to correct (see above), and other poses sufficiently distinct to this one (see the appendix for more details) are marked as negatives. The model is then trained with binary cross entropy loss.

## 4 Experiments

We evaluate our method for 6D pose estimation of novel objects using the seven challenging datasets of the BOP [2, 62] 6D pose estimation benchmark, and the ModelNet [20] dataset. The dataset and the standard 6D pose estimation metrics we use are detailed in Section 4.1. In all our experiments, the objects are considered novel, i.e. they are only available during inference on a new image and they are not used during training. In Section 4.2, we evaluate the performance of our approach composed of coarse and refinement networks. Notably, we show that (i) our method is competitive with others that require the object models to be known in advance, and (iii) our refiner outperforms current state-of-the-art on the ModelNet and YCB-V datasets. Section 4.3 validates our technical contributions and shows the crucial importance of the training data in the success of our method. Finally, we discuss the limitations in Section 4.4.

### 4.1 Dataset and metrics

We consider the seven core datasets of the BOP challenge [62, 2]: LineMod Occlusion (LM-O) [63], T-LESS [64], TUD-L [62], IC-BIN [65], ITODD [66], HomebrewedDB (HB) [67] and

<sup>1</sup>Our method can consume either RGB or RGB-D images depending on the input modalities that are available.



Pose Initialization		Pose Refinement			BOP Datasets							
Method	Novel objects	Method	Novel objects	RGB-D Input	LM-O	T-LESS	TUD-L	IC-BIN	ITODD	HB	YCB-V	Mean
1 CosyPose [4]	✗	CosyPose	✗		63.3	64.0	68.5	58.3	21.6	65.6	57.4	57.0
2 SurfEmb [3]	✗	BFGS	✗		66.3	73.5	71.5	58.8	41.3	79.1	64.7	65.0
3 SurfEmb [3]	✗	BFGS+ICP	✓	✓	75.8	82.8	85.4	65.6	49.8	86.7	80.6	75.2
4 OSOP [17]	✓	Multi-Hyp.	✓		31.2	-	-	-	-	49.2	33.2	-
5 OSOP [17]	✓	MH+ICP	✓	✓	48.2	-	-	-	-	60.5	57.2	-
6 (PPF, Sift) + Zephyr [21]	✓	-	✓	✓	<b>59.8</b>	-	-	-	-	-	51.6	-
7 (PPF, Sift) + Our coarse	✓	Our refiner	✓	✓	57.0	-	-	-	-	-	62.3	-
8 CosyPose [4]	✗	-			53.6	52.0	57.6	53.0	13.1	33.5	33.3	42.3
9 CosyPose [4]	✗	Ours	✓		65.5	72.0	70.1	57.3	28.4	67.0	56.8	59.6
10 CosyPose [4]	✗	Ours	✓	✓	71.2	63.8	85.0	55.1	39.9	73.2	69.2	66.0
11 Ours	✓	-			18.7	19.7	20.5	15.3	8.00	18.6	13.9	16.2
12 Ours	✓	Ours	✓		53.7	<b>62.2</b>	58.4	<b>43.6</b>	30.1	72.9	60.4	54.5
13 Ours	✓	Ours	✓	✓	58.3	54.3	<b>71.2</b>	37.1	<b>40.4</b>	<b>75.7</b>	<b>63.3</b>	<b>57.2</b>

**Table 1: Results on the BOP challenge datasets.** We report the AR score on each of the 7 datasets considered in the BOP challenge and the mean score across datasets. With the exception of Zephyr (row 11), all approaches are trained purely on synthetic data. For each column, we denote the best over result in *italics* and the best novel-object pose estimation method in **bold**.

YCB-Video (YCB-V) [47]. These datasets exhibit 132 different objects in cluttered scenes with occlusions. These objects present many factors of variation: textured or untextured, symmetric or asymmetric, household or industrial (e.g. watcher pitcher, stapler, bowls, multi-socket plug adaptor) which makes them representative of objects that are typically encountered in robotic scenarios. The ModelNet dataset depicts individual instances of objects from 7 classes of the ModelNet [32] dataset (bathtub, bookshelf, guitar, range hood, sofa, tv stand and wardrobe). We use initial poses provided by adding noise to the ground truth, similar to previous works [20, 16, 15]. The focus is on refining these initial poses. We follow the evaluation protocol of [2] for BOP datasets, and of DeepIM [20] for ModelNet.

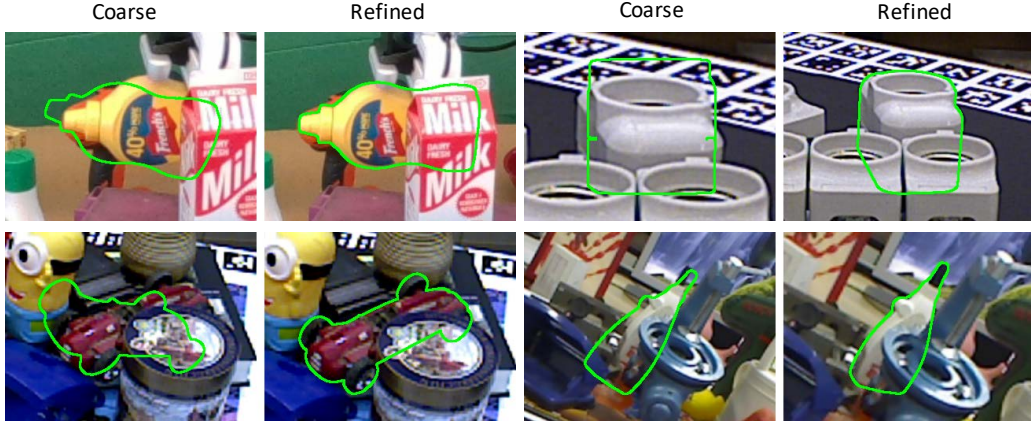
## 4.2 6D pose estimation of novel objects

**Performance of coarse+refiner.** Table 1 reports results of our novel-object pose estimation method on the BOP datasets. We first use the detections and pose hypotheses provided by a combination of PPF and SIFT, similar to the state-of-the-art method Zephyr [21]. For each object detection, these algorithms provide multiple pose hypotheses. We find the best hypothesis using the score of our coarse network, and apply 5 iterations of our refiner. Results are reported in row 7. On YCB-V, our method achieves a +10.7 AR score improvement compared to Zephyr (row 6). Averaged across the YCB-V and LM-O datasets, the AR score of our approach is 59.7 compared to 55.7 for Zephyr (row 6). Next, we provide a complete set of results using the detections from Mask-RCNN networks. Please note that since detection of novel objects is outside the scope of this paper, we use the networks trained on the synthetic PBR data of the target objects [2] which are publicly available for each dataset. We report the results of our coarse estimation strategy (Table 1, row 11), and after running the refiner network, on RGB (row 12) or RGB-D (row 13) images. We observe that (i) our refinement network significantly improves the coarse estimates (+41.0 mean AR score for our RGBD refiner) and (ii) the performance of both models is competitive with the learning-based refiner of CosyPose [4] (row 1) while not requiring to be trained on the test objects. The recent SurfEmb [3] performs better than our approach, but heavily relies on the knowledge of the objects for training and cannot generalize to novel objects.

**Performance of the refiner.** We now focus on the evaluation of our refiner which can be used to refine arbitrary initial poses. Our refiner is the only learning-based approach in Table 1 which can be applied to novel objects. In rows 9 and 10, we apply our refiner to the coarse estimates of CosyPose [4] (row 11). Again, we observe that our refiner significantly improves the accuracy of these initial pose estimates (+23.7 in average for the RGB-D model). Notably, the RGB-only method (row 9) performs better than the CosyPose refiner (row 1) on average, while not having seen the BOP objects during training. This is thanks to our large-scale training on thousands of various objects, while CosyPose is only trained on tens of objects for each dataset.

Method	RGB-D	Average Recall		
		(5°, 5cm)	ADD (0.1d)	Proj2D (5px)
DeepIM [20]	✓	64.3	83.6	73.3
Multi-Path [16]		84.8	90.1	81.6
LatentFusion[15]	✓	85.5	94.3	94.7
Ours		88.6	90.5	88.9
Ours	✓	<b>97.6</b>	<b>98.9</b>	<b>97.5</b>

**Table 2: Evaluation of the refiner on the ModelNet [20] dataset.** The mean average recall is computed over the seven classes of the dataset.



**Figure 3: Qualitative results.** For each pair of images, the left image is a visualization of our coarse estimate, and the right image is after applying 5 iterations of our refiner. None of these objects from the YCBV, LMO, HB, or T-LESS datasets were used for training our approach. Please notice the high accuracy of MegaPose despite (i) severe occlusions and (ii) the varying properties of the novel objects (e.g. the texture-less industrial plug in the top-right example, textured mustard bottle in the top-left).

One iteration of our refiner takes approximately 50 milliseconds on a RTX 2080 GPU, making it suitable for use in an online tracking application. Five iterations of our refiner are also 5 times faster than the object-specific refiner of SurfEmb [3] which takes around 1 second per image crop. Finally, we evaluate our refiner on ModelNet and compare it with the state-of-the-art methods MP-AAE [16] and LatentFusion [15]. For this experiment, we remove the ShapeNet categories that overlap with the test ones in ModelNet from our training set in order to provide a fair comparison on novel instances and novel categories similar to [15, 16, 20]. Results reported in Table 2 show that our refiner significantly outperforms existing approaches across all metrics.

### 4.3 Ablations

In this section we perform ablations of our approach to validate our main contributions. Additional ablations are in the appendix. For these ablations, we consider the RGB-only refiner and re-train several models with different configurations of hyper-parameters and training data.

**Encoding the anchor point and object shape.** As discussed in Section 3.1 the refiner must have information about the anchor point  $\mathcal{O}$  in order to generalize to novel objects. We accomplish this by using 4 rendered views pointing towards the anchor point, see Figure 2(b). Table 3(a) shows the performance of the refiner increases as we increase the number of views from 1 to 4, validating our design choice. Multiple views may also help the network to understand the object’s appearance from alternate viewpoints, thus potentially helping the refiner to overcome large initial pose errors. We also validate our choice to provide a normal map of the object to the network. This information can help the network use subtle object appearance variations that are only visible under different illumination like the details on the cross of a guitar.

**Number of training objects.** We now show the crucial role of the training data. We report in Table 3 (b) the results for our refiner trained on an increasing number of CAD models. The performance steadily increases with the number of objects, which validates that training on a large number of object

				Training objects	Num. objects	BOP5	ModelNet ADD(0.1d)
Rendered views	Rendered normals	BOP5	ModelNet ADD(0.1d)	GSO+ShapeNet	10 + 100	47.9	28.7
1	✓	52.0	83.3	GSO+ShapeNet	100 + 1000	49.3	80.3
2	✓	59.0	90.4	GSO+ShapeNet	250 + 2000	56.9	82.0
4	✓	<b>61.7</b>	<b>96.1</b>	GSO+ShapeNet	500 + 10000	59.3	89.3
4		59.1	83.1	GSO+ShapeNet	1000 + 20000	61.7	<b>96.1</b>
(a)				GSO	1000	62.2	95.7
				BOP	132	<b>62.6</b>	93.4
				(b)			

**Table 3: Ablation study.** We study (a) using multiple rendered object views and normal maps as input to our RGB-only refiner model and (b) training the refiner on different variations of our large-scale synthetic dataset. Average recall is reported on BOP5 (mean of LM-O, T-LESS, TUD-L, IC-BIN and YCB-V) and ModelNet.

models is important to generalize to novel ones. These results also suggest that the performance of our approach *could* be improved as more datasets of high-quality CAD models like GSO [7] become available.

**Variety in the training objects.** Next, we restrict the training to different sets of objects. We observe in the bottom of Table 3(b) that models from the GoogleScannedObjects are more important to the performance of the method on the BOP dataset compared to using both ShapeNet and GSO. We hypothesize this is due to the presence of high-quality textured objects in the GSO dataset. Finally, we train our model on the 132 objects of the BOP dataset. When testing on the same BOP objects, the performance benefits from knowing these objects during training is small compared to using our GSO+ShapeNet or GSO dataset.

#### 4.4 Limitations

While MegaPose shows promising results in robot experiments (**please see the supplementary video**) and 6D pose estimation benchmarks, there is still room for improvement. We illustrate the failure modes of our approach in the supplementary material. The most common failure mode is due to inaccurate initial pose estimates from the coarse model. The refiner model is trained to correct poses that are within a constrained range but can fail if the initial error is too large. There exist multiple potential approaches to alleviate this problem. We can increase the number of pose hypotheses  $M$  at the expense of increased inference time, improve the accuracy of the coarse model, and increase the basin of attraction for refinement model. Another limitation is the runtime of our coarse model. We use  $M = 520$  pose hypotheses per object which takes around 2.5 seconds to be rendered and evaluated by our coarse model. In a tracking scenario however, the coarse model is run just once at the initial frame and the object can be tracked using the refiner which runs at 20Hz. Additionally, our refiner could also be coupled with alternate coarse estimation approaches such as [17, 18] to achieve improved runtime performance.

## 5 Conclusion

We propose MegaPose, a method for 6D pose estimation of novel objects. Megapose can estimate the 6D pose of novel objects given a CAD model of the object available only at test time. We quantitatively evaluated MegaPose on hundreds of different objects depicted in cluttered scenes, and performed ablation studies to validate our network design choices and highlight the importance of the training data. We release our models and large-scale synthetic dataset to stimulate the development of novel methods that are practical to use in the context of robotic manipulation where rapid deployment to new scenes with new objects is crucial. While this work focuses on the coarse estimation and fine refinement of an object pose, detecting any unknown object given only a CAD model is still a difficult problem that remains to be solved for having a complete framework for detection and pose estimation of novel objects. Future work will address zero-shot object detection using our large-scale synthetic dataset.

## Acknowledgements

This work was partially supported by the HPC resources from GENCI-IDRIS (Grant 011011181R2), the European Regional Development Fund under the project IMPACT (reg. no. CZ.02.1.01/0.0/0.0/15 003/0000468), EU Horizon Europe Programme under the project AGIMUS (No. 101070165), Louis Vuitton ENS Chair on Artificial Intelligence, and the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute).

