Supplementary material for Open-vocabulary object 6D pose estimation

Jaime Corsetti[1,2] Davide Boscaini[1] Changjae Oh[3] Andrea Cavallaro[4,5] Fabio Poiesi[1]

[1]Fondazione Bruno Kessler [2]University of Trento [3]Queen Mary University of London [4]Idiap Research Institute [5]EPFL

{jcorsetti,dboscaini,poiesi}@fbk.eu c.oh@qmul.ac.uk andrea.cavallaro@epfl.ch

1. Introduction

We provide additional material in support of our main paper. This document is organized as follows:

- In Sec. 2, we compare the assumptions and requirements of Oryon with those of state-of-the-art methods that address pose estimation in a generalizable setting.
- In Sec. 3, we provide additional implementation details and display the architecture of the fusion module (ϕ_{TV}) in Sec. 3.1 and the decoder (ϕ_D) in Sec. 3.2.
- In Sec. 4, we outline the process we followed to generate our training and test datasets.
- In Sec. 5 we extend the discussion on the pose metrics.
- In Sec. 6, we list the textual prompts we used for evaluation on REAL275 [23] and Toyota-Light [7] (TOYL for short), including the alternative prompts used in the ablation studies. We also present some examples of synonym sets from ShapeNet6D [26] (SN6D for short).
- In Sec. 7, we show additional qualitative results on pose estimation (Sec. 7.1) and segmentation (Sec. 7.2) on TOYL and REAL275. Moreover, we provide examples of feature distance visualization for $\mathbf{F}^A$ and $\mathbf{F}^Q$, demonstrating how the features change when different prompts are used (Sec. 7.3).

2. Setting comparison

In Tab. 1, we present a comparison of the data requirements of state-of-the-art methods for generalizable pose estimation. For each method, we report the input data, the reference used to estimate the pose, and whether the method can estimate the full pose or only the rotation component. We also summarize the object onboarding process, which has to be carried out at evaluation time for each novel object. Finally, we report the localization prior used, i.e., an external module that is used to crop or segment the object of interest. Note that some methods include the localization module within their pipeline ([13, 20], and ours), instead of employing an external module.

We identify three main groups in the current literature. The first is model-based methods, which require a 3D model of the object at test time [2, 10, 20]. When present, the object onboarding phase requires generating a set of templates (i.e., a set of views of the novel object) from the object 3D model [2, 20].

The second is model-free methods, which assume a video sequence of the object to be available [5, 13, 22]. In these methods, the object onboarding phase is particularly cumbersome. Gen6D [13] requires the user to perform SfM to reconstruct the object point cloud, which is then manually cropped and oriented. The resulting point cloud is used as an object model. OnePose [22] and OnePose++ [5] also perform SfM on the video sequence, but they only use it to recover the camera poses, so the manual cropping of the point cloud is not needed. Therefore, the onboarding process of model-free methods requires human intervention, as opposed to that of model-based methods. OnePose and OnePose++ both adopt a prior detector (YOLOv5 [9]), while in Gen6D the localization is included in the pipeline.

The third group is composed of methods for relative pose estimation [12, 16, 24, 27]. These methods do not require any type of object model but instead rely on one or more support views of the novel object. In particular, NOPE [16] adopts a single support view, while RelPose [27] can work with two or more support views. Both methods rely on RGB data, and no geometric information is present. Because of this, they only estimate the relative rotation. On the other hand, PoseDiffusion [24] and RelPose++ [12] assume that the scenes are object-centric, and enstablish a constraint on the distance of the first view from the origin of the coordinate system (e.g., the unit distance for RelPose++). This allows both methods to estimate a relative translation, up to a global scale transformation of the scene. Due to the object-centric assumption, both methods rely on an object detector to crop the object of interest. A different approach is used in LatentFusion [17], in which a neural renderer is used to predict the depth and estimate the pose with respect

to a set of reference views. LatentFusion works with as less as a single reference view, and can estimate a complete pose in an absolute scale as also the depth information is provided. Note that the original method requires the ground truth segmentation mask at both train and test time.

Oryon is fundamentally different from model-based methods as the object 3D model is not required. With respect to model-free methods instead, Oryon does not rely on complex object onboarding procedures, and it does not assume that the novel object is physically available so that a video sequence can be captured. Instead, to test on a novel object, only a textual description of it is required. Finally, compared to most relative pose estimation methods, Oryon is able to estimate the complete pose due to the requirement of depth information, including the absolute value of the translation component. Moreover, Oryon does not rely on an external module for localization, as this step is carried out within the pipeline in an open-vocabulary fashion.

3. Implementation details

3.1. Text-visual fusion architecture

In Fig. 1, we show the details of a single layer of the fusion module ϕ_{TV}. Note that the input visual feature map $\mathbf{E}$ and the prompt features $\mathbf{e}^T$ have different feature sizes, as $\mathbf{E} \in \mathbb{R}^{1024\times24\times24}$, while $\mathbf{e}^T \in \mathbb{R}^{80\times768}$, where 80 is the number of prompt templates. To compute the cost volume, we require the same feature dimension; therefore we apply a 2D convolution to $\mathbf{E}$ to obtain a matrix $\in \mathbb{R}^{768\times24\times24}$. The guidance backbone ϕ_G is used to output a feature map $\mathbf{G} \in \mathbb{R}^{512\times24\times24}$, which is projected to a feature map $\in \mathbb{R}^{128\times24\times24}$ before concatenating with the keys and queries of the two attention layers. We also project the prompt features $\mathbf{e}^T$ to a lower-dimensional space to obtain a feature list $\in \mathbb{R}^{80\times128}$, which is concatenated to the visual feature map in the text guidance head. The text guidance head is used to inject prompt information into a lower-resolution feature map to enhance the semantic content of the features.

Note that the fusion module ϕ_{TV} is composed of two of the layers depicted in Fig. 1. Experimentally, we found beneficial to use the weights of CATSeg [3] for the window attention and shifted window attention parts of the module. In CATSeg, these two modules are followed by a Linear Transformer layer, which is used to model the relationship among the different classes in the textual prompt. Instead, in our setting, the prompt T only describes a single class (i.e., object); therefore we replace the Linear Transformer layer with the text guidance head, which is trained from scratch.

3.2. Decoder architecture

In Fig. 2, we show the details of the decoder ϕ_D. A single decoder layer is composed by a transposed 2D convolution that doubles the spatial resolution of the input feature map. The result is concatenated with a projection of the feature map derived from the guidance backbone ϕ_G. We use $\mathbf{G}_1$ for the first layer and $\mathbf{G}_2$ for the second, while the third layer does not use any guidance. After the concatenation, we apply two blocks of operations, each composed of a 2D convolution followed by group normalization and ReLU activation. For better visualization, this group of operations is depicted as ConvGroup in Fig. 2.

