Zero-Shot Object Pose Estimation

Roberto Pellerito, Alessandro Burzio, Diego Machain, Lorenzo Piglia
rpellerito@ethz.ch, aburzio@ethz.ch, dmachain@ethz.ch, lpiglia@ethz.ch

[1]

Abstract

Zero-shot object pose estimation is a challenging task in computer vision that involves estimating the pose of an object, including its position and orientation, from an image or a sequence of images, without any prior training on that specific object. This is an important problem to solve for mixed reality applications, as it enables a variety of human-machine interaction applications, such as overlaying real objects with virtual labels or instructions.

In this work, we present a zero-shot object pose estimation pipeline extending the work of [12] that leverages self-supervised deep learning methods [7] to produce pose estimates for everyday objects, even those that have not been seen during training. Our approach extends the work of OnePose [12], using visual localization from RGB video frames to produce 3D bounding boxes for objects present in the scene following [9], by incorporating Deep Spectral Methods [7], which provides an unsupervised