## References

- [1] Project page: <https://megapose6d.github.io>.
- [2] T. Hodaň, M. Sundermeyer, B. Drost, Y. Labbé, E. Brachmann, F. Michel, C. Rother, and J. Matas. BOP Challenge 2020 on 6D object localization. In *ECCVW*, 2020.
- [3] R. L. Haugaard and A. G. Buch. Surfemb: Dense and continuous correspondence distributions for object pose estimation with learnt surface embeddings. In *CVPR*, 2022.
- [4] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic. CosyPose: Consistent multi-view multi-object 6D pose estimation. In *ECCV*, 2020.
- [5] L. Lipson, Z. Teed, A. Goyal, and J. Deng. Coupled iterative refinement for 6D multi-object pose estimation. In *CVPR*, 2022.
- [6] G. Wang, F. Manhardt, F. Tombari, and X. Ji. GDR-Net: Geometry-guided direct regression network for monocular 6D object pose estimation. In *CVPR*, 2021.
- [7] 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.
- [8] Y. Lin, J. Tremblay, S. Tyree, P. A. Vela, and S. Birchfield. Single-stage keypoint-based category-level object pose estimation from an RGB image. In *ICRA*, 2022.
- [9] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *CVPR*, 2019.
- [10] X. Chen, Z. Dong, J. Song, A. Geiger, and O. Hilliges. Category level object pose estimation via neural analysis-by-synthesis. In *ECCV*, 2020.
- [11] F. Li, I. Shugurov, B. Busam, M. Li, S. Yang, and S. Ilic. Polarmesh: A star-convex 3d shape approximation for object pose estimation. *IEEE Robotics and Automation Letters*, 7(2):4416–4423, 2022.
- [12] F. Manhardt, G. Wang, B. Busam, M. Nickel, S. Meier, L. Minciullo, X. Ji, and N. Navab. CPS++: Improving class-level 6D pose and shape estimation from monocular images with self-supervised learning. *arXiv preprint arXiv:2003.05848*, 2020.
- [13] L. Manuelli, W. Gao, P. Florence, and R. Tedrake. KPAM: Keypoint affordances for category-level robotic manipulation. In *ISRR*, 2019.
- [14] B. Wen, W. Lian, K. Bekris, and S. Schaal. Catgrasp: Learning category-level task-relevant grasping in clutter from simulation. *arXiv preprint arXiv:2109.09163*, 2021.
- [15] K. Park, A. Mousavian, Y. Xiang, and D. Fox. LatentFusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. In *CVPR*, 2020.
- [16] M. Sundermeyer, M. Durner, E. Y. Puang, Z.-C. Marton, N. Vaskevicius, K. O. Arras, and R. Triebel. Multi-path learning for object pose estimation across domains. In *CVPR*, 2020.
- [17] I. Shugurov, F. Li, B. Busam, and S. Ilic. OSOP: A multi-stage one shot object pose estimation framework. In *CVPR*, 2022.
- [18] V. N. Nguyen, Y. Hu, Y. Xiao, M. Salzmann, and V. Lepetit. Templates for 3D object pose estimation revisited: Generalization to new objects and robustness to occlusions. In *CVPR*, 2022.
- [19] Y. Xiao, X. Qiu, P.-A. Langlois, M. Aubry, and R. Marlet. Pose from shape: Deep pose estimation for arbitrary 3D objects. In *BMVC*, 2019.

- [20] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. DeepIM: Deep iterative matching for 6D pose estimation. In *ECCV*, 2018.
- [21] B. Okorn, Q. Gu, M. Hebert, and D. Held. ZePHYR: Zero-shot pose hypothesis rating. In *ICRA*, 2021.
- [22] Y. Liu, Y. Wen, S. Peng, C. Lin, X. Long, T. Komura, and W. Wang. Gen6D: Generalizable model-free 6-DoF object pose estimation from RGB images. In *arXiv:2204.10776*, 2022.
- [23] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *CVPR*, 2010.
- [24] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An accurate  $O(n)$  solution to the PnP problem. *IJCV*, 2009.
- [25] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992.
- [26] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152, 1994.
- [27] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. In *ICCV*, 2017.
- [28] A. Osokin, D. Sumin, and V. Lomakin. Os2d: One-stage one-shot object detection by matching anchor features. In *ECCV*. Springer, 2020.
- [29] J.-P. Mercier, M. Garon, P. Giguere, and J.-F. Lalonde. Deep template-based object instance detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1507–1516, 2021.
- [30] Y. Xiao and R. Marlet. Few-shot object detection and viewpoint estimation for objects in the wild. In *European Conference on Computer Vision (ECCV)*, 2020.
- [31] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam. Blenderproc. *arXiv preprint arXiv:1911.01911*, 2019.
- [32] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.
- [33] L. G. Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, 1963.
- [34] D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artif. Intell.*, 1987.
- [35] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.
- [36] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *ECCV*, 2006.
- [37] A. Collet and S. S. Srinivasa. Efficient multi-view object recognition and full pose estimation. In *ICRA*, 2010.
- [38] A. Collet, M. Martinez, and S. S. Srinivasa. The MOPED framework: Object recognition and pose estimation for manipulation. *The International Journal of Robotics Research*, 2011.
- [39] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *ICCV*, 2011.
- [40] F. Jurie and M. Dhome. A simple and efficient template matching algorithm. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 544–549. IEEE, 2001.
- [41] M. Rad and V. Lepetit. BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth. In *CVPR*, 2017.
- [42] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. In *Conference on Robot Learning (CoRL)*, 2018.
- [43] B. Tekin, S. N. Sinha, and P. Fua. Real-time seamless single shot 6d object pose prediction. In *CVPR*, 2018.

- [44] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. Pynet: Pixel-wise voting network for 6dof pose estimation. In *CVPR*, 2019.
- [45] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis. 6-dof object pose from semantic keypoints. In *ICRA*, 2017.
- [46] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann. Segmentation-driven 6D object pose estimation. In *CVPR*, 2019.
- [47] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. In *RSS*, 2018.
- [48] K. Park, T. Patten, and M. Vincze. Pix2Pose: Pixel-wise coordinate regression of objects for 6D pose estimation. In *ICCV*, 2019.
- [49] C. Song, J. Song, and Q. Huang. Hybridpose: 6D object pose estimation under hybrid representations. In *CVPR*, 2020.
- [50] S. Zakharov, I. Shugurov, and S. Ilic. Dpod: 6d pose object detector and refiner. In *CVPR*, 2019.
- [51] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
- [52] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3343–3352, 2019.
- [53] K. Pauwels and D. Kragic. Simtrack: A simulation-based framework for scalable real-time object pose detection and tracking. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1300–1307. IEEE, 2015.
- [54] M. Oberweger, P. Wohlhart, and V. Lepetit. Training a feedback loop for hand pose estimation. In *ICCV*, 2015.
- [55] F. Manhardt, W. Kehl, N. Navab, and F. Tombari. Deep model-based 6d pose refinement in rgb. In *ECCV*, 2018.
- [56] M. Aubry, D. Maturana, A. A. Efros, B. Russell, and J. Sivic. Seeing 3D chairs: Exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *CVPR*, 2014.
- [57] Y. Xiao, Y. Du, and R. Marlet. PoseContrast: Class-agnostic object viewpoint estimation in the wild with pose-aware contrastive learning. In *3DV*, 2021.
- [58] Y. He, Y. Wang, H. Fan, J. Sun, and Q. Chen. FS6D: Few-shot 6D pose estimation of novel objects. In *CVPR*, 2022.
- [59] J. Sun, Z. Wang, S. Zhang, X. He, H. Zhao, G. Zhang, and X. Zhou. OnePose: One-shot object pose estimation without CAD models. In *CVPR*, 2022.
- [60] E. Olson. AprilTag: A robust and flexible visual fiducial system. In *ICRA*, 2011.
- [61] 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.
- [62] T. Hodan, F. Michel, E. Brachmann, W. Kehl, A. GlentBuch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, et al. Bop: Benchmark for 6d object pose estimation. In *ECCV*, 2018.
- [63] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision (ACCV)*, 2012.
- [64] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis. T-LESS: An RGB-D dataset for 6D pose estimation of Texture-Less objects. In *Winter Conference on Applications of Computer Vision (WACV)*, 2017.
- [65] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim. Recovering 6d object pose and predicting next-best-view in the crowd. In *CVPR*, 2016.



- [66] B. Drost, M. Ulrich, P. Bergmann, P. Hartinger, and C. Steger. Introducing mvtec itodd-a dataset for 3d object recognition in industry. In *ICCV Workshops*, 2017.
- [67] R. Kaskman, S. Zakharov, I. Shugurov, and S. Ilic. Homebreweddb: Rgb-d dataset for 6d pose estimation of 3d objects. In *ICCV Workshops*, 2019.
- [68] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. In *CVPR*, 2019.
- [69] The pillow imaging library. <https://github.com/python-pillow/pillow>.
- [70] 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.
- [71] 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.
- [72] 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.

## Appendix

This appendix is organized as follows. In section A, we provide the equations of the pose update predicted by the refiner network, and show it depends on the anchor point. In section B, we give details on the loss used to train the refiner network. Section C explains the normalization strategy we apply to the observed and rendered depth images of the RGB-D refiner. Section D details the pose hypotheses used during training and inference of the coarse model. In section E, we provide examples of training images and give details on the data augmentation and training hardware. In section F, 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 G, we illustrate qualitatively that our approach is robust to illumination condition variations. Section H illustrates the main failure modes of our approach. Finally, section I 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.

## A Pose update and anchor point

**Pose update.** We use the same pose update as DeepIM [20] and CosyPose [4]. 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 anchor point  $\mathcal{O}$  and the object frame  $O$ ):

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

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

$$z_{\mathcal{O}}^{k+1} = v_z z_{\mathcal{O}}^k, \quad (3)$$

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

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 [68] by orthogonalizing the basis defined by the two predicted rotation vectors  $e_1, e_2$  similar to [4]. 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):

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

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

$$v_z^1 = \frac{z_{O^1}^{k+1}}{z_{O^1}^k} \quad (7)$$

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

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

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

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

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

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

From eq. (12), we have  $R^1 (R^2)^T = Id$ . 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  $v_z^1 - v_z^2 = \frac{z_{12}(z_{O^1}^{k+1} - z_{O^1}^k)}{z_{O^1}^k(z_{O^1}^k + z_{12})}$  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.

## B Refiner loss

Our refiner network is trained using the same loss as in CosyPose [4], 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  $O$ :

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

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  $\mathcal{T}_{CO}^k$ , the predictions of the neural network  $[v_x, v_y, v_z]$  and  $R$ , and outputs the updated pose:

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

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

$$\mathcal{L} = \sum_{k=1}^K D_O(F(\mathcal{T}_{CO}^k, [v_x, v_y, v_z^*], R^*), \mathcal{T}_{CO}^*) \quad (15)$$

$$+ D_O(F(\mathcal{T}_{CO}^k, [v_x^*, v_y^*, v_z], R^*), \mathcal{T}_{CO}^*) \quad (16)$$

$$+ D_O(F(\mathcal{T}_{CO}^k, [v_x^*, v_y^*, v_z^*], R), \mathcal{T}_{CO}^*), \quad (17)$$

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.

## C 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  $z_O^k + 1$ , where  $z_O^k$  is the depth of the anchor point on the object in the input pose at iteration  $k$ :

$$D \leftarrow \text{clip}(D^k, 0, z_O^k + 1), \quad (18)$$

and (ii) centering of the depth values:

$$D \leftarrow \frac{D}{z_O^k} - 1. \quad (19)$$

## D Pose hypotheses in the coarse model

**Training hypotheses.** Given the ground truth object pose  $\mathcal{T}_{CO}^*$ , 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  $2z'_O$ , where  $z'_O$  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  $90^\circ$ ,  $180^\circ$  and  $270^\circ$  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  $\mathcal{T}_{CO}^*$  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  $z_{Op}^{guess} = 1\text{m}$  and use this initial value to estimate the coordinates  $x_{Op}$  and  $y_{Op}$  of the anchor point in the camera frame:

$$x_{Op}^{guess} = u_{det,x} \frac{z_{Op}^{guess}}{f_x} \quad (20)$$

$$y_{Op}^{guess} = u_{det,y} \frac{z_{Op}^{guess}}{f_y}, \quad (21)$$

where  $f_x$  and  $f_y$  are the (known) focal lengths of the camera. We then update the depth estimate  $z_{Op}^{guess}$  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:

$$z_{Op} = z_{Op}^{guess} \frac{1}{2} \left( f_x \frac{\Delta u_{guess,x}}{\Delta u_{det,x}} + f_y \frac{\Delta u_{guess,y}}{\Delta u_{det,y}} \right) \quad (22)$$

and use this new depth to compute  $x_{Op}$  and  $y_{Op}$  using equations (20) and (21) that were used to define  $x_{Op}^{guess}$  and  $y_{Op}^{guess}$ . The rotation  $R^p$  and 3-vector  $[x_{Op}, y_{Op}, z_{Op}]$  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.

## E Training details

**Training images.** We generate 2 million photorealistic images using BlenderProc [31] as explained in Section 3.2 of the main paper. Randomly sampled images from the training set are shown in Figure 4.

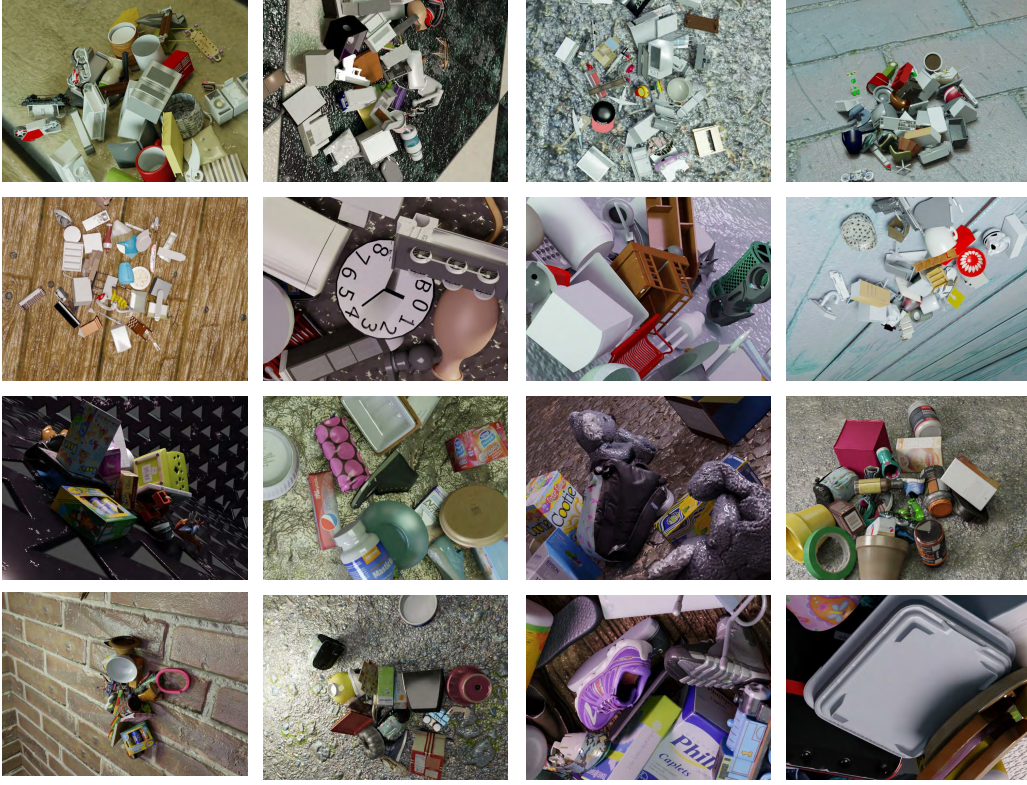
**Data augmentation.** We apply data augmentation to the synthetic images during training. We use the same data augmentation as CosyPose [4] for the RGB images. It includes Gaussian blur, contrast, brightness, colors and sharpness filters from the Pillow library [69]. For the depth images, we take inspiration from the augmentations used in [70, 71, 72]. 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.