As for the fusion module ϕ_{TV}, for the first two upsampling layers and the guidance projections, we finetune the weights provided by CATSeg [3]. We find this approach to be beneficial, as the upsampling layers are already trained for segmentation and therefore provide a good initialization to learn fine-grained features necessary for matching. The third upsampling layer is instead trained from scratch.

3.3. Hardware and implementation

We implement Oryon with PyTorch Lightning [1]. We train each model on four Nvidia V100 GPUs and set the batch size to 32. Training a single model for 20 epochs takes about 12 hours in our standard setting. We test on a single GPU of the same model; a complete evaluation on a dataset takes about one hour, including I/O operations and metric computations.

4. Dataset generation

In this section, we describe the process to generate the scene pairs for our training and test datasets. Given an object O in two scenes A and Q, let P^A, P^Q be the point clouds resulting from the unprojection of the two depth maps D^A, D^Q, and let RGB^A, RGB^Q be the respective RGB images. The point clouds P^A, P^Q are then filtered by using the ground-truth mask of A and Q of the object O, thus retaining only the points that belong to the object of interest in the two scenes. The objective is to find the transformation $T_{A\to Q}$ that aligns P^A to P^Q. Let T_A, T_Q be the ground-truth pose of O in A and Q respectively.

In order to generate the poses required for the training process, the following strategy is adopted:

1. Sample a random object O, and a random scene A in which O is present.
2. Consider all the images that contain O that belong to a different scene from A. Sample a random Q from this set.
3. Given T_A and T_Q, compute the relative pose $T_{A\to Q} = T_Q(T_A)^{-1}$.
4. Apply $T_{A\to Q}$ to P_A, thus aligning it to P_Q. Use the Nearest Neighbor algorithm to compute a set of 3D matches between P_A and P_Q. To be considered a match, the Euclidean distance between two points must be ≤ 2 mm.

Table 1. Comparison of the data requirements of Oryon with examples of state-of-the-art methods for generalizable pose estimation. We classify the methods based on: **Input**: the type of input data, typically RGB or RGBD; **Reference**: additional data used to identify the novel object at test time; **Full pose**: whether the method is capable of estimating the 6D pose or is limited to the rotation component; **Object onboarding**: eventual process required at test time to acquire data about the object to pose; **Localization prior**: eventual external modules used to localize the object, typically a segmentation mask or a detector.

Method	Input	Reference	Full pose	Object onboarding	Localization prior
OVE6D [2]	D	3D model	✓	Generates and encodes 4K object templates	Segm. mask
MegaPose [10]	RGBD	3D model	✓	-	Detector
OSOP [20]	RGB	3D model	✓	Generates 90 object templates	-
Gen6D [13]	RGB	Video sequence	✓	SfM and manual cropping of point cloud	-
OnePose [22]	RGB	Video sequence	✓	SfM to retrieve camera viewpoints	Detector
OnePose++ [5]	RGB	Video sequence	✓	SfM to retrieve camera viewpoints	Detector
NOPE [16]	RGB	Single supp. view		-	-
RelPose [27]	RGB	Two or more supp. views		-	-
RelPose++ [12]	RGB	Two or more supp. views	✓	-	Detector
PoseDiffusion [24]	RGB	Two or more supp. views	✓	-	Detector
LatentFusion [17]	RGBD	One or more supp. views	✓	-	Segm. mask
Oryon (ours)	RGBD	Single supp. view	✓	Expression of textual prompt	-

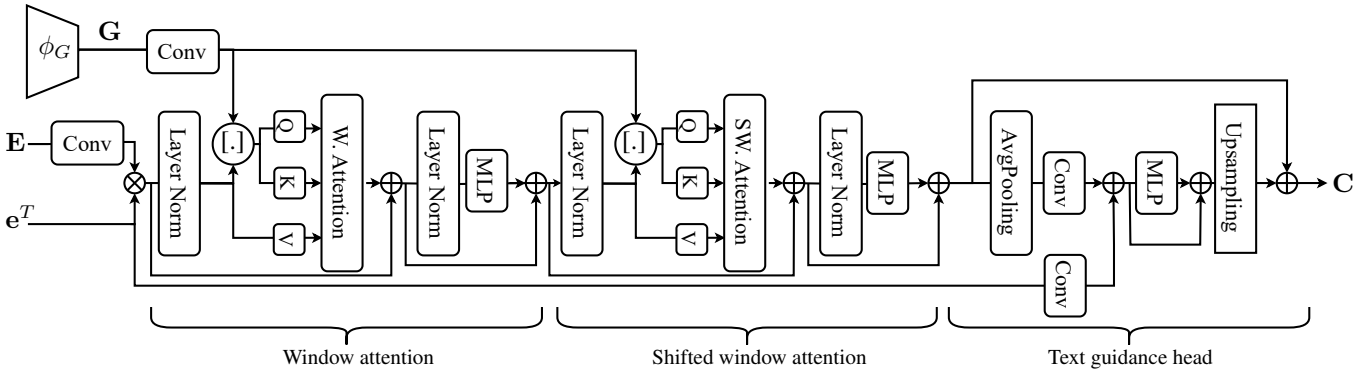


Figure 1. Architecture detail of a layer of the fusion module ϕ_{TV}. **W. Attention**: attention layer that performs self-attention within windows, as in [14]; **SW. Attention**: attention layer that performs self-attention among shifted windows, as in [14]; **Upsampling**: upsampling by interpolation; $\oplus$ denotes feature sum; $\otimes$ denotes cost computation by cosine similarity; [.] denotes feature concatenation.

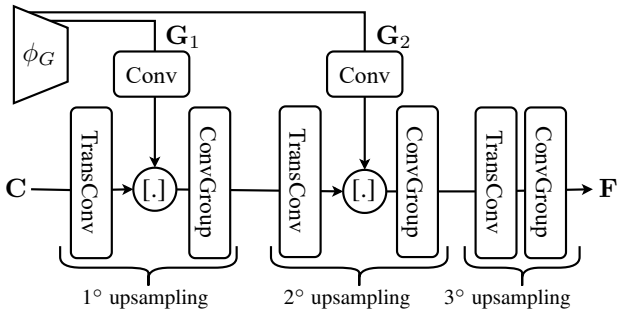


Figure 2. Architecture detail of the decoder ϕ_D. **TransConv**: transposed 2D convolution; **ConvGroup**: group composed by two blocks, each contains a 2D convolution, a group normalization and a ReLU; [.] denotes feature concatenation.

5. Use the intrinsic camera parameters to project the set of matches in 2D, thus obtaining the pixel-level correspondences $\mathbf{x}^A$, $\mathbf{x}^Q$ between RGB^A and RGB^Q.