## F 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 [21]. 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 4. 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.





**Figure 4: Example of training images.** Randomly sampled images from our large-scale synthetic dataset generated with BlenderProc [31]. CAD models from the ShapeNet [61] and GoogleScannedObjects [7] datasets are used.

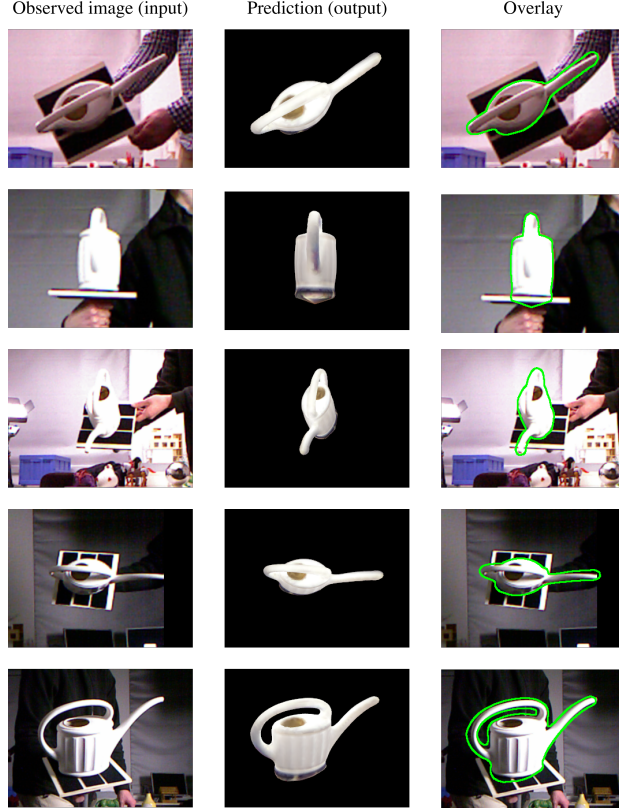
Pose hypotheses	YCB-V	LM-O
PPF	52.7	34.4
PPF+Zephyr [21]	59.8	45.8
PPF+Our coarse	<b>61.6</b>	<b>52.1</b>

**Table 4: Performance of the coarse model.** We compare the performance of our coarse network with Zephyr [21].

**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 [4] 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



**Figure 5:** 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.

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 [4], 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.

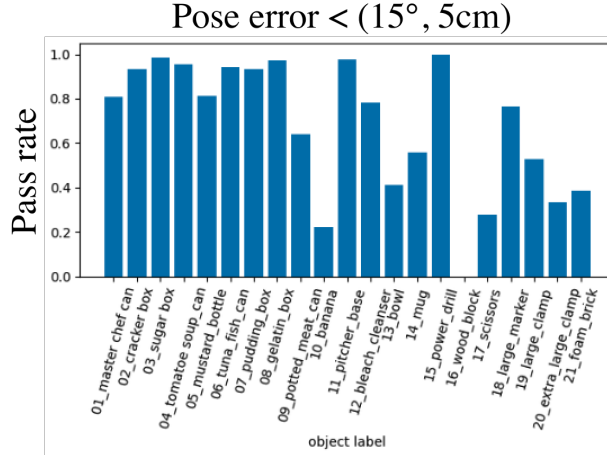
## G Robustness to illumination conditions

In Figure 5, 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.

## H 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  $15^\circ$  in rotation and 5cm in translation. Results are reported in Figure 6.

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 7. We



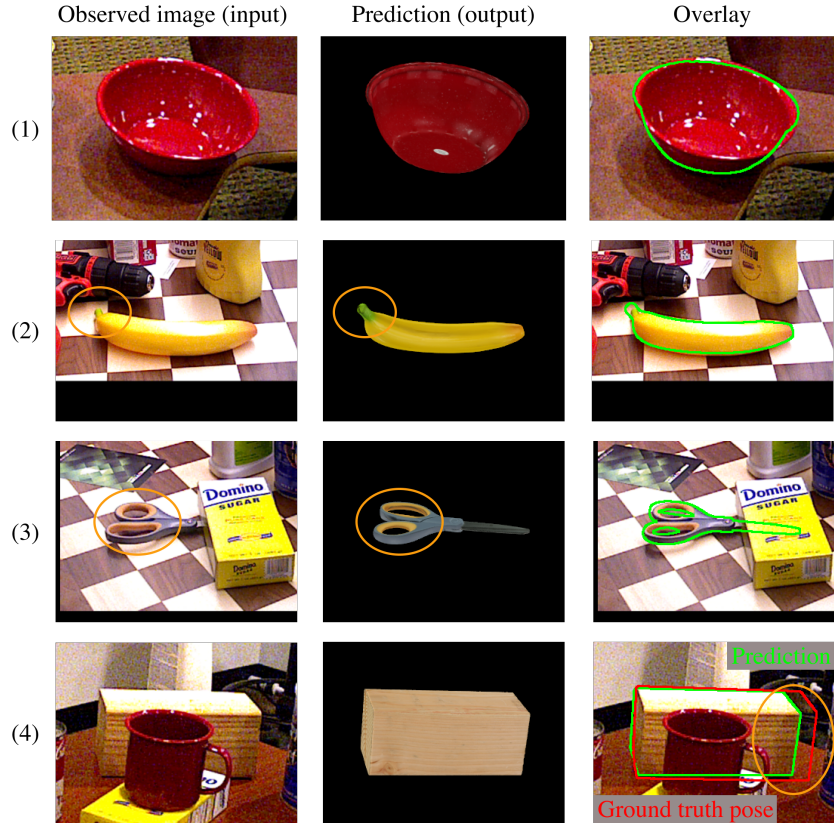
**Figure 6:** Per-object analysis on the YCB-V dataset. For each object, we report the percentage of estimates for which the error between our pose prediction and the ground truth is within 5 centimeters in translation and 15 degrees in rotation.

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.

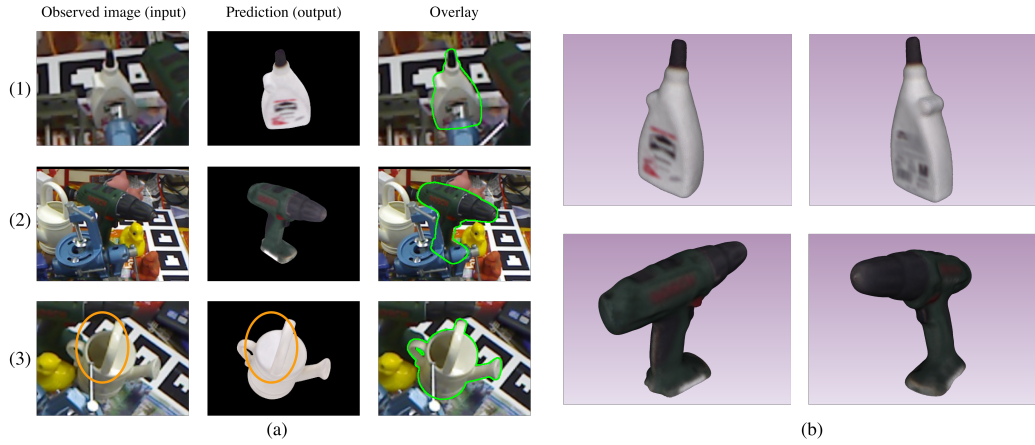
## I 3D model quality

Our approach can be applied even if the 3D model of the object does not exactly matches the real object. In figure 8, 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.





**Figure 7:** 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 8:** 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.

## **B Visual MPC paper [9]**

# Visually Guided Model Predictive Robot Control via 6D Object Pose Localization and Tracking

Mederic Fourmy♣ Vojtech Priban♣ Jan Kristof Behrens♣ Nicolas Mansard◇ Josef Sivic♣ Vladimir Petrik♣

**Abstract**—The objective of this work is to enable manipulation tasks with respect to the 6D pose of a dynamically moving object using a camera mounted on a robot. Examples include maintaining a constant relative 6D pose of the robot arm with respect to the object, grasping the dynamically moving object, or co-manipulating the object together with a human. Fast and accurate 6D pose estimation is crucial to achieve smooth and stable robot control in such situations. The contributions of this work are three fold. First, we propose a new visual perception module that asynchronously combines accurate learning-based 6D object pose localizer and a high-rate model-based 6D pose tracker. The outcome is a low-latency accurate and temporally consistent 6D object pose estimation from the input video stream at up to 120 Hz. Second, we develop a visually guided robot arm controller that combines the new visual perception module with a torque-based model predictive control algorithm. Asynchronous combination of the visual and robot proprioception signals at their corresponding frequencies results in stable and robust 6D object pose guided robot arm control. Third, we experimentally validate the proposed approach on a challenging 6D pose estimation benchmark and demonstrate 6D object pose-guided control with dynamically moving objects on a real 7 DoF Franka Emika Panda robot.

## I. INTRODUCTION

Visually-guided control is at the core of many robotic applications, from path following by mobile robots [1] to visual servoing [2]. In order to achieve a stable and robust feedback loop, the perception system has to recover the estimated state both accurately and at a high rate. In the context of object manipulation, a commonly chosen state representation is the 6D pose of objects of interest, *i.e.*, the 3D translation and 3D rotation of the objects in the scene with respect to the camera coordinate frame. While some manipulation tasks can be achieved with a static scene model [3], many applications are of an inherently dynamic nature with human-robot handovers [4], [5], human-robot co-manipulation [6] or mobile manipulation [7] being the prime examples. This is challenging as it requires accurate and low-latency 6D pose estimation of the target objects in the scene. In addition, pose estimates need to be integrated with a robust and reactive controller that is capable of meeting the dynamic requirements of the application.

♣ CIIRC, Czech Technical University in Prague

◇ LAAS-CNRS, Université de Toulouse, CNRS, Toulouse

This work was partly supported by the AGIMUS project, funded by the European Union under GA no.101070165, by the European Regional Development Fund under project Robotics for Industry 4.0 (reg. no. CZ.02.1.01/0.0/0.0/15.003/0000470), and by the Czech Science Foundation (project no. GA21-31000S). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

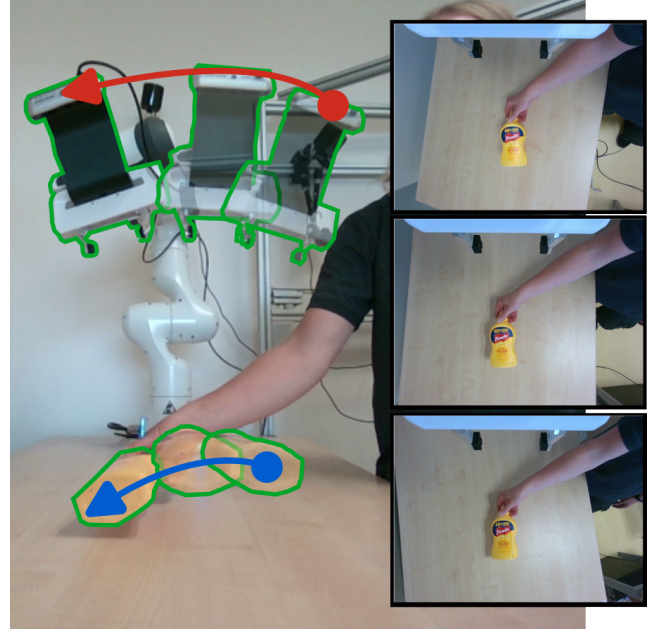


Fig. 1: **Robot arm control by 6D pose of the object.** The objective is to control the robot arm with a mounted camera (red arrow) by commanding joint torques such that the object 6D pose (blue arrow) w.r.t. the camera remains constant. This is illustrated by three frames (see insets) captured by the robot camera corresponding to the robot/object poses shown by green contours in the main image. Please note (see the insets) how the object pose remains stable while the background changes in the captured frames. **More results and experimental analysis in the companion video.**

Despite promising recent progress in object detection and 6D pose estimation [8], [9], [10], [11], [12], [13], 6D object pose estimation algorithms usually focus on accuracy rather than speed. On the other end, object 6D pose tracking methods offer fast pose updates of already detected objects, but require an initialization from the user [14]. As a result of these limitations, many real-world applications rely instead on fiducial markers [15], [16], [17], motion capture systems [6], [5] or ad-hoc detection such as color based segmentation [18].

In this paper, we propose a visual perception module that builds on (i) state-of-the-art accurate learning-based 6D object pose detector and (ii) state-of-the-art high-rate model-based 6D pose tracker to achieve object pose estimation limited only by the rate of image acquisition. Further, we develop a visually guided robot controller based on the

model predictive control (MPC) that is able to reactively incorporate perception updates to meet application targets (e.g., positioning the robot’s gripper). The *anticipatory* nature of MPC makes it particularly suitable for generating efficient motions in real time [19]. In principle, the only input data required is a 3D object model to configure the 6D pose tracker and the 6D pose estimator. Our method runs the pose detector and several instances of the object tracker in separate processes and uses the results of the pose detector to re-initialize the trackers. New images are directly fed to a tracker so that a pose estimation result is available within the tracker runtime of approximately 5 ms. To quantitatively validate our perception module, we build on the BOP challenge evaluation [9] and the YCBV dataset [20] to measure the performance of the proposed method and several baseline methods. Lastly, we demonstrate visually-guided robot arm control with hand held objects (see Fig. 1).

**Contributions.** The paper has the following three main contributions: (i) we propose a new visual perception module that asynchronously combines accurate learning-based 6D object pose localizer and a high-rate model-based 6D pose tracker; (ii) we develop a visually guided robot arm controller that leverages the new visual perception module in a torque-based model predictive control algorithm; and (iii) we experimentally validate the proposed approach on a challenging 6D pose estimation benchmark and demonstrate 6D object pose-guided control with dynamically moving objects on a real 7 DoF Franka Emika Panda robotic platform. We will make the code publicly available.

## II. RELATED WORK

**Object 6D localization.** The field of object detection and 6D pose estimation has shown impressive progress over recent years, which has been documented and fostered by the *benchmark for 6D object pose estimation* (BOP) and the associated BOP challenge [8], [9]. Most methods follow a two-step approach, first performing object detection in RGB frames [21] followed by 6D pose estimation, assuming the availability of object meshes. Learning-based techniques have dominated the field. Among the diverse approaches, render-and-compare methods [22], [10], [13], have achieved superior performance by iteratively refining the 6D pose based on predictions by a neural network. Due to their iterative nature these methods achieve superior performance but are slow to be used in real-time control. In this work, we propose to combine a slow “render and compare” 6D pose localizer with a fast 6D pose tracker. Although the proposed approach can work with an arbitrary localizer, we use pre-trained CosyPose [10] in all our experiments.

**Object 6D pose tracking.** When a good initial guess of the object 6D pose is available, it can be tracked frame to frame by fast local methods. Object pose tracking methods rely on object edges [23], extracted point features [24], [25], or depth [26]. Region-based tracking approaches propose to solve the 2D object shape segmentation and 6D pose tracking problems jointly by constructing an image-wise posterior distribution [27]. Although initial versions required

highly optimized GPU implementations to run at the camera frequency [28], [29], a sparse formulation based on contour point sampling [30] dramatically reduces the computation time, down to a few milliseconds per image [31], [14] on a single CPU. In this work, we rely on the ICG implementation [14] that features both region-based and depth modalities. The combination of the 6D pose localizer and the local 6D pose tracker that we propose in this paper combines the benefits of the both world, *i.e.* the detection capability and the accuracy of the localizer and speed of the tracker.

**Visual servoing.** Visual servoing aims at building a closed-loop controller using visual information from a camera stream to achieve a certain goal. The various methods are traditionally classified into two broad categories [2]: (i) image-based visual servoing [32], [33], which uses 2D geometric primitives (e.g., points, curves) to define control objective in an image space; and (ii) pose-based visual servoing [34], [35] which assumes the availability of the estimate of a target 6D pose. Some works propose switching between image- and pose-based servoing depending on the phase of movement [36]. Although image-based servoing allows to naturally incorporate visibility constraints in the control law, pose-based servoing is closer to applications such as object grasping [5]. We address the challenge of obtaining robust and fast estimates of 6D poses for pose-based control. These works typically implement control laws at the joint velocity level, which lack the natural impedance of torque-based control.

**Model predictive control for visual servoing.** For image-based visual servoing, MPC has shown superior performance to traditional reactive controllers [37], [38]. System dynamics in an image space is either approximated analytically [37], [38] or computed through learning-based optical flow estimation methods [39]. Image-based MPC servoing was successfully applied to control drone [40], legged platform [18], or mobile manipulator [41], [42]. Contrary to the state-of-the-art methods using image space MPC, we propose using MPC for pose-based control, where 6D pose is obtained by the proposed perception module.

## III. OBJECT POSE GUIDED MODEL PREDICTIVE CONTROL

The objective of the proposed system is to perform manipulation tasks with respect to the 6D pose of a dynamically moving object using a camera mounted on a robot. The key technical challenge in such situations addressed by our method lies in achieving an accurate 6D pose estimation without introducing a significant delay into the control loop. The estimated 6D pose is then used in combination with MPC to achieve optimal control of the robot.

The proposed method is a 1 kHz torque level MPC controller taking reference from 6D object poses obtained from the 30 Hz image stream. To achieve this real-time robot control performance, the perception and control modules run asynchronously, as shown in Fig. 2. The perception module detects objects of interest in the scene and tracks them in a fast and temporally consistent manner as described in

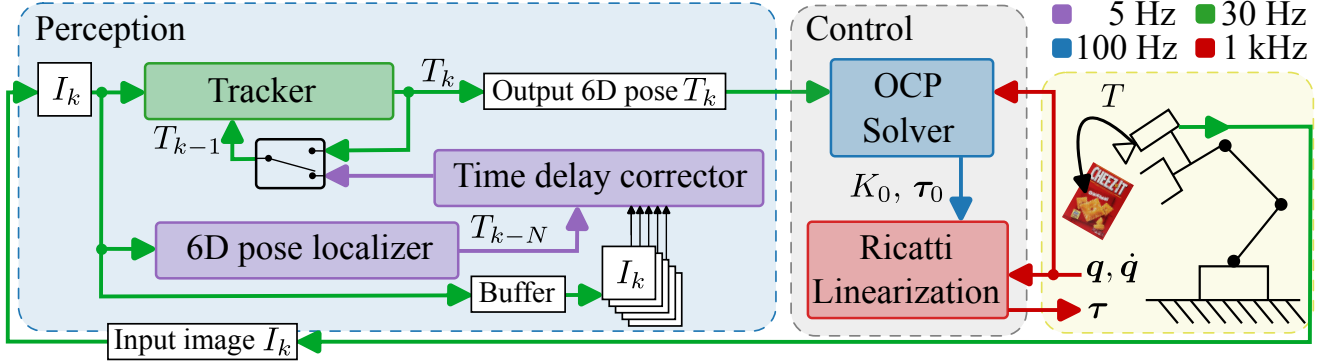


Fig. 2: **Overview of the perception-control cycle.** The objective of the feedback control is to track 6D pose of an object seen by a camera, as illustrated on the right by a robot and red cheez-it box. To achieve that, we designed a perception module that runs a fast local **Tracker** on an input image  $I_k$  with the initial pose  $T_{k-1}$  selected either from the previous run of the tracker or from the **6D pose localizer** & **Time delay corrector** modules, if that information is available. The **6D pose localizer** is slow and the objective of the **Time delay corrector** is to *catch-up* in time by quickly tracking through images stored in the buffer while the 6D pose localizer was computing. The output of the tracker, the pose  $T_k$ , is used by the **OCP solver** to compute Ricatti gains  $K_0$  and torques  $\tau_0$  that are used by the **Ricatti Linearization** module to provide fast feedback for real-time robot control. Typical processing frequencies of individual modules are 5 Hz for the 6D pose localizer and the time delay corrector, 30 Hz for the camera and tracker, 100 Hz for the OCP solver, and 1 kHz for real-time robot control.

Sec. III-A. The key innovation is handling the inherent asynchronicity of the accurate-but-slow 6D pose localizer and fast-but-local tracker via the *time delay corrector* module that operates on the buffered images in order to *catch-up* in time. The 6D poses of objects detected by the perception module are used by the Optimal Control Problem (OCP) [43] solver and a feedback controller using a Ricatti Gains linearization of the solution [44] to compute torques for the robot at 1 kHz, as described in Sec. III-B. In the following sections,  $I$  symbols represent RGB image frame, while  $T \in \mathbf{SE}(3)$  is a rigid body transformation.

#### A. Temporally consistent 6D object pose tracker

The objective of the perception module is to compute the 6D poses of objects in the scene based on the input image  $I_k$ , observed at discrete time  $k$  while introducing as little delay as possible. Although the proposed method can track an arbitrary number of objects, we describe the tracking of a single object pose  $T_k$  to simplify the notation.

**6D pose localizer.** With unlimited computational resources, the pose of an object can be estimated by the 6D pose localizer  $T_k = f_{\text{localize}}(I_k)$  that detects the object of interest in the image and estimates its 6D pose with respect to the camera coordinate frame and, as a consequence, also the robot coordinate frame, as we assume the camera is calibrated with respect to the robot. However, robust object localizers are slow, *e.g.* 0.25 s for CosyPose [10] or even 30 s for MegaPose [13] on up-to-date hardware (see Sec. IV). The long computation time makes the localizer impractical for closed-loop control.

**Tracker.** To mitigate this limitation, we combine the localizer with a fast local tracker  $T_k = f_{\text{track}}(I_k, T_{k-1})$  that computes the pose of objects from a given image  $I_k$  and initial guess of the pose  $T_{k-1}$ . Compared to the localizer, a single pass of

the tracker is fast, introducing only a few milliseconds delay into the system. However, it acts only locally, and it thus requires an initial pose that is refined based on the observed image. The tracker is not able to discover the presence of new objects, nor does it detect that the object is no longer visible by the camera when it is, for example, occluded.

**Object localization and tracking (OLT).** We combine the localizer and the tracker into a single perception module  $T_k = f_{\text{OLT}}(I_k, T_{k-1})$  that computes fast feedback at the frequency of  $f_{\text{track}}$  while running  $f_{\text{localize}}$  in parallel for object (re-)discovery and more accurate pose estimation. Our architecture, shown in the perception plate of Fig. 2, runs  $f_{\text{track}}(I_k, T_{\text{init}})$  on the current image with the initial pose  $T_{\text{init}}$  selected either from (i) the previous iteration of the tracker, *i.e.*  $T_{k-1}$  or (ii) the separate process that localizes the object if that information has already been computed by the *time delay corrector* for the previous image  $I_{k-1}$ , *i.e.* end of the image buffer. The main tracker is initialized once the first frame to enter the system has been processed by the localizer and time delay corrector.

**Time delay corrector.** In the parallel process, a single instance of the localizer is run all the time the resources are available. Let us assume that the localizer started processing input image  $I_{k-N}$  at time  $k-N$ . It takes some time to get output of  $f_{\text{localize}}$  during which new images arrive and are stacked inside a buffer. Once the pose  $T_{k-N} = f_{\text{localize}}(I_{k-N})$  is computed, a second instance of the tracker is run on all images inside the buffer, *i.e.*  $T_i = f_{\text{track}}(I_i, T_{i-1})$  iteratively for  $i \in \{k-N+1, \dots, k-1\}$  while providing the final pose computed at the time  $k-1$  to the main tracker process. The timeline of the perception module is illustrated in Fig. 3. Note that our architecture assumes that the frequency of  $f_{\text{track}}(\cdot)$  is higher than the frequency of the input image stream. Otherwise, the localizer process would



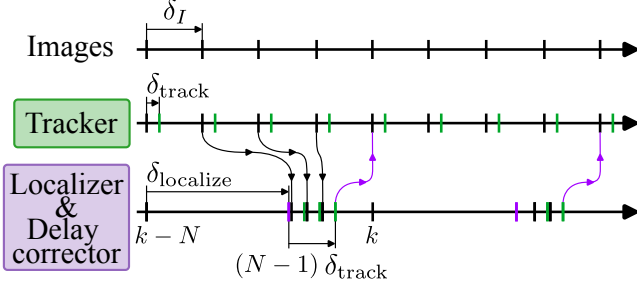


Fig. 3: **Perception module timeline.** The first row illustrates the stream of images with typical delay between images  $\delta_I$  being 33 ms. The second row illustrates the delay caused by the **tracker** module (*i.e.*  $f_{\text{track}}$ ), denoted by  $\delta_{\text{track}}$  that corresponds to a few milliseconds and therefore output poses (green ticks) are produced at the frequency of the input image stream. The tracker needs initial pose that is taken either from previous run of the tracker or from the **6D pose localizer & time delay corrector** modules if possible as indicated by purple arrows. The 6D pose localizer runs  $f_{\text{localize}}$  (with typical  $\delta_{\text{localize}}$  being a few hundreds of milliseconds) followed by  $f_{\text{track}}$  applied  $N-1$  times on the buffered images (green ticks in the third row).

never *catch-up* with the main process. With the state-of-the-art tracker [14], this feedback can be easily calculated for image frequencies up to 120 Hz. However, the higher the input image frequency, the longer it takes to inject information from the localizer process. This affects the tracking accuracy as we analyze in Sec. IV.

### B. 6D pose-based visual servoing using MPC

We design a controller that brings the camera attached to the robot end-effector to a user-defined relative pose w.r.t. the object pose  $T_k$  obtained from the object tracker. As a practical application, one may choose a reference pose from a set of predefined grasp poses for a given object. The challenge of the control lies in a real-time requirement of the robot to receive torque commands at 1 kHz. To address this challenge, we build on [44] and split the control into solving the optimal control problem at 100 Hz and computing 1 kHz feedback through Ricatti linearization, but here incorporating the 6D object poses as a guiding reference in the problem formulation.

**OCP solver.** The control module's main objective is to follow the object given the latest object pose estimates  $T_k$  provided by the perception and the current robot state  $\mathbf{x} = (\mathbf{q}^\top \quad \dot{\mathbf{q}}^\top)^\top$ , with  $\mathbf{q}$  and  $\dot{\mathbf{q}}$  being the measured joint angles and velocities, respectively. We control the manipulator at the torque level (*i.e.*  $\mathbf{u} = \boldsymbol{\tau}$ ) to be able to exploit the natural dynamics of the manipulator and obtain smoother motions. Solving the Optimal Control Problem (OCP) produces optimal state and control trajectories over a fixed time horizon, where optimality is defined by a set of weighted high-level objectives. The resolution of the OCP is done in the framework of Differential Dynamic Programming (DDP) [45] and is implemented using the Feasibility-driven

DDP solver [43]. The OCP is transcribed to a nonlinear program by discretizing the continuous problem using a direct multiple-shooting strategy:

$$\begin{aligned} \arg \min_{\substack{\mathbf{u}_0, \dots, \mathbf{u}_{M-1} \\ \mathbf{x}_1, \dots, \mathbf{x}_M}} & \sum_{i=0}^{M-1} l_i(\mathbf{x}_i, \mathbf{u}_i) + l_M(\mathbf{x}_M), \\ \text{s.t. } & \mathbf{x}_{i+1} = f(\mathbf{x}_i, \mathbf{u}_i), \forall i \in \{0, \dots, M-1\}, \\ & \mathbf{x}_0 = \hat{\mathbf{x}}, \end{aligned} \quad (1)$$

where  $\hat{\mathbf{x}}$  is the latest robot state measurement,  $\mathbf{x}_i$  and  $\mathbf{u}_i$  are the state of the robot and the applied control at discrete time  $i$ ,  $f(\mathbf{x}_i, \mathbf{u}_i)$  describes the robot dynamics (*i.e.* articulated body algorithm), and  $l_i$  and  $l_M$  are running and terminal costs, respectively. The tracking objective of the control is specified by the costs, as we describe in Sec. III-C. Given an initial guess, the DDP algorithm solves (1) and returns a sequence of states and control actions by iterating Bellman recursions (see [43] for more details).

**Ricatti linearization.** Besides trivial systems and a short time horizon, it is impossible to solve OCP at the robot control frequency, *i.e.* 1 kHz. However, it has been shown [44] that the Ricatti gains  $K_0$  obtained as a byproduct of the OCP solution can be used to implement a first-order approximation of the optimal policy. Denoting by  $\boldsymbol{\tau}_0 = \mathbf{u}_0^*$  the first step of the optimal control obtained from the OCP solver, the linear approximation of the optimal policy is:

$$\boldsymbol{\tau}(\mathbf{x}) = \boldsymbol{\tau}_0 + K_0(\mathbf{x} - \mathbf{x}_0), \quad (2)$$

where  $\mathbf{x}$  is the latest robot state measurement and  $\mathbf{x}_0$  is the state of the robot for which the OCP solution was found. This computation is immediate, it decouples the OCP problem complexity from the real-time constraints, and it allows us to solve long-horizon problems.