Note that the relative pose $T_{A\to Q}$ is only used for evaluation, as the training objectives are the ground-truth matches $\mathbf{x}^A$, $\mathbf{x}^Q$. The above process is repeated until the desired number of image pairs is obtained (i.e., 2K for REAL275 and TOYL, 20K for SN6D). Additionally, when generating the training set from SN6D, we reject all scene pairs with fewer than 100 ground-truth matches.

5. Pose metrics details

We adopt the Average Recall (AR) [8] as main pose metric, which is defined as the average of three different metrics (VSD, MSSD, MSPD). Additionally, we report the ADD(S)-0.1d [6] and its definition. In the following definitions, let $\mathbf{O} \in \mathbb{R}^{N\times 3}$ be the point cloud representing the object model, and $\hat{\mathbf{T}}$, $\mathbf{T} \in \mathbb{R}^{3\times 4}$ be the ground-truth and predicted 6D pose respectively. Let also d be the diameter of $\mathbf{O}$.

Visible Surface Discrepancy (VSD) compares two dis-

tance maps, obtained by rendering $\mathbf{O}$ respectively with $\hat{\mathbf{T}}$ and $\mathbf{T}$ on the original depth of the image. The error e_{VSD} is thus obtained as the mean of the distance map error, and the recall θ_{VSD} is defined as the mean of a set of recalls, in which the thresholds varies from 5% to 50% of d. Finally, VSD is defined as the mean of a set of recalls on θ_{VSD}, in which the threshold varies from 0.05 to 0.5. Due to its definition, VSD is agnostic to object symmetry.

Maximum Symmetry-Aware Surface Distance (MSSD). Consider the maximum distance (i.e., between any pair of points) between $\mathbf{O}$ transformed with the ground-truth pose $\hat{\mathbf{T}}$ and $\mathbf{O}$ transformed with the predicted pose $\mathbf{T}$. This error e_{MSSD} is used to compute a set of recalls, in which the thresholds vary from 5% to 50% of d. The average of the recall set is the final metric MSSD. In order to take into account rotation to symmetry, e_{MSSD} is computed for each symmetry axis of $\mathbf{O}$, and the smallest error is selected [8].

Maximum Symmetry-Aware Projection Distance (MSPD) is defined similarly to MSSD. Instead of computing the error in the 3D space, the transformed point clouds are projected in 2D before computing the maximum distance. As for VSD and MSSD, the resulting error e_{MSPD} is used to compute a set of recalls, whose thresholds vary from 5% to 50% of $w/640$, where w is the width of the image. As in the MSSD, e_{MSPD} is computed with all the symmetry sets of $\mathbf{O}$, and the smallest distance is used to compute the recalls.

ADD(S)-0.1d [6] (ADD for short) is quite different from the metrics defined in the BOP challenge. The error e_{ADD} is defined as the mean distance between $\mathbf{O}$ transformed with the ground-truth pose $\hat{\mathbf{T}}$ and $\mathbf{O}$ transformed with the predicted pose $\mathbf{T}$. Instead of computing a mean of recalls, ADD is defined as a single recall of e_{ADD}, where the threshold is defined as 10% of d.

The error measured by ADD is similar to the one of MSSD, but the first is more effective at measuring the success rate at a quite restrictive threshold (i.e., the cases in which the pose is very accurate). On the other hand, MSSD is less sensible to noise as it is defined as a set of recalls, and is less dependent on object model shape as it measure the maximum distance instead of the mean [6]. MSPD shares similar characteristics of MSSD, while VSD is effective at measuring the depth distance, regardless of the object symmetry. For this reasons, we adopt Average Recall (AR) as main metric, and we also report ADD(S)-0.1d due to its wide use in the pose estimation community [21, 25].

6. Prompt details

In this section, we present the textual prompts used for all our tests with REAL275 [23] and TOYL [7], along with some examples of textual metadata from the training set, SN6D [26]. Note that we omit the prompt template used to augment the object descriptions. We adopt the template list used by CLIP [18], which includes 80 templates.

6.1. REAL275 prompts

In Fig. 3, we provide the exhaustive list of prompts used for each object model in REAL275. For each object, we show an example crop obtained from our test set and the textual information used to compose the prompt. We report in black the object name, in green the description used in our default setting, and in red the misleading description used in the ablation study.

6.2. Toyota-Light prompts

In Fig. 4, we present the exhaustive list of prompts used for each object model in the TOYL. For each object, we show an example crop obtained from our test set and the textual information used to compose the prompt. We report in black the object name and in green the description used in our default setting. Note how, with respect to REAL275, the poses of the objects show more variety (e.g., objects tilted on one side or upside-down), as well as more object types.

6.3. ShapeNet6D synsets

In Fig. 5, we showcase some examples of objects in the SN6D [26] dataset, along with the object names and the synsets (i.e., synonym sets) we adopt to build the prompt. For each object, we show an example crop obtained from the training set and the list of synonyms obtained from ShapeNetSem [19]. The first name in the list is the default one, and the subsequent names (if present) are the synonyms. This dataset is completely synthetic and does not depict a realistic scenario, as all the objects are rendered in random poses and on a random background. Nonetheless, SN6D shows a wide variety of poses and objects, along with rich textual information useful for constructing the textual prompts.

7. Additional qualitative results

In Sec. 7.1, we report additional qualitative results on pose estimation on the test datasets. In Sec. 7.2, we show qualitative results of the segmentation mask obtained by Oryon in our best setting and compare them with the ground-truth masks and the ones predicted by OVSeg [11].

7.1. Pose results

In Fig. 6, we show qualitative results of the object poses on REAL275 [23]. Oryon is generally effective at localizing the object of interest, while the main source of errors in the pose is related to the rotation components (see Fig. 6(a,c,f)). In all these cases, the object appears small, and therefore the retrieved matches are coarse-grained. On the other hand, ObjectMatch [4] and SIFT [15] show larger errors, often related to the translation components (Fig. 6(a,b,d) for ObjectMatch, and Fig. 6(e) for SIFT). Note that all methods