### C. Tracking objective

The behavior of the controller is defined by the formulation of the running and terminal costs in eq (1). For our tracking problem, we formulate the costs as follows:

$$\begin{aligned} l_i(\mathbf{x}_i, \mathbf{u}_i) &= w_v l_v(\mathbf{x}_i) + l_x(\mathbf{x}_i) + l_u(\mathbf{x}_i, \mathbf{u}_i), \\ l_M(\mathbf{x}_M) &= w_v l_v(\mathbf{x}_M) + l_x(\mathbf{x}_M), \end{aligned} \quad (3)$$

where  $l_v(\cdot)$  is the tracking cost scaled by  $w_v$  and  $l_x(\cdot)$  and  $l_u(\cdot)$  are state and control regularization costs, respectively.

**Tracking cost.** We define a tracking cost to minimize the SE(3) distance between the estimated pose of the object and the reference pose of the object  $T_{\text{ref}}$ , both expressed in the robot base frame, *i.e.*:

$$l_v(\mathbf{x}) = \left\| \log \left( (T_{\text{BC}}(\mathbf{q}_k) T_k)^{-1} T_{\text{BC}}(\mathbf{q}) T_{\text{ref}} \right) \right\|^2, \quad (4)$$

where  $T_k = f_{\text{OLT}}(I_k)$  is object pose estimated by the proposed tracker,  $T_{\text{BC}}(\cdot)$  represents the forward kinematics from the robot base to the camera, and the operator  $\log$  represents the SE(3) log map [46]. The tracking cost approaches zero if the transformation between the robot camera and estimated object pose approaches  $T_{\text{ref}}$ .



**State regularization cost.** We define the cost of state regularization as  $l_x(\mathbf{x}) = (\mathbf{x} - \mathbf{x}_{\text{rest}})^\top Q_x (\mathbf{x} - \mathbf{x}_{\text{rest}})$  with  $\mathbf{x}_{\text{rest}} = (\mathbf{q}_{\text{rest}}^\top \mathbf{0}^\top)^\top$  penalizing joint configurations far from a fixed rest configuration  $\mathbf{q}_{\text{rest}}$  and penalizing high joint velocities at the same. The objective of the regularization cost is to prevent robot *null-space* motion, *i.e.* motion that does not affect the pose of the camera itself. It is required for redundant robots, where the number of DoF for robot is higher than the task-space number of DoF. Regularization of the joint velocity prevents the solver from computing motions that are too aggressive. We set  $\mathbf{q}_{\text{rest}}$  to be the first configuration read after starting the controller.

**Control regularization cost.** The control regularization, achieved by  $l_u(\mathbf{x}, \mathbf{u}) = (\mathbf{u} - \mathbf{u}_{\text{rest}}(\mathbf{x}))^\top Q_u (\mathbf{u} - \mathbf{u}_{\text{rest}}(\mathbf{x}))$ , regularizes the controls so that they are not far from  $\mathbf{u}_{\text{rest}}(\mathbf{x})$ , where  $\mathbf{u}_{\text{rest}}(\mathbf{x})$  is a torque that compensates for gravity at the robot configuration  $\mathbf{x}$ .

#### IV. EXPERIMENTS

In this section, we first quantitatively evaluate the proposed perception module on the YCBV dataset [20] that contains standardized objects that we also use in the second part of the section for the 6D pose-guided feedback control task on a real Franka Emika Panda robot. For the implementation of the localizer, we use CosyPose [10] and for the tracker we use ICG [14] unless specified otherwise.

##### A. Quantitative evaluation of the perception module

We quantitatively evaluate the new perception module on the YCBV dataset [20] using the 6D object pose (BOP benchmark) evaluation metrics [9]. The YCBV dataset consists of several videos of a moving camera showing a subset of 22 objects available in the dataset. Every frame of the video is annotated with the ground truth poses for all objects visible in the scene. We use the YCBV dataset because of the availability of the real objects for real-world experiments and of the pre-trained models for the CosyPose [10] object pose estimator. We use the BOP toolkit [9] to compute standard 6D pose error metrics to assess the quality of pose estimates. The evaluation procedure feeds the images of the input video sequence in order and with a given frequency to the perception module. The output poses are compared with the ground truth by evaluating *BOP Average Recall* score defined in [9]. The results of the evaluation procedure are shown in Fig. 4 and discussed next.

**Evaluation baselines.** There are two main baselines shown in the plot: (i) *Localizer*, that runs localizer on every frame of the video, and (ii) *Tracker-InitLocalizer* that runs the tracker with initialization computed by the localizer on the first frame of the video. Both baselines are shown as horizontal lines as they were evaluated independently on the frequency of the image stream. The *Localizer* is introducing high time delay in the system and, therefore, is not suitable for closed-loop control. However, the results show that the localizer is accurate. The *Tracker*, on the other hand, runs online with only a small delay, but is not capable of (re-)discovering new or lost object tracks. Therefore, its average recall is small.

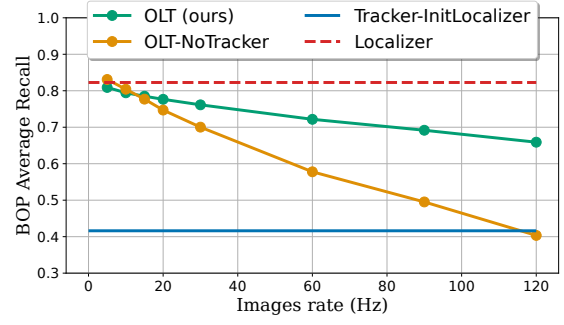


Fig. 4: Average recall (higher is better) of BOP metrics [9] measuring the accuracy of 6D pose estimation of different implementations of object localization and tracking. The comparison was run on the YCBV video dataset replayed at different frequencies on the same hardware.

**OLT evaluation.** Our method (*OLT*), lies in between the two baselines. It runs online with the same delay as *Tracker* but also achieves the *Localizer*'s recall for the low frequencies of the input image stream. The average recall drops with increasing frequency as the output pose of the localizer is injected into the tracker less frequently. Asymptotically, for image stream frequency approaching the tracker computation frequency, the *OLT*'s recall would approach performance of pure tracker as the time delay corrector would never *catch-up* in time with the tracker process.

**OLT without tracker.** To assess the influence of the tracker, we perform an ablation study in which we redefine the tracker as identity mapping, *i.e.*  $T_k = f_{\text{track}}(I_k, T_{k_1}) := T_{k-1}$ . The recall obtained for various image stream frequencies is shown as *OLT-NoTracker* curve in Fig. 4. It can be seen that the influence of the ICG tracker (*i.e.* *OLT* (ours)) increases with frequency compared to the identity tracker. Therefore, the local tracker plays an important role during the computing time of the localizer. Note that this effect is more pronounced for fast-moving objects which are not present in the YCBV dataset.

**Replicability of the results.** As shown in Fig. 4, the average recall of our method depends on the input image

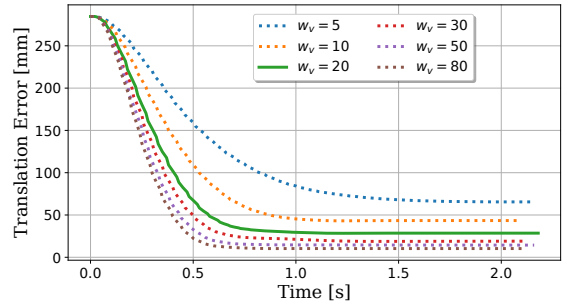


Fig. 5: Step-response of the MPC (without the perception) after simulating the target pose rotation by 30 degrees.

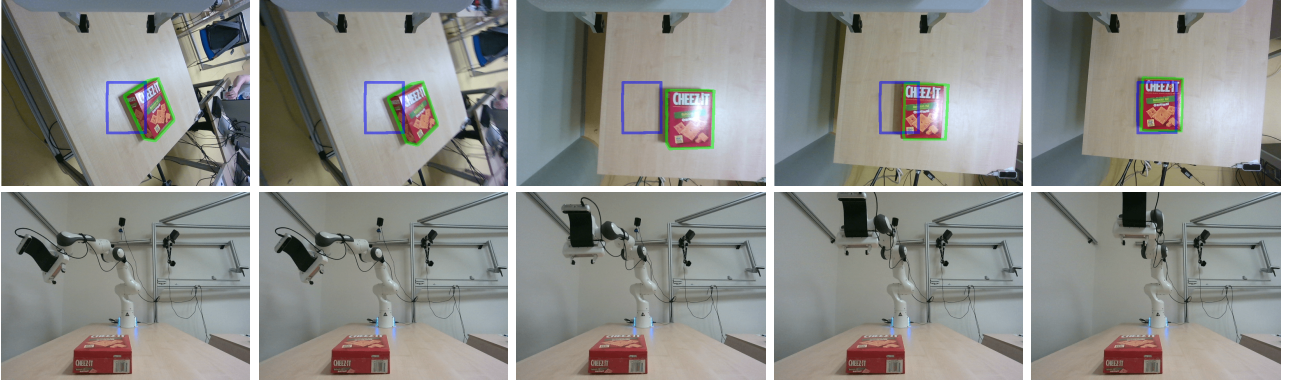


Fig. 6: **Visualization of the step-response experiment.** The first row shows the images captured by the camera mounted on the robot together with the projection of the pose estimated by our perception module (green contour) and the projection of the target reference pose (blue contour). The second row shows the images captured by an external camera depicting the motion of the robot. The goal of the controller is to move the robot end-effector to a given fixed relative pose w.r.t. the detected object pose. The initial configuration of the manipulator is intentionally set away from the target to evaluate the step response of the system. The controller brings the camera close to the desired reference pose in 1.5 seconds. **Please see the supplementary video for additional examples and experimental analysis.**

stream frequency, as the localizer produces a more accurate 6D pose less often. Therefore, the computed values also depend on the hardware, and thus are only comparable when run on comparable setups. We evaluated all methods on the same computer equipped with 12 cores *AMD® Ryzen Threadripper PRO 3945WX* CPU and a *NVIDIA GeForce RTX 3080* GPU.

### B. Visually guided feedback control

We experimentally validate the design of our 6D object perception module together with the MPC controller using the following experimental setup. We use a 7 DoF Franka Emika Panda robot equipped with a RealSense D455 camera attached to its end-effector (eye-in-hand configuration). The camera mounting w.r.t. end-effector was calibrated. We configured the camera to produce an RGB video stream at 30 Hz with a 640x480 resolution. We control the Panda robot in torque-level control mode that requires commands to be sent at 1 kHz frequency. This justifies the use of the Ricatti Linearization module, which guarantees that a torque command will be computed at this rate. The OCP is solved with Crocoddyl [43] which uses the efficient robot dynamics implementation from Pinocchio [47]. To model the dynamics of the robot, we use inertial parameters from [48]. Computations of the perception module and OCP solving is handled by a computer described in Sec. IV-A. The 1 kHz Ricatti linearization control loop is computed on another real-time-preempted computer to guarantee the response time requirements of the Panda robot. The communication between the real-time and the non-real-time computer is implemented using the robotic operating system (ROS) [49].

**MPC control evaluation.** To assess the quality of the MPC control, we perform an experiment on a Panda robot where we analyze the step response of the controller after artificially rotating the target 6D pose by 30 degrees, *i.e.* the perception

module is not used in this experiment. The evolution of the translation tracking error is shown in Fig. 5 for different values of tracking weight  $w_v$  (see eq. (3)). The results confirm that the tracker converges towards the target with steady-state error depending on the tracking weight. The orientation error (not shown) follows the same pattern. Based on the results, we have chosen the tracking weight  $w_v$  to be equal to 20 since the steady-state error is acceptable for our task and a lower weight leads to less aggressive behavior of the controller. The other weights in the cost were set as  $Q_x = \text{diag}(0.3, \dots, 0.3, 3, \dots, 3)$  and  $Q_u = \text{diag}(0.1, \dots, 0.1)$ . **MPC tracking validation.** We set up a closed-loop robot control experiment shown in Fig. 6 in which the perception and control modules enable us to bring the end-effector of the robot to a desired reference pose with respect to a YCBV object. Despite the relatively fast motion of the end effector and the presence of specular reflections on the object surface, the tracker is able to maintain an accurate estimation of the object pose throughout the trajectory. More examples are presented in the accompanying video.

### V. CONCLUSION

Accurate and low-latency object pose estimation is necessary to enable robot interaction with dynamically moving objects, for example, in human-robot handover tasks. Our work shows that a high-accuracy but slow 6D pose localizer and fast frame-to-frame 6D pose object trackers can be combined to obtain low latency ( $< 5$  ms) pose estimates. The proposed algorithm has been validated through both (i) a quantitative study on a benchmark of common household objects and (ii) by developing an MPC-based object pose tracking feedback controller. This work opens up the possibility of visually guided manipulation in 3D dynamic environments, for example, in human-robot collaboration or mobile robot manipulation, without the need for fiducial markers or motion capture systems.

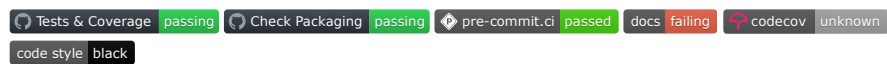
## REFERENCES

- [1] P. Furgale and T. D. Barfoot, “Visual teach and repeat for long-range rover autonomy,” *Journal of field robotics*, vol. 27, no. 5, pp. 534–560, 2010.
- [2] F. Chaumette, S. Hutchinson, and P. Corke, “Visual servoing,” *Springer handbook of robotics*, pp. 841–866, 2016.
- [3] T. Chabal, R. Strudel, E. Arlaud, J. Ponce, and C. Schmid, “Assembly planning from observations under physical constraints,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10223–10229, IEEE, 2022.
- [4] V. Ortenzi, A. Cosgun, T. Pardi, W. Chan, E. Croft, and D. Kulic, “Object handovers: a review for robotics,” Jan 2022. arXiv:2007.12952 [cs, eess].
- [5] S. S. Mirrazavi Salehian, N. Figueroa, and A. Billard, “A unified framework for coordinated multi-arm motion planning,” *The International Journal of Robotics Research*, vol. 37, no. 10, pp. 1205–1232, 2018.
- [6] N. Figueroa, S. Faraji, M. Koptev, and A. Billard, “A dynamical system approach for adaptive grasping, navigation and co-manipulation with humanoid robots,” in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7676–7682, 2020.
- [7] J. Haviland, N. Sünderhauf, and P. Corke, “A holistic approach to reactive mobile manipulation,” *IEEE Robotics and Automation Letters*, vol. 7, p. 3122–3129, Apr 2022.
- [8] T. Hodaň, M. Sundermeyer, B. Drost, Y. Labbé, E. Brachmann, F. Michel, C. Rother, and J. Matas, “Bop challenge 2020 on 6d object localization,” in *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 577–594, Springer, 2020.
- [9] M. Sundermeyer, T. Hodaň, Y. Labbe, G. Wang, E. Brachmann, B. Drost, C. Rother, and J. Matas, “Bop challenge 2022 on detection, segmentation and pose estimation of specific rigid objects,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2784–2793, 2023.
- [10] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, “Cosypose: Consistent multi-view multi-object 6d pose estimation,” in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII 16*, pp. 574–591, Springer, 2020.
- [11] R. L. Haugaard and A. G. Buch, “Surfemb: Dense and continuous correspondence distributions for object pose estimation with learnt surface embeddings,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6749–6758, 2022.
- [12] L. Lipson, Z. Teed, A. Goyal, and J. Deng, “Coupled iterative refinement for 6d multi-object pose estimation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6728–6737, 2022.
- [13] Y. Labbé, L. Manuelli, A. Mousavian, S. Tyree, S. Birchfield, J. Tremblay, J. Carpentier, M. Aubry, D. Fox, and J. Sivic, “Megapose: 6d pose estimation of novel objects via render & compare,” *arXiv preprint arXiv:2212.06870*, 2022.
- [14] M. Stoiber, M. Sundermeyer, and R. Triebel, “Iterative corresponding geometry: Fusing region and depth for highly efficient 3d tracking of textureless objects,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6855–6865, 2022.
- [15] M. Fiala, “Artag, a fiducial marker system using digital techniques,” in *CVPR*, 2005.
- [16] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, “Automatic generation and detection of highly reliable fiducial markers under occlusion,” *Pattern Recognition*, 2014.
- [17] E. Olson, “Apriltag: A robust and flexible visual fiducial system,” in *ICRA*, 2011.
- [18] R. Parosi, M. Risiglione, D. G. Caldwell, C. Semini, and V. Barasuol, “Kinematically-decoupled impedance control for fast object visual servoing and grasping on quadruped manipulators,” *arXiv preprint arXiv:2307.04918*, 2023.
- [19] E. Dantec, M. Naveau, P. Fernbach, N. Villa, G. Saurel, O. Stasse, M. Taix, and N. Mansard, “Whole-body model predictive control for biped locomotion on a torque-controlled humanoid robot,” in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, pp. 638–644, IEEE, 2022.
- [20] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, “Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols,” *arXiv preprint arXiv:1502.03143*, 2015.
- [21] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.
- [22] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, “Deepim: Deep iterative matching for 6d pose estimation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 683–698, 2018.
- [23] C. Harris and C. Stennett, “Rapid-a video rate object tracker,” in *BMVC*, pp. 1–6, 1990.
- [24] E. Rosten and T. Drummond, “Fusing points and lines for high performance tracking,” in *Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1*, vol. 2, pp. 1508–1515, Ieee, 2005.
- [25] A. I. Comport, E. Marchand, M. Pressigout, and F. Chaumette, “Real-time markerless tracking for augmented reality: the virtual visual servoing framework,” *IEEE Transactions on visualization and computer graphics*, vol. 12, no. 4, pp. 615–628, 2006.
- [26] S. Trinh, F. Spindler, E. Marchand, and F. Chaumette, “A modular framework for model-based visual tracking using edge, texture and depth features,” in *2018 IEEE, in RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 89–96.
- [27] B. Rosenhahn, T. Brox, and J. Weickert, “Three-dimensional shape knowledge for joint image segmentation and pose tracking,” *International Journal of Computer Vision*, vol. 73, pp. 243–262, 2007.
- [28] V. A. Prisacariu and I. D. Reid, “Pwp3d: Real-time segmentation and tracking of 3d objects,” *International journal of computer vision*, vol. 98, pp. 335–354, 2012.
- [29] V. A. Prisacariu, O. Köhler, D. W. Murray, and I. D. Reid, “Real-time 3d tracking and reconstruction on mobile phones,” *IEEE transactions on visualization and computer graphics*, vol. 21, no. 5, pp. 557–570, 2014.
- [30] W. Kehl, F. Tombari, S. Ilic, and N. Navab, “Real-time 3d model tracking in color and depth on a single cpu core,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 745–753, 2017.
- [31] M. Stoiber, M. Pfanne, K. H. Strobl, R. Triebel, and A. Albu-Schäffer, “Srt3d: A sparse region-based 3d object tracking approach for the real world,” *International Journal of Computer Vision*, vol. 130, no. 4, pp. 1008–1030, 2022.
- [32] L. Weiss, A. Sanderson, and C. Neuman, “Dynamic sensor-based control of robots with visual feedback,” *IEEE Journal on Robotics and Automation*, vol. 3, no. 5, pp. 404–417, 1987.
- [33] J. T. Feddema and O. R. Mitchell, “Vision-guided servoing with feature-based trajectory generation (for robots),” *IEEE Transactions on Robotics and Automation*, vol. 5, no. 5, pp. 691–700, 1989.
- [34] W. J. Wilson, C. W. Hulls, and G. S. Bell, “Relative end-effector control using cartesian position based visual servoing,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 684–696, 1996.
- [35] B. Thuilot, P. Martinet, L. Cordesses, and J. Gallice, “Position based visual servoing: keeping the object in the field of vision,” in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 2, pp. 1624–1629, IEEE, 2002.
- [36] J. Haviland, F. Dayoub, and P. Corke, “Control of the final-phase of closed-loop visual grasping using image-based visual servoing,” *arXiv preprint arXiv:2001.05650*, 2020.
- [37] M. Sauvée, P. Poignet, E. Dombre, and E. Courtial, “Image based visual servoing through nonlinear model predictive control,” in *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 1776–1781, IEEE, 2006.
- [38] G. Allibert and E. Courtial, “What can prediction bring to image-based visual servoing?,” in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5210–5215, IEEE, 2009.
- [39] P. Katara, Y. Harish, H. Pandya, A. Gupta, A. Sanchawala, G. Kumar, B. Bhowmick, and M. Krishna, “Deepmpcv: Deep model predictive control for visual servoing,” in *Conference on Robot Learning*, pp. 2006–2015, PMLR, 2021.
- [40] M. Jacquet and A. Franchi, “Motor and perception constrained nmpc for torque-controlled generic aerial vehicles,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 518–525, 2020.
- [41] H. Bildstein, A. Durand-Petiteville, and V. Cadenat, “Visual predictive control strategy for mobile manipulators,” in *2022 European Control Conference (ECC)*, pp. 1672–1677, IEEE, 2022.
- [42] H. Bildstein, A. Durand-Petiteville, and V. Cadenat, “Multi-camera visual predictive control strategy for mobile manipulators,” in *2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 476–482, IEEE, 2023.

- [43] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, "Crocodyl: An efficient and versatile framework for multi-contact optimal control," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2536–2542, IEEE, 2020.
- [44] E. Dantec, M. Taix, and N. Mansard, "First order approximation of model predictive control solutions for high frequency feedback," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4448–4455, 2022.
- [45] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.
- [46] J. Sola, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," *arXiv preprint arXiv:1812.01537*, 2018.
- [47] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiriaux, O. Stasse, and N. Mansard, "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.
- [48] C. Gaz, M. Cagnetti, A. Oliva, P. R. Giordano, and A. De Luca, "Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using penalty-based optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4147–4154, 2019.
- [49] M. Quigley, "Ros: an open-source robot operating system," in *IEEE International Conference on Robotics and Automation*, 2009.

## C Documentation of HappyPose

# HappyPose



Toolbox and trackers for object pose-estimation. Based on the work [CosyPose](#) and [MegaPose](#). This directory is currently under development. Please refer to the [documentation](#) for more details.

## Installation

This installation procedure will be curated.

```
git clone --branch dev --recurse-submodules https://github.com/agimus-
project/happypose.git
cd happypose
conda env create -f environment.yml
conda activate happypose
cd happypose/pose_estimators/cosypose
pip install .
cd ../../..
pip install -e .
```

Installation of bop\_toolkit :

```
conda activate happypose
cd happypose/pose_estimators/megapose/deps/bop_toolkit_challenge/
# Remove all versions enforcing on requirements.txt
pip install -r requirements.txt -e .
```

## Create data directory

Create data dir /somewhere/convenient. The dataset to store are quite large.  
export HAPPYPOSE\_DATA\_DIR=/somewhere/convenient

## Configuration for the evaluation

If you plan on evaluating CosyPose and Megapose, you need to modify the following lines in `bop_toolkit_lib/config.py`, replace

```
##### Basic #####

# Folder with the BOP datasets.
if 'BOP_PATH' in os.environ:
    datasets_path = os.environ['BOP_PATH']
else:
    datasets_path = r'/path/to/bop/datasets'

# Folder with pose results to be evaluated.
results_path = r'/path/to/folder/with/results'

# Folder for the calculated pose errors and performance scores.
eval_path = r'/path/to/eval/folder'
```

with

```
##### Basic #####

# Folder with the BOP datasets.
datasets_path = str(os.environ['BOP_DATASETS_PATH'])
results_path = str(os.environ['BOP_RESULTS_PATH'])
eval_path = str(os.environ['BOP_EVAL_PATH'])
```

You will also need to install [TEASER++](#) if you want to use the depth for MegaPose. To do so, please run the following commands to install it :

```
# Go to HappyPose root directory
apt install -y cmake libeigen3-dev libboost-all-dev
conda activate happypose
mamba install compilers -c conda-forge
pip install open3d
mkdir /build && cd /build && git clone https://github.com/MIT-SPARK/TEASER-
plusplus.git
cd TEASER-plusplus && mkdir build && cd build
cmake -DTEASERPP_PYTHON_VERSION=3.9 .. && make teaserpp_python
cd python && pip install .
```



# CosyPose: Consistent multi-view multi-object 6D pose estimation

Yann Labbé, Justin Carpentier, Mathieu Aubry, Josef Sivic

ECCV: European Conference on Computer Vision, 2020

[\[Paper\]](#) [\[Project page\]](#) [\[Video \(1 min\)\]](#) [\[Video \(10 min\)\]](#) [\[Slides\]](#)

Winner of the [BOP Challenge 2020](#) at ECCV'20 [\[slides\]](#) [\[BOP challenge paper\]](#)

## Citation

If you use this code in your research, please cite the paper:

```
@inproceedings{labbe2020,  
  title={CosyPose: Consistent multi-view multi-object 6D pose estimation}  
  author={Y. {Labbe} and J. {Carpentier} and M. {Aubry} and J. {Sivic}},  
  booktitle={Proceedings of the European Conference on Computer Vision (ECCV)},  
  year={2020}}
```

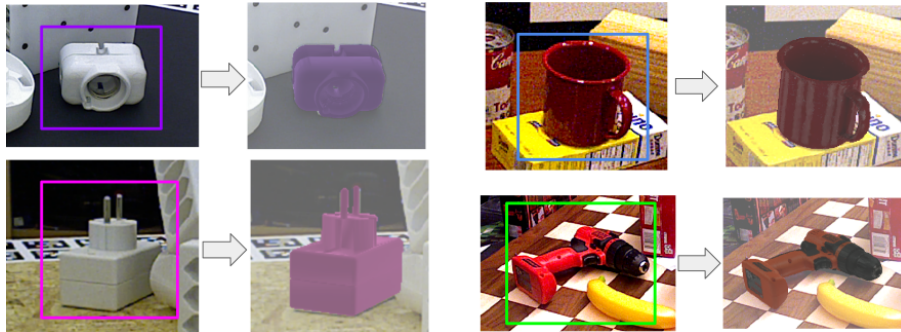
## News

- CosyPose is the winning method in the [BOP challenge 2020](#) (5 awards in total, including best overall method and best RGB-only method) ! All the code and models used for the challenge are available in this repository.
- We participate in the [BOP challenge 2020](#). Results are available on the public [leaderboard](#) for 7 pose estimation benchmarks. We release 2D detection models (MaskRCNN) and 6D pose estimation models (coarse+refiner) used on each dataset.
- The paper is available on arXiv and full code is released.
- Our paper on CosyPose is accepted at ECCV 2020.

This repository contains the code for the full CosyPose approach, including:

## Overview

### Single-view single-object 6D pose estimator



Given an RGB image and a 2D bounding box of an object with known 3D model, the 6D pose estimator predicts the full 6D pose of the object with respect to the camera. Our method is inspired by DeepIM with several simplifications and technical improvements. It is fully implemented in pytorch and achieve single-view state-of-the-art on YCB-Video and T-LESS. We provide pre-trained models used in our experiments on both datasets. We make the training code that we used to train them available. It can be parallelized on multiple GPUs and multiple nodes.

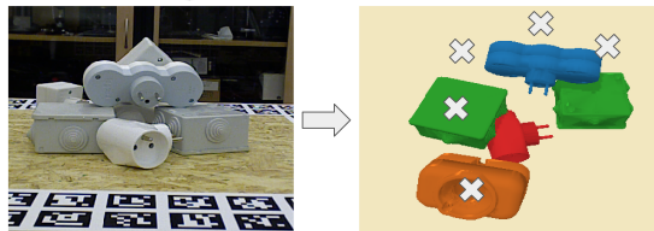
## Synthetic data generation



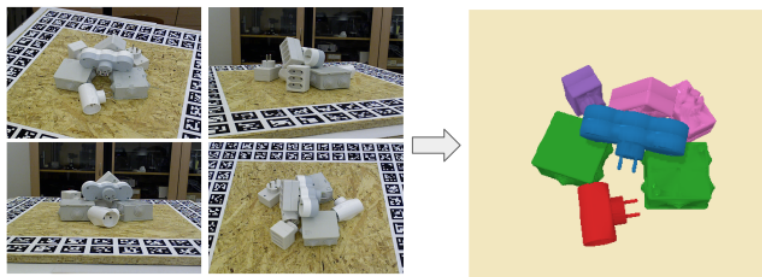
The single-view 6D pose estimation models are trained on a mix of synthetic and real images. We provide the code for generating the additional synthetic images.

## Multi-view multi-object scene reconstruction

Single-view reconstruction



Multi-view reconstruction with CosyPose



Single-view object-level reconstruction of a scene often fails because of detection mistakes, pose estimation errors and occlusions; which makes it impractical for real applications. Our multi-view approach, CosyPose, addresses these single-view limitations and helps improving 6D pose accuracy by leveraging information from multiple cameras with unknown positions. We provide the full code, including robust object-level multi-view matching and global scene refinement. The method is agnostic to the 6D pose estimator used, and can therefore be combined with many other existing single-view object pose estimation method to solve problems on other datasets, or in real scenarios. We provide a utility for running CosyPose given a set of input 6D object candidates in each image.