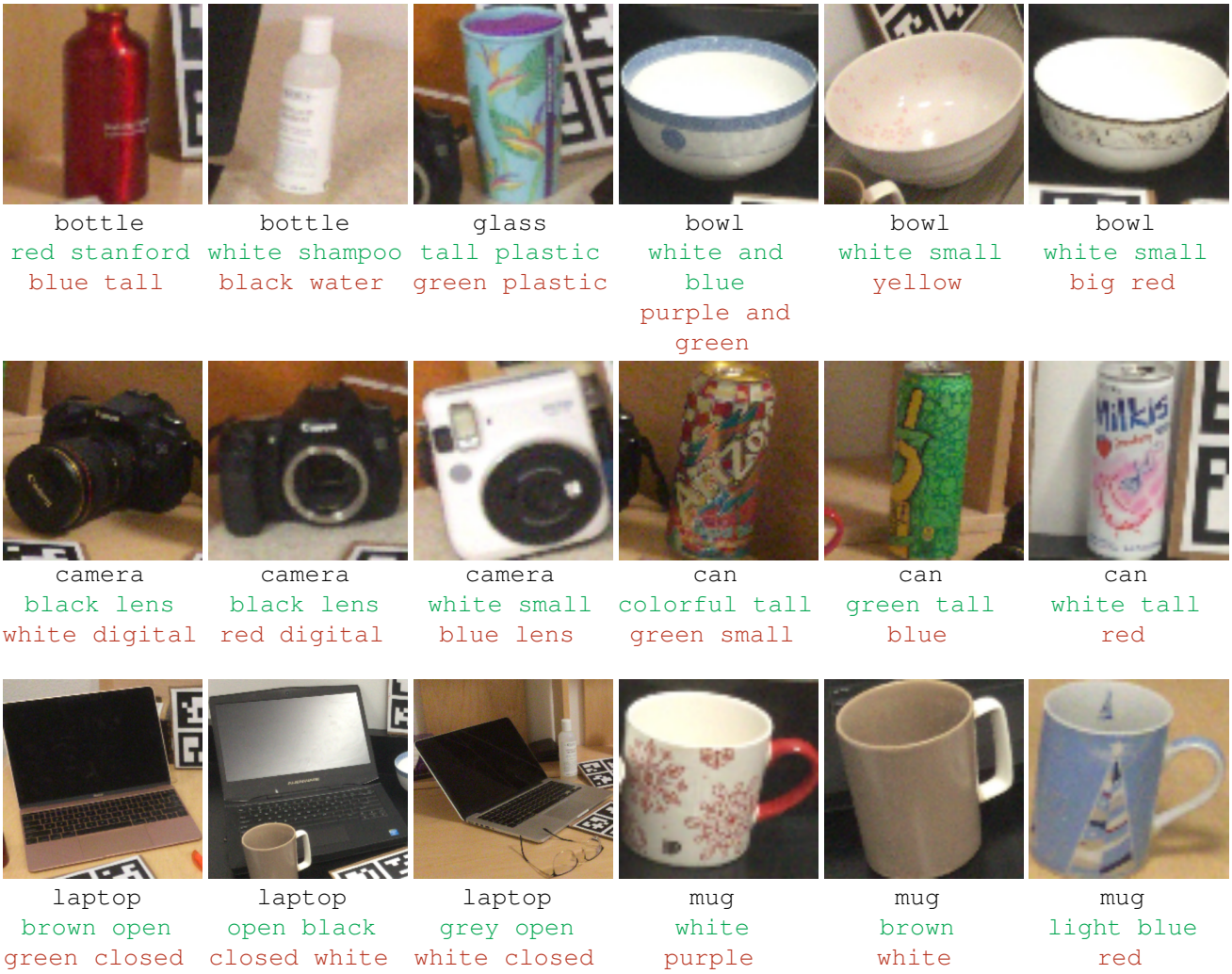


Figure 3. Prompts used in the REAL275 [23] dataset. For each object, we show the object name, the detailed description (i.e., the default one), and the misleading description we adopt in the ablation study. All the objects are cropped for better visualization.

in this visualization use our segmentation mask; therefore, translation errors are due to spurious matches rather than an incorrect mask.

In Fig. 7, we show qualitative results of the object poses on TOYL [7]. As in REAL275, Oryon localization on TOYL is quite accurate. Rotation errors are still present and are related to small objects (see Fig. 7(a, d, f)), which in TOYL are more common due to higher variation in poses compared to REAL275. We observe that ObjectMatch's effectiveness in this dataset is limited: in some cases, large errors cause the projected object to fall outside the image plane (Fig. 7(a, b)), while in other cases, significant translation errors are present (Fig. 7(d, e, f)). As observed in the main results, the high variation in light conditions of TOYL is not tolerated by ObjectMatch, which fails to find accurate matches. On the other hand, SIFT performances are closer to ours. Localization is more effective than ObjectMatch (Fig. 7(a, b, c, d, f)), but rotation errors are still present (Fig. 7(a, b, f)). The scale invariance properties of SIFT make it more effective for this scenario.

7.2. Segmentation results

In Fig. 8, we show some qualitative results of the segmentation masks on REAL275 [23]. On this dataset, Oryon outperforms OVSeg [11] by 10.1 points in mean Intersection-over-Union (mIoU). In Fig. 8, we can observe that the masks predicted by Oryon are generally coarser than the ones predicted by OVSeg. This is caused by the lower resolution we adopt for the segmentation mask (192×192), while OVSeg generates the masks using the original image resolution (480×640). There are cases in which this results in a clearly better performance for OVSeg (Fig. 8(a, d)). However, compared to Oryon, false positives are much more frequent in OVSeg: the object is completely missed in

Figure 4. Prompts used in the TOYL [7] dataset. For each object, we show the object name and the detailed description (i.e., the default one). All the objects are cropped for better visualization.

Fig. 8(c, e, f), while in Fig. 8(b) the whole table is selected in the anchor scene, and the wrong object is segmented in the query one.

On the other hand, Oryon is less prone to false negatives: most errors are instead due to the inclusion of background in the mask (Fig. 8(b)) or segmenting only a part of the object (Fig. 8(d)). Both these errors can still provide a sufficient number of valid matches to perform registration, while a false positive results in an automatic failure in the pose estimation task. Note the particular case of Fig. 8(f), in which two mugs are present in the anchor image. This causes an ambiguity, and results in a separate mask for each mug. Similarly, in the query image of Fig. 8(c), the wrong mug is segmented.

In Fig. 9, we show some qualitative results of the segmentation masks on TOYL [7]. On this dataset, Oryon is outperformed by OVSeg [11] by 7.4 mIoU. Compared to REAL275, TOYL does not show additional objects other than the object of interest; therefore it is an easier dataset for the segmentation task. We observe that both Oryon and OVSeg can correctly locate the objects of interest, and all the errors in the examples are due to imprecision in the segmentation mask, instead of false positives or false negatives. Similarly to REAL275, OVSeg masks are generally more accurate than the ones predicted by Oryon. The higher resolution of OVSeg is an important advantage in this dataset, as the object can appear very small due to perspective (see Fig. 9(a, b, c)). Most of the errors in Oryon are due to partial object segmentation (Fig. 9(b, d, f)) and background inclusion in the mask (see Fig. 9(a, e)).

7.3. Feature visualization

In Fig. 10, we present some examples of the visualization of the feature distance in the feature maps $\mathbf{F}^A$, $\mathbf{F}^Q$ obtained by Oryon. For each image pair, we first sample a random reference point from the ground-truth segmentation mask on the anchor image (shown in green). We then compute the cosine similarity between the feature at the reference

Figure 5. Sample objects from the SN6D [26] dataset. For each object, we show the object default name, followed by its synonyms. Note that not all objects have an associated synonym set (e.g., the rifle, skateboard, and table objects). All the objects are cropped for better visualization, and padding is added when necessary.

point and all the other features, both on the anchor and on the query image. In this way, we obtain a distance measure for each pixel, which is normalized across the pair and mapped to the RGB space. For each example, we show three versions of the feature maps obtained with the prompts we adopted in the ablation study in the main paper (i.e., the standard prompt, the prompt with only the object name, and the prompt with a misleading description).

When the standard prompt is adopted (first row), we can observe a sharp difference between the features of the background and the ones on the objects. Note also how in Fig. 10(a), the most similar points to the reference one on the query image are on the bowl border, as the reference point itself. When the object description is removed (second row), we can observe noticeable changes in Fig. 10(a), in which the background appears more noisy, and in Fig. 10(c), where a set of similar features appears on top of the camera in the query image. Instead, in Fig. 10(b), the difference with the distances of the original prompt is less relevant. As we reported in the ablation study, this alternative prompt causes a small drop of 2.2 in AR and of 3.1 in mIoU. Finally, in the last row, a misleading description is used in the prompt. This causes a clear drop in feature quality in Fig. 10(a, c): in the first example, the object outline is still visible due to similarity in the borders of the bowl, and in the second example only spurious similar features are present. Instead, the laptop object in Fig. 10(b) is less affected. Note that in the ablation study, the misleading prompt causes a drop of 6.8 and 10.1 in AR and mIoU, respectively. These results suggest that the use of a misleading or more generic prompt can impact Oryon performances differently depending on the object of interest.

References

[1] Pytorch lightning documentation. https://lightning.ai/docs/pytorch/stable/. Last access: 21/11/2023. 2

[2] D. Cai, J. Heikkila, and E. Rahtu. OVE6D: Object Viewpoint Encoding for Depth-based 6D Object Pose Estimation. In *CVPR*, 2022. 1, 3

[3] S. Cho, H. Shin, S. Hong, S. An, S. Lee, A. Arnab, P. H. Seo, and S. Kim. Cat-seg: Cost aggregation for open-vocabulary semantic segmentation. In *arXiv:2303.11797*, 2023. 2

[4] C. Gümeli, A. Dai, and M. Nießner. ObjectMatch: Robust Registration using Canonical Object Correspondences. In *CVPR*, 2023. 4, 8, 9

[5] X. He, J. Sun, Y. Wang, D. Huang, H. Bao, and X. Zhou. OnePose++: Keypoint-Free One-Shot Object Pose Estimation without CAD Models. In *NeurIPS*, 2022. 1, 3

[6] T. Hodan, J. Matas, and S. Obdrzalek. On evaluation of 6D object pose estimation. In *ECCV*, 2016. 3, 4

[7] T. Hodan, F. Michel, E. Brachmann, W. Kehl, A. GlentBuch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, et al. Bop:

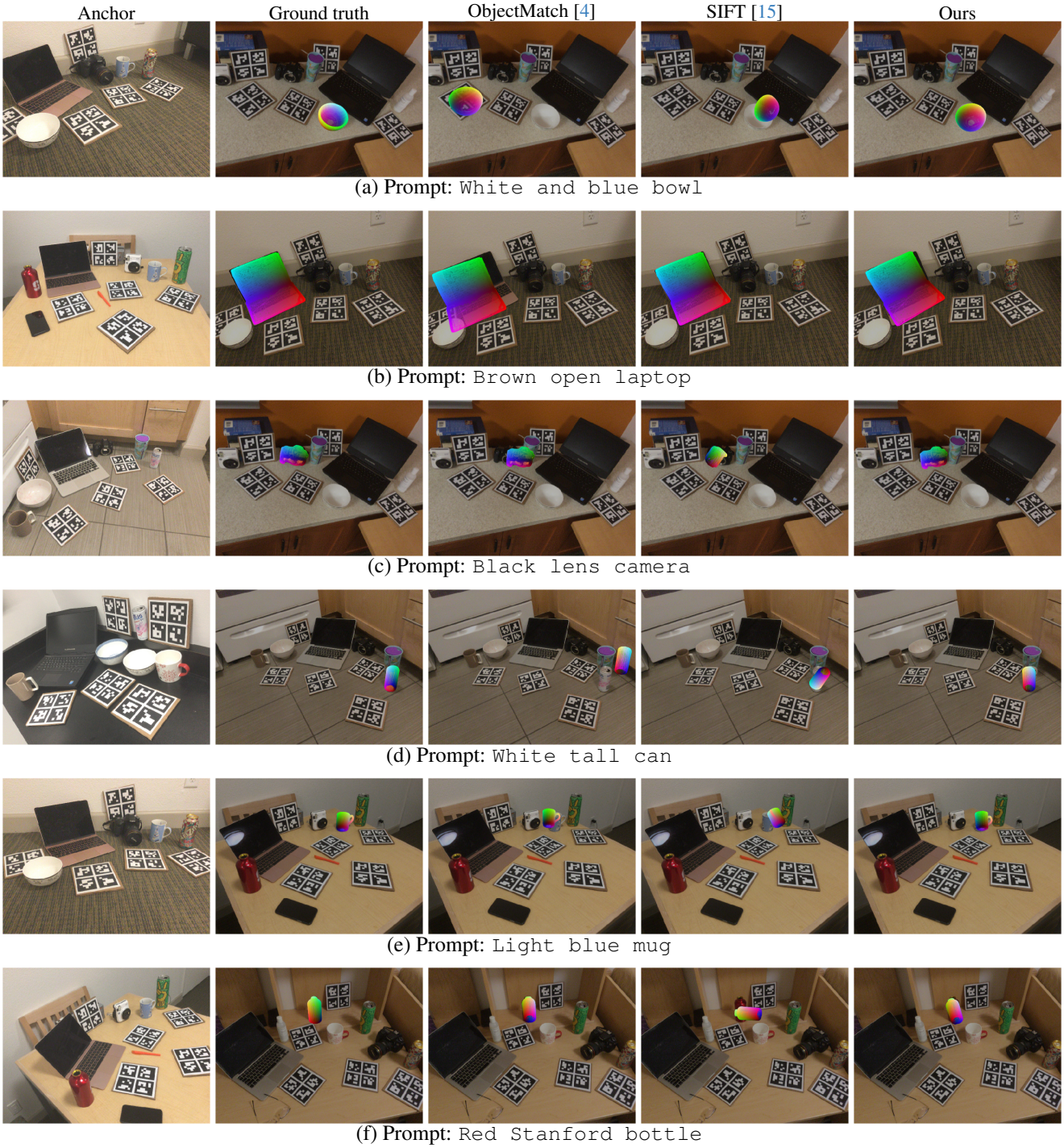


Figure 6. Examples of qualitative pose results from the REAL275 [23] dataset. All the results use the segmentation mask predicted by Oryon. We show the object model colored by mapping its 3D coordinates to the RGB space. Query images are darkened to highlight the object poses.

Benchmark for 6D object pose estimation. In *ECCV*, 2018. 1, 4, 5, 6, 9, 11

[8] T. Hodan, M. Sundermeyer, B. Drost, Y. Labbe, E. Brachmann, F. Michel, C. Rother, and J. Matas. BOP challenge 2020 on 6D object localization. In *ECCV*, 2020. 3, 4

[9] Glenn Jocher. Yolov5 by ultralytics, 2020. 1

[10] Y. Labbé, L. Manuelli, A. Mousavian, S. Tyree, S. Birchfield, J. Tremblay, J. Carpentier, M. Aubry, D. Fox, and J. Sivic.

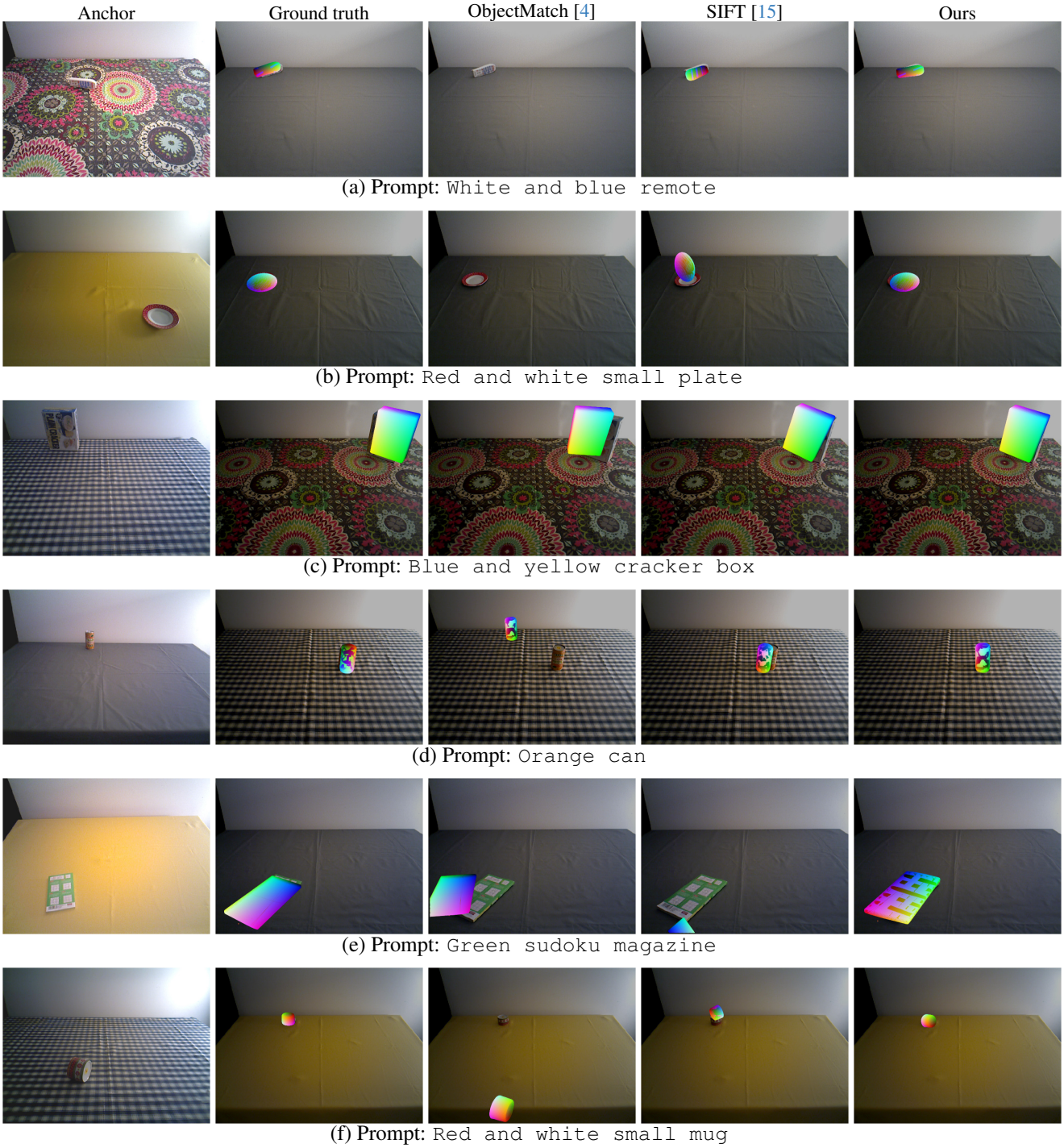
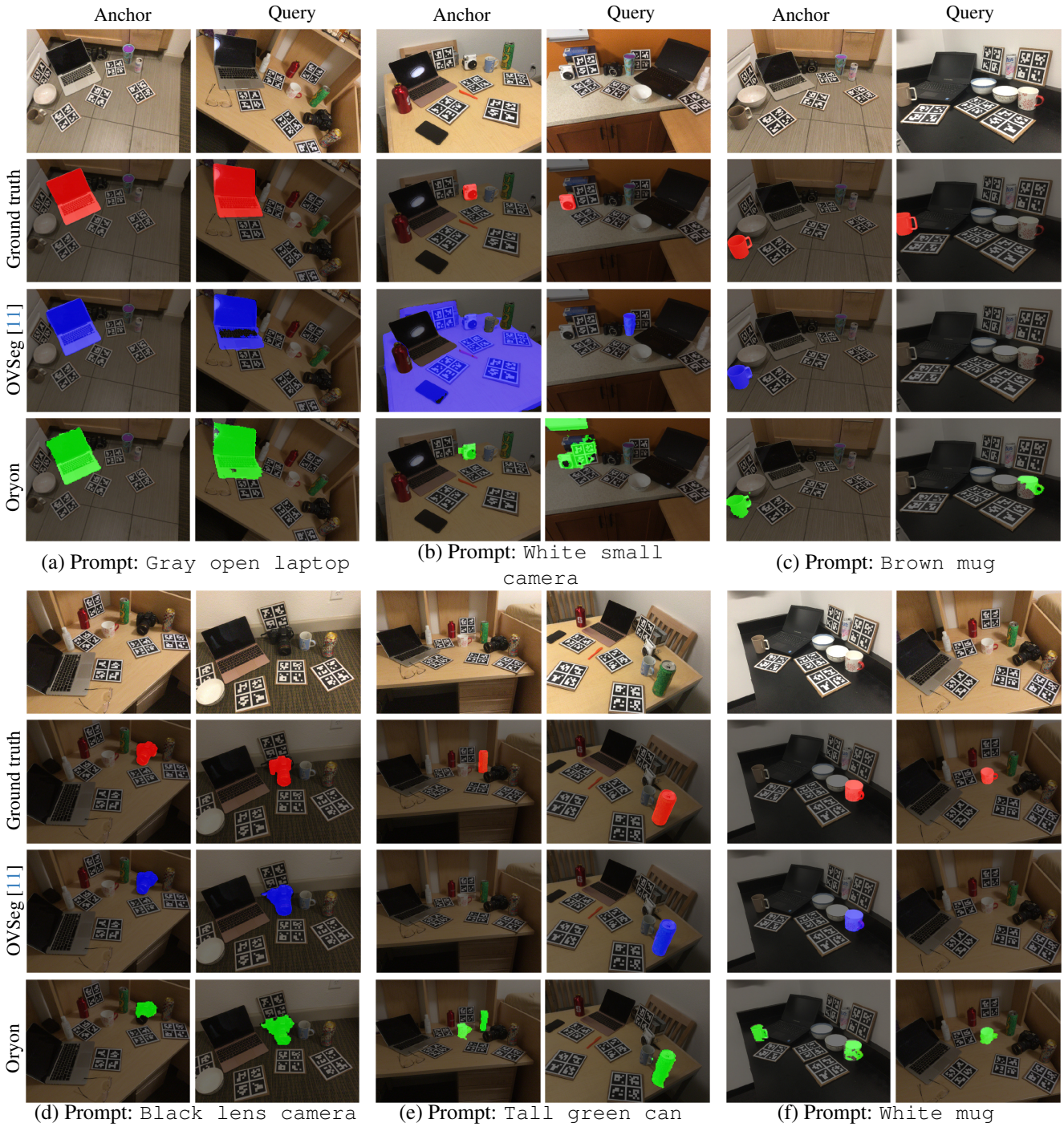


Figure 7. Examples of qualitative pose results from the TOYL [7] dataset. All the results use the segmentation mask predicted by Oryon. We show the object model colored by mapping its 3D coordinates to the RGB space. Query images are darkened to highlight the object poses.

Megapose: 6D pose estimation of novel objects via render & compare. In *CoRL*, 2022. 1, 3

[11] F. Liang, B. Wu, X. Dai, K. Li, Y. Zhao, H. Zhang, P. Zhang, P. Vajda, and D. Marculescu. Open-vocabulary semantic segmentation with mask-adapted clip. In *CVPR*, 2023. 4, 5, 6, 10, 11

[12] A. Lin, J. Y. Zhang, D. Ramanan, and S. Tulsiani. Relpose++: Recovering 6d poses from sparse-view observa-



(a) Prompt: `Gray open laptop` (b) Prompt: `White small camera` (c) Prompt: `Brown mug`

(d) Prompt: `Black lens camera` (e) Prompt: `Tall green can` (f) Prompt: `White mug`

Figure 8. Examples of qualitative segmentation results from the REAL275 [23] dataset. Images are darkened to highlight the masks.

tions. In *arXiv:2305.04926*, 2023. 1, 3

[13] 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 *ECCV*, 2022. 1, 3

[14] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 3

[15] D. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999. 4, 8, 9

[16] V. Nguyen, T. Groueix, Y. Hu, M. Salzmann, and V. Lepetit. Nope: Novel object pose estimation from a single image. In *arXiv:2303.13612*, 2023. 1, 3

[17] K. Park, A. Mousavian, Y. Xiang, and D. Fox. Latentfusion: End-to-end differentiable reconstruction and rendering

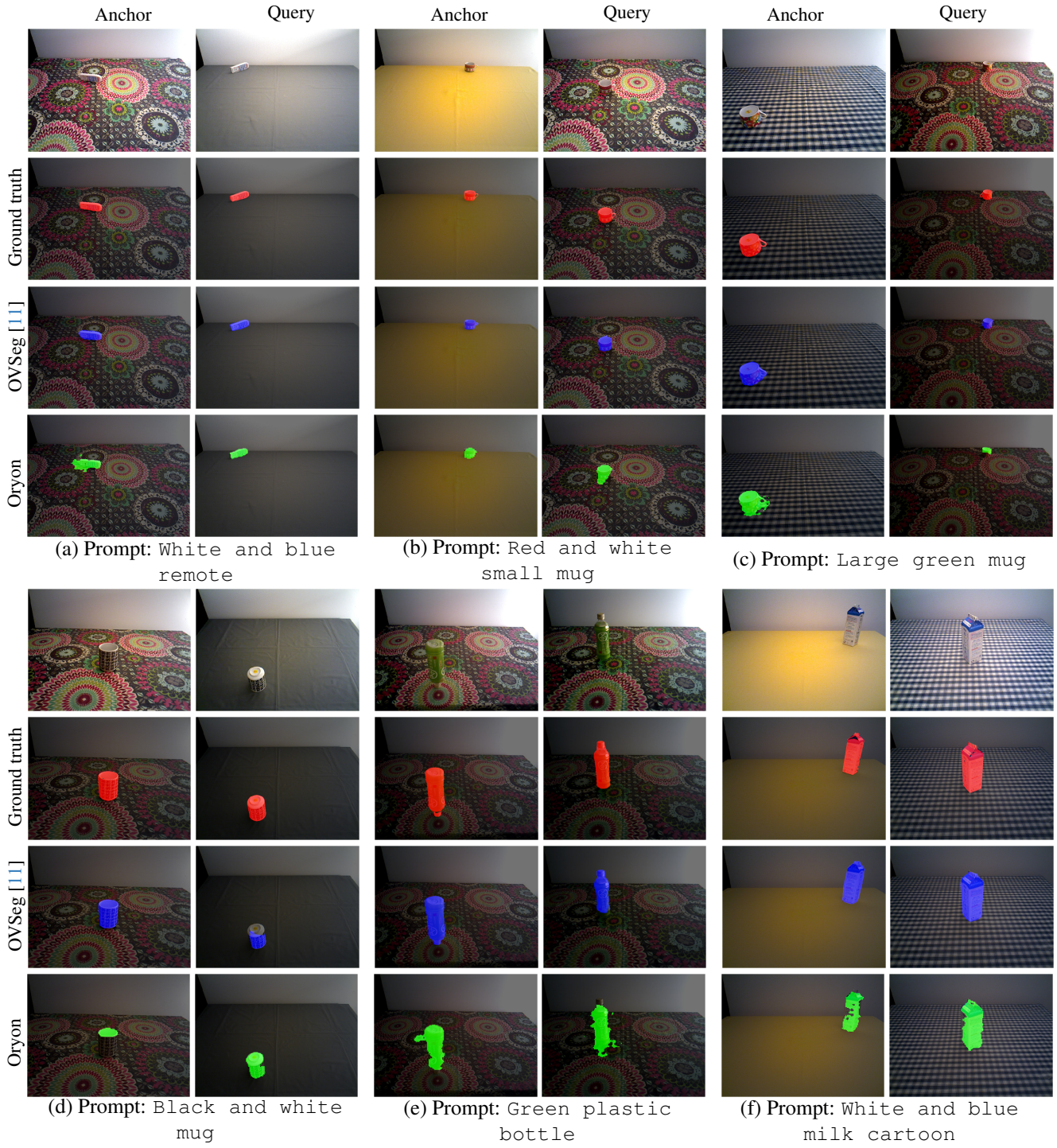


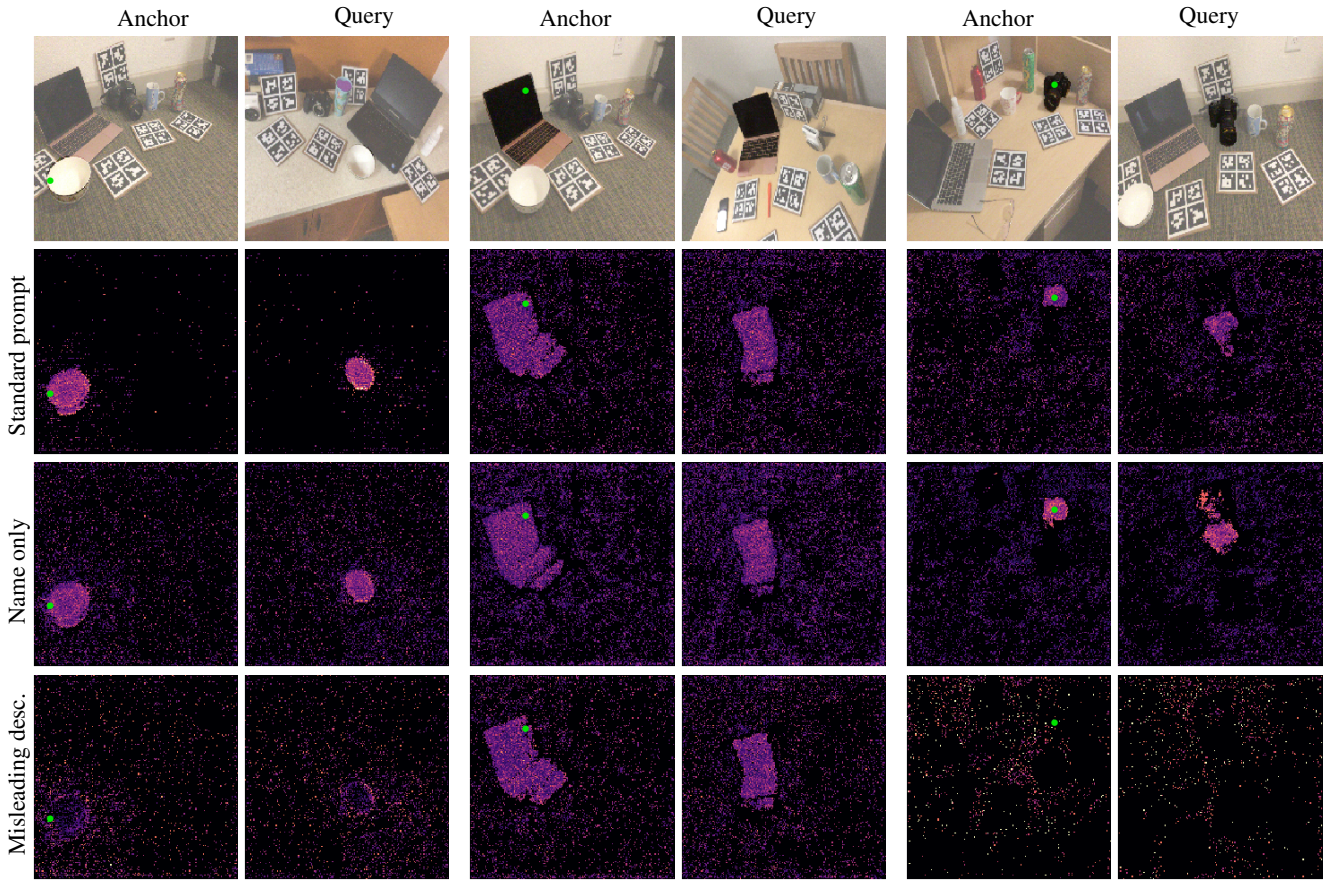
Figure 9. Examples of qualitative segmentation results from the TOYL [7] dataset. Images are darkened to highlight the masks.

for unseen object pose estimation. In *CVPR*, 2020. 1, 3

[18] A. Radford, J. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 4

[19] M. Savva, A. Chang, and Hanrahan P. Semantically-Enriched 3D Models for Common-sense Knowledge. In *CVPR*, 2015. 4

[20] I. Shugurov, F. Li, B. Busam, and S. Ilic. OSOP: a multi-stage one shot object pose estimation framework. In *CVPR*,



(a) Prompt: `White small bowl` (b) Prompt: `Open brown laptop` (c) Prompt: `Black lens camera`

Figure 10. Visualization of the feature distance in $\mathbf{F}^A$, $\mathbf{F}^Q$ with respect to a reference point on the object (denoted as • in the anchor image) on REAL275 [23]. For each sample, we test three different prompts we adopted in the ablation study: **Standard prompt**: the standard prompt we adopt, composed of a description and the object name; **Name only**: the standard prompt without the description; **Misleading desc.**: the standard description is replaced by a misleading one (see Fig. 3). The RGB images are modified to highlight the object of interest.

2022. 1, 3

[21] Y. Su, M. Saleh, T. Fetzer, J. Rambach, N. Navab, B. Busam, D. Stricker, and F. Tombari. ZebraPose: Coarse to Fine Surface Encoding for 6DoF Object Pose Estimation. In *CVPR*, 2022. 4

[22] 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. 1, 3

[23] 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. 1, 4, 5, 8, 10, 12

[24] J. Wang, C. Rupprecht, and D. Novotny. Posediffusion: Solving pose estimation via diffusion-aided bundle adjustment. In *ICCV*, 2023. 1, 3

[25] C. Wu, L. Chen, S. Wang, H. Yang, and J. Jiang. Geometric-aware Dense Matching Network for 6D Pose Estimation of Objects from RGB-D Images. In *Pattern Recognition*, 2023. 4

[26] H. Yisheng, W. Yao, F. Haoqiang, C. Qifeng, and S. Jian. Fs6d: Few-shot 6D pose estimation of novel objects. In *CVPR*, 2022. 1, 4, 7

[27] J. Zhang, D. Ramanan, and S. Tulsiani. Relpose: Predicting probabilistic relative rotation for single objects in the wild. In *ECCV*, 2022. 1, 3