## BOP challenge 2020: single-view 2D detection + 6D pose estimation models



We used our {coarse+refinement} single-view 6D pose estimation method in the [BOP challenge 2020](#). In addition, we trained a MaskRCNN detector (torchvision's implementation) on each of the 7 core datasets (LM-O, T-LESS, TUD-L, IC-BIN, ITODD, HB, YCB-V). We provide 2D detectors and 6D pose estimation models for these datasets. All training (including 2D detector), inference and evaluation code are available in this repository. It can be easily used for another dataset in the BOP format.

## Main entry points

This repository is divided into different entry points

- **Inference:** `run_cosypose_on_example.py` is used to run the inference pipeline on a single example image.
- **Evaluation:** `run_full_cosypose_evaluation.py` is used to first run inference on one or several datasets, and then use the results obtained to evaluate the method on these datasets.
- **Training:** `run_detector_training.py` is used to train the detector part of Cosypose. `run_pose_training.py` can be used to train the `coarse` model or the `refiner` model.

In this repository, the version provided of CosyPose is different to the one of the original repository. In particular, we switched the 3D renderer from [PyBullet](#) to [Panda3d](#). Thus, the results obtained may differ from the one reported in the original paper and repository.

## Downloading and preparing the data

All data used (datasets, models, results, ...) are stored in a directory `$HAPPYPOSE_DATA_DIR` that you created in the Readsme. We provide the utilities for downloading required data and models. All of the files can also be [downloaded manually](#).

### BOP Datasets

For both T-LESS and YCB-Video, we use the datasets in the [BOP format](#). If you already have them on your disk, place them in `$HAPPYPOSE_DATA_DIR/bop_datasets`. Alternatively, you can download it using :

```
python -m happypose.toolbox.utils.download --bop_dataset=ycbv
python -m happypose.toolbox.utils.download --bop_dataset=tless
```

Additional files that contain information about the datasets used to fairly compare with prior works on both datasets.

```
python -m happypose.toolbox.utils.download --bop_extra_files=ycbv
python -m happypose.toolbox.utils.download --bop_extra_files=tless
```

We use [pybullet](#) for rendering images which requires object models to be provided in the URDF format. We provide converted URDF files, they can be downloaded using:

```
python -m happypose.toolbox.utils.download --urdf_models=ycbv
python -m happypose.toolbox.utils.download --urdf_models=tless.cad
```

In the BOP format, the YCB objects `002_master_chef_can` and `040_large_marker` are considered symmetric, but not by previous works such as PoseCNN, PVNet and DeepIM. To ensure a fair comparison (using ADD instead of ADD-S for ADD(S) for these objects), these objects must *not* be considered symmetric in the evaluation. To keep the uniformity of the models format, we generate a set of YCB objects `models_bop-compat_eval` that can be used to fairly compare our approach against previous works. You can download them directly:

```
python -m happypose.toolbox.utils.download --ycbv_compat_models
```



Notes:

- The URDF files were obtained using these commands (requires `meshlab` to be installed):

```
python -m
happypose.pose_estimators.cosypose.cosypose.scripts.convert_models_to_urdf --
models=ycbv
python -m
happypose.pose_estimators.cosypose.cosypose.scripts.convert_models_to_urdf --
models=tless.cad
```

- Compatibility models were obtained using the following script:

```
python -m
happypose.pose_estimators.cosypose.cosypose.scripts.make_ycbv_compat_models
```

## Models for minimal version

```
#ycbv
python -m happypose.toolbox.utils.download --cosypose_model=detector-bop-ycbv-pbr-
-970850
python -m happypose.toolbox.utils.download --cosypose_model=coarse-bop-ycbv-pbr-
-724183
python -m happypose.toolbox.utils.download --cosypose_model=refiner-bop-ycbv-pbr-
-604090

#tless
python -m happypose.toolbox.utils.download --cosypose_model=detector-bop-tless-pbr-
-873074
python -m happypose.toolbox.utils.download --cosypose_model=coarse-bop-tless-pbr-
-506801
python -m happypose.toolbox.utils.download --cosypose_model=refiner-bop-tless-pbr-
-233420
```

## Pre-trained models for single-view estimator

The pre-trained models of the single-view pose estimator can be downloaded using:

```
# YCB-V Single-view refiner
python -m happypose.toolbox.utils.download --cosypose_model=ycbv-refiner-finetune-251020

# YCB-V Single-view refiner trained on synthetic data only
# Only download this if you are interested in retraining the above model
python -m happypose.toolbox.utils.download --cosypose_model=ycbv-refiner-synonly-596719

# T-LESS coarse and refiner models
python -m happypose.toolbox.utils.download --cosypose_model=tless-coarse--10219
python -m happypose.toolbox.utils.download --cosypose_model=tless-refiner--585928
```

## 2D detections

To ensure a fair comparison with prior works on both datasets, we use the same detections as DeepIM (from PoseCNN) on YCB-Video and the same as Pix2pose (from a RetinaNet model) on T-LESS. Download the saved 2D detections for both datasets using

```
python -m happypose.toolbox.utils.download --detections=ycbv_posecnn

# SiSo detections: 1 detection with highest per score per class per image on all
images
# Available for each image of the T-LESS dataset (primesense sensor)
# These are the same detections as used in Pix2pose's experiments
python -m happypose.toolbox.utils.download --
detections=tless_pix2pose_retinanet_asiso_top1

# ViVo detections: All detections for a subset of 1000 images of T-LESS.
# Used in our multi-view experiments.
python -m happypose.toolbox.utils.download --
detections=tless_pix2pose_retinanet_vivo_all
```

If you are interested in re-training a detector, please see the BOP 2020 section.

Notes:

- The PoseCNN detections (and coarse pose estimates) on YCB-Video were extracted and converted from [these PoseCNN results](#).
- The Pix2pose detections were extracted using [pix2pose's](#) code. We used the detection model from their paper, see [here](#). For the ViVo detections, their code was slightly modified. The code used to extract detections can be found [here](#).

## Inference

Here are provided the minimal commands you have to run in order to run the inference of CosyPose. You need to set up the environment variable `$HAPPYPOSE_DATA_DIR` as explained in the README.

### 1. Download pre-trained pose estimation models

```
#ycbv
python -m happypose.toolbox.utils.download --cosypose_model=detector-bop-ycbv-pbr-
-970850
python -m happypose.toolbox.utils.download --cosypose_model=coarse-bop-ycbv-pbr-
-724183
python -m happypose.toolbox.utils.download --cosypose_model=refiner-bop-ycbv-pbr-
-604090
```

### 2. Download YCB-V Dataset

```
python -m happypose.toolbox.utils.download --bop_dataset=ycbv
```

### 3. Download the example

```
cd $HAPPYPOSE_DATA_DIR
wget https://memmo-data.laas.fr/static/examples.tar.xz
tar xf examples.tar.xz
```

### 4. Run the script

```
python -m happypose.pose_estimators.cosypose.cosypose.scripts.run_inference_on_example
crackers --run-inference
```

## 5. Results

The results are stored in the visualization folder created in the crackers example directory.



## Evaluating CosyPose

Please make sure you followed the steps relative to the evaluation in the main readme.

Please run the following command to evaluate on YCBV dataset

```
python -m
happypose.pose_estimators.cosypose.cosypose.scripts.run_full_cosypose_eval_new
detector_run_id=bop_pbr coarse_run_id=coarse-bop-ycbv-pbr--724183
refiner_run_id=refiner-bop-ycbv-pbr--604090 ds_names=["ycbv.bop19"] result_id=ycbv-
debug detection_coarse_types=[["detector","$03_grid"]]
```

The other BOP datasets are supported as long as you download the correspond models.

## Train CosyPose

Disclaimer : This part of the repository is still under development.

Training the detector part of the pose estimation part are independant.

## Training Pose Estimator

This script can be used to train both the coarse model or the refiner model.

```
python -m happypose.pose_estimators.cosypose.cosypose.scripts.run_pose_training --  
config ycbv-refiner-syntonly
```

## Training Detector

```
python -m happypose.pose_estimators.cosypose.cosypose.scripts.run_detector_training --  
config bop-ycbv-synt+real
```

All the models were trained on 32 GPUs.

# MegaPose

This repository contains code, models and dataset for our MegaPose paper.

Yann Labbé, Lucas Manuelli, Arsalan Mousavian, Stephen Tyree, Stan Birchfield, Jonathan Tremblay, Justin Carpentier, Mathieu Aubry, Dieter Fox, Josef Sivic. "MegaPose: 6D Pose Estimation of Novel Objects via Render & Compare." In: CoRL 2022.

[\[Paper\]](#) [\[Project page\]](#)

## News

- **09.01.2023** We release two new variants of our approach (see the [Model Zoo](#)).
- **09.01.2023** Code, models and dataset are released in this repository.
- **10.09.2022** The paper is accepted at CoRL 2022.

## Contributors

The main contributors to the code are:

- [Yann Labbé](#) (Inria, NVIDIA internship)
- [Lucas Manuelli](#) (NVIDIA Seattle Robotics Lab)

## Citation

If you find this source code useful please cite:

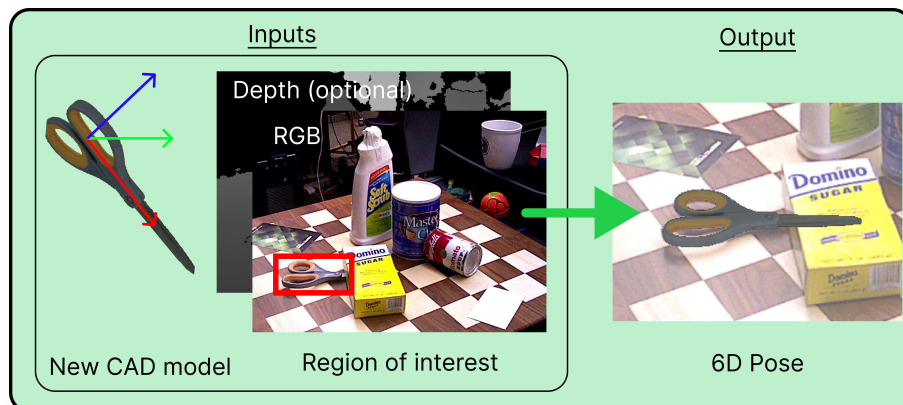


```
@inproceedings{labbe2022megapose,
  title = {{{MegaPose}}: {{6D Pose Estimation}} of {{Novel Objects}} via {{Render}} \&
{{Compare}}},
  booktitle = {CoRL},
  author = {Labbe, Yann and Manuelli, Lucas and Mousavian, Arsalan and Tyree,
Stephen and Birchfield, Stan and Tremblay, Jonathan and Carpentier, Justin and Aubry,
Mathieu and Fox, Dieter and Sivic, Josef},
  date = {2022}
}
```

## Overview

This repository contains pre-trained models for pose estimation of novel objects, and our synthetic training dataset. Most notable features are listed below.

### Pose estimation of novel objects



We provide pre-trained models for 6D pose estimation of novel objects.

Given as inputs:

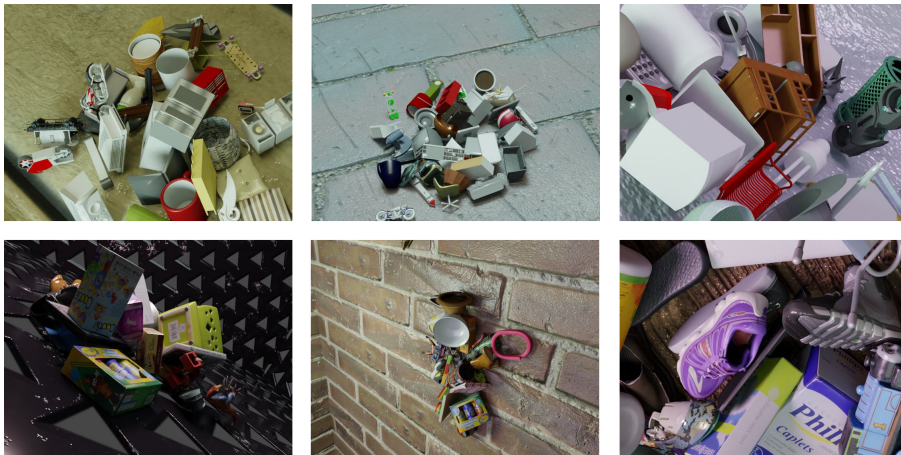
- an RGB image (depth can also be used but is optional),
- the intrinsic parameters of the camera,

- a mesh of the object,
- a bounding box of that object in the image,

our approach estimates the 6D pose of the object (3D rotation + 3D translation) with respect to the camera.

We provide a script and an example for inference on novel objects. After installation, please see the [Inference tutorial](#).

## Large-scale synthetic dataset



We provide the synthetic dataset we used to train MegaPose. The dataset contains 2 million images displaying more than 20,000 objects from the Google Scanned Objects and ShapeNet datasets. After installation, please see the [Dataset section](#).

## Main entry points

This repository is divided into different entry points

- **Inference:** `run_megapose_on_example.py` is used to run the inference pipeline on a single example image.
- **Evaluation:** `run_full_megapose_eval.py` is used to first run inference on one or several datasets, and then use the results obtained to evaluate the method on these datasets.

## Model Zoo

Model name	Input
megapose-1.0-RGB	RGB
megapose-1.0-RGBD	RGB-D
megapose-1.0-RGB-multi-hypothesis	RGB
megapose-1.0-RGB-multi-hypothesis-icp	RGB-D

- `megapose-1.0-RGB` and `megapose-1.0-RGBD` correspond to method presented and evaluated in the paper.
- `-multi-hypothesis` is a variant of our approach which:
  - Uses the coarse model, extracts top-K hypotheses (by default K=5);
  - For each hypothesis runs K refiner iterations;
  - Evaluates refined hypotheses using score from coarse model and selects the highest scoring one.
- `-icp` indicates running ICP refinement on the depth data.

For optimal performance, we recommend using `megapose-1.0-RGB-multi-hypothesis` for an RGB image and `megapose-1.0-RGB-multi-hypothesis-icp` for an RGB-D image. An extended paper with full evaluation of these new approaches is coming soon.

## Download example data for minimal testing

```
cd $HAPPYPOSE_DATA_DIR
wget https://memmo-data.laas.fr/static/examples.tar.xz
tar xf examples.tar.xz
```

## Download pre-trained pose estimation models

Download pose estimation models to `$HAPPYPOSE_DATA_DIR/megapose-models`:

```
python -m happypose.toolbox.utils.download --megapose_models
```

## Dataset

### Dataset information

The dataset is available at this [url](#). It is split into two datasets: `gso_1M` (Google Scanned Objects) and `shapenet_1M` (ShapeNet objects). Each dataset has 1 million images which were generated using [BlenderProc](#).

Datasets are released in the [webdataset](#) format for high reading performance. Each dataset is split into chunks of size ~600MB containing 1000 images each.

We provide the pre-processed meshes ready to be used for rendering and training in this [directory](#):

- `google_scanned_objects.zip`
- `shapenetcorev2.zip`

**Important:** Before downloading this data, please make sure you are allowed to use these datasets i.e. you can download the original ones.

## Usage

We provide utilities for loading and visualizing the data.

The following commands download 10 chunks of each dataset as well as metadatas:

```
cd $HAPPYPOSE_DATA_DIR
rclone copyto megapose_public_readonly:/webdatasets/ webdatasets/ --include
"00000000*.tar" --include "*.json" --include "*.feather" --config
$MEGAPOSE_DIR/rclone.conf -P
```

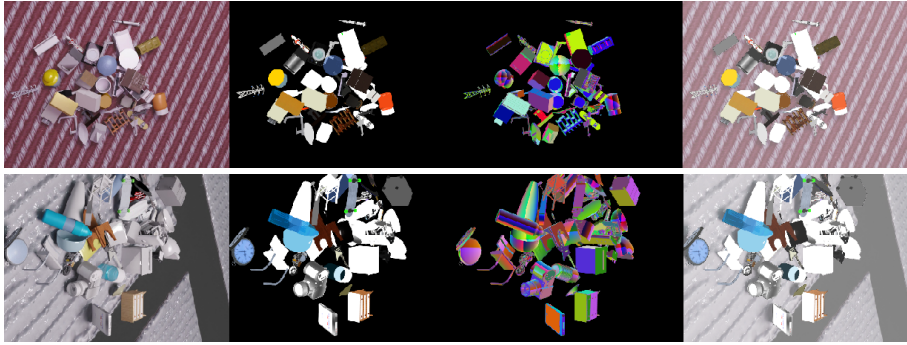
We then download the object models (please make sure you have access to the original datasets before downloading these preprocessed ones):

```
cd $HAPPYPOSE_DATA_DIR
rclone copyto megapose_public_readonly:/tars tars/ --include "shapenetcorev2.zip" --
include "google_scanned_objects.zip" --config $MEGAPOSE_DIR/rclone.conf -P
unzip tars/shapenetcorev2.zip
unzip tars/google_scanned_objects.zip
```

Your directory structure should look like this:

```
$HAPPYPOSE_DATA_DIR/
  webdatasets/
    gso_1M/
      infos.json
      frame_index.feather
      00000001.tar
      ...
    shapenet_1M/
      infos.json
      frame_index.feather
      00000001.tar
      ...
    shapenetcorev2/
      ...
    googlescannedobjects/
      ...
```

You can then use the [render\\_megapose\\_dataset.ipynb](#) notebook to load and visualize the data and 6D pose annotations.



## Inference

Here are provided the minimal commands you have to run in order to run the inference of CosyPose. You need to set up the environment variable `$HAPPYPOSE_DATA_DIR` as explained in the README.

### 1. Download pre-trained pose estimation models

```
python -m happypose.toolbox.utils.download --megapose_models
```

### 2. Download the example

We estimate the pose for a barbecue sauce bottle (from the [HOPE](#) dataset, not used during training of MegaPose).

```
cd $HAPPYPOSE_DATA_DIR
wget https://memmo-data.laas.fr/static/examples.tar.xz
tar xf examples.tar.xz
```

The input files are the following:

```
$HAPPYPOSE_DATA_DIR/examples/barbecue-sauce/
image_rgb.png
image_depth.png
camera_data.json
inputs/object_data.json
meshes/barbecue-sauce/hope_000002.ply
meshes/barbecue-sauce/hope_000002.png
```

- `image_rgb.png` is a RGB image of the scene. We recommend using a 4:3 aspect ratio.
- `image_depth.png` (optional) contains depth measurements, with values in `mm`. You can leave out this file if you don't have depth measurements.
- `camera_data.json` contains the 3x3 camera intrinsic matrix  $\kappa$  and the camera resolution in `[h,w]` format.



```
{ "K": [[605.9547119140625, 0.0, 319.029052734375], [0.0, 605.006591796875, 249.67617797851562], [0.0, 0.0, 1.0]], "resolution": [480, 640]}
```

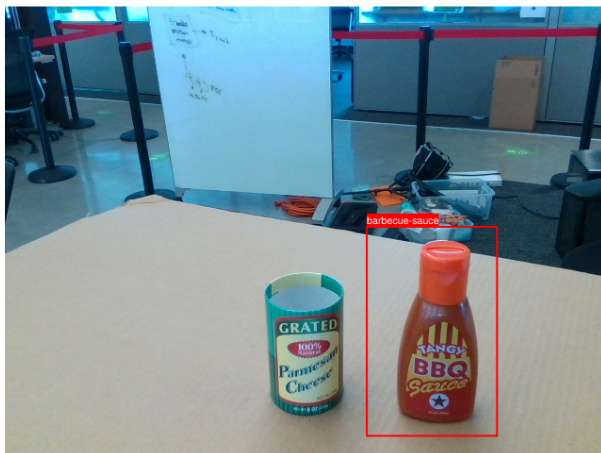
- `inputs/object_data.json` contains a list of object detections. For each detection, the 2D bounding box in the image (in `[xmin, ymin, xmax, ymax]` format), and the label of the object are provided. In this example, there is a single object detection. The bounding box is only used for computing an initial depth estimate of the object which is then refined by our approach. The bounding box does not need to be extremely precise (see below).

```
[{"label": "barbecue-sauce", "bbox_model": [384, 234, 522, 455]}]
```

- `meshes/barbecue-sauce` is a directory containing the object's mesh. Mesh units are expected to be in millimeters. In this example, we use a mesh in `.ply` format. The code also supports `.obj` meshes but you will have to make sure that the objects are rendered correctly with our renderer.

You can visualize input detections using :

```
python -m happypose.pose_estimators.megapose.scripts.run_inference_on_example
barbecue-sauce --vis-detections
```



### 3. Run pose estimation and visualize results

Run inference with the following command:

```
python -m happypose.pose_estimators.megapose.scripts.run_inference_on_example  
barbecue-sauce --run-inference
```

by default, the model only uses the RGB input. You can use of our RGB-D megapose models using the `--model` argument. Please see our [Model Zoo](#) for all models available.

The previous command will generate the following file:

```
$HAPPYPOSE_DATA_DIR/examples/barbecue-sauce/  
outputs/object_data.json
```

This file contains a list of objects with their estimated poses . For each object, the estimated pose is noted `Two` (the world coordinate frame correspond to the camera frame). It is composed of a quaternion and the 3D translation:

```
[{"label": "barbecue-sauce", "Two": [[0.5453961536730983, 0.6226545207599095,  
-0.43295293693197473, 0.35692612413663855], [0.10723329335451126, 0.07313819974660873,  
0.45735278725624084]]}]
```

Finally, you can visualize the results using:

```
python -m happypose.pose_estimators.megapose.scripts.run_inference_on_example  
barbecue-sauce --run-inference --vis-outputs
```

which write several visualization files:

```
$HAPPYPOSE_DATA_DIR/examples/barbecue-sauce/  
visualizations/contour_overlay.png  
visualizations/mesh_overlay.png  
visualizations/all_results.png
```



## Evaluating Megapose

Please make sure you followed the steps relative to the evaluation in the main readme.

An example to run the evaluation on `ycbv` dataset. Several datasets can be added to the list.

```
python -m
happypose.pose_estimators.megapose.src.megapose.scripts.run_full_megapose_eval
detector_run_id=bop_pbr coarse_run_id=coarse-rgb-906902141 refiner_run_id=refiner-rgb-
653307694 ds_names=[ycbv.bop19] result_id=fastsam_kbestdet_1posehyp
detection_coarse_types=[["sam","S03_grid"]] inference.n_pose_hypotheses=1
skip_inference=true run_bop_eval=true
```

To reproduce the results we obtained for the BOP-Challenge, please run the following commands :

```
# RGB 1 hyp
python -m
happypose.pose_estimators.megapose.src.megapose.scripts.run_full_megapose_eval
detector_run_id=bop_pbr coarse_run_id=coarse-rgb-906902141 refiner_run_id=refiner-rgb-
653307694 ds_names=
[ycbv.bop19,lmo.bop19,tless.bop19,tudl.bop19,icbin.bop19,hb.bop19,itodd.bop19]
result_id=fastsam_kbestdet_1posehyp detection_coarse_types=[["sam","S03_grid"]]
inference.n_pose_hypotheses=1 skip_inference=False run_bop_eval=true
```

Results :

Date (UTC)	Method	Test image	AR <sub>Core</sub>	AR <sub>LM-0</sub>	AR <sub>FLESS</sub>	AR <sub>TUD-L</sub>	AR <sub>ICBIN</sub>	AR <sub>ITODD</sub>	AR <sub>HB</sub>	AR <sub>YCB-V</sub>	Time (s)
1 2023-09-04	<a href="#">CNOS_fastSAM+MegaPose</a>	RGB	0.509	0.499	0.477	0.653	0.367	0.315	0.654	0.601	31.724

```
# RGB 5 hyp
python -m
happypose.pose_estimators.megapose.src.megapose.scripts.run_full_megapose_eval
detector_run_id=bop_pbr coarse_run_id=coarse-rgb-906902141 refiner_run_id=refiner-rgb-
653307694 ds_names=
[ycbv.bop19,lmo.bop19,tless.bop19,tudl.bop19,icbin.bop19,hb.bop19,itodd.bop19]
result_id=fastsam_kbestdet_5posehyp detection_coarse_types=[["sam","S03_grid"]]
inference.n_pose_hypotheses=5 skip_inference=False run_bop_eval=true
```

Results :

Date (UTC)	Method	Test image	AR <sub>Core</sub>	AR <sub>LM-0</sub>	AR <sub>FLESS</sub>	AR <sub>TUD-L</sub>	AR <sub>ICBIN</sub>	AR <sub>ITODD</sub>	AR <sub>HB</sub>	AR <sub>YCB-V</sub>	Time (s)
1 2023-09-26	<a href="#">CNOS_fastSAM+MegaPose_MultiHyp</a>	RGB	0.540	0.548	0.507	0.665	0.400	0.337	0.702	0.620	67.438

```
# RGB-D 5 hyp
python -m torch.distributed.run --nproc_per_node gpu -m
happypose.pose_estimators.megapose.src.megapose.scripts.run_full_megapose_eval
detector_run_id=bop_pbr coarse_run_id=coarse-rgb-906902141 refiner_run_id=refiner-rgb-
653307694 ds_names=[tless.bop19,tudl.bop19,icbin.bop19,hb.bop19,itodd.bop19]
result_id=fastsam_kbestdet_5posehyp_teaserpp detection_coarse_types=
[["sam", "S03_grid"]] inference.n_pose_hypotheses=5 inference.run_depth_refiner=true
inference.depth_refiner=teaserpp skip_inference=False run_bop_eval=True
```

Results :

Date (UTC)	Method	Test image	AR <sub>Core</sub>	AR <sub>LM-O</sub>	AR <sub>FLESS</sub>	AR <sub>TUD-L</sub>	AR <sub>IC-BIN</sub>	AR <sub>TODD</sub>	AR <sub>HB</sub>	AR <sub>YCB-V</sub>	Time (s)
1 2023-09-07	CNOS_fastSAM+Megapose_Multihyp_Teaserpp	RGB-D	0.609	0.588	0.477	0.848	0.423	0.461	0.704	0.758	97.309

## Example on Jean Zay supercalculator

In particular, for this challenge, we used Jean Zay, a french supercalculator. Here is a quick documentation, for additional information on who can use this calculator, please refer to the [official documentation](#).

You need to create an account to log on Jean Zay : <https://www.edari.fr/>

To connect by ssh to Jean Zay using this account, you need to register the IP address of the machine you use to connect to Jean Zay. If you work in a french research laboratory, your laboratory probably have a bouncing machine that is registered.

Once you are connected to Jean Zay, you will have access to different storage space: \$HOME , \$WORK , \$SCRATCH , \$STORE . More details on [Jean Zay website](#)

You should store your code in \$WORK and the data on \$SCRATCH . Be careful, everything not used during 30 days on \$SCRATCH is deleted.

Before following the regular installation procedure of HappyPose , make sur to load this module :  
 module load anaconda-py3/2023.03

Then, you can follow the procedure in your current shell.

Once it is done, to run a job you need to use slurm . More detail on [Jean Zay website](#).

Here are some examples of slurm scripts used during the project. To run a slurm script, use the following command : sbatch script.slurm . You can use the command sacct to see the state of

your script. You can visualize the content of the logs using the command `tail -f`. For example to see the error logs, use `tail -f logs/inference-happypose.err`.

```
# inference.slurm
#!/bin/bash
#SBATCH --job-name=happypose-inference
#SBATCH --output=logs/inference-happypose.out
#SBATCH --error=logs/inference-happypose.err
#SBATCH --nodes=1 # on demande un noeud
#SBATCH --ntasks-per-node=1 # avec une tache par noeud (= nombre de GPU ici)
#SBATCH --gres=gpu:1
#SBATCH --cpus-per-task=10
#SBATCH --hint=nomultithread
#SBATCH --account zbb@v100
#SBATCH --time=00:10:00

## load Pytorch module
module purge
module load module load anaconda-py3/2023.03
conda activate happypose

cd happypose
# python -m
happypose.pose_estimators.megapose.src.megapose.scripts.run_inference_on_example
barbecue-sauce --run-inference --vis-outputs
python -m happypose.pose_estimators.cosypose.cosypose.scripts.run_inference_on_example
crackers --run-inference
```

```

# evaluation.slurm
#!/bin/bash
#SBATCH --job-name=happypose-evaluation-1H
#SBATCH --output=logs/evaluation-happypose-1h.out
#SBATCH --error=logs/evaluation-happypose-1h.err
#SBATCH -C v100-32g
#SBATCH --nodes=1                # on demande un noeud
#SBATCH --ntasks-per-node=4      # avec une tache par noeud (= nombre de GPU ici)
#SBATCH --gres=gpu:4
#SBATCH --cpus-per-task=10
#SBATCH --hint=nomultithread
#SBATCH --account zbb@v100
#SBATCH --time=04:00:00

## load Pytorch module
module purge
module load anaconda-py3/2023.03
conda activate happypose_pytorch3d

cd happypose

python -m torch.distributed.run --nproc_per_node gpu -m
happypose.pose_estimators.megapose.src.megapose.scripts.run_full_megapose_eval
detector_run_id=bop_pbr coarse_run_id=coarse-rgb-906902141 refiner_run_id=refiner-rgb-
653307694 ds_names=[lmo.bop19] result_id=fastsam_kbestdet_1posehyp
detection_coarse_types=[["sam", "S03_grid"]] inference.n_pose_hypotheses=1
skip_inference=False run_bop_eval=true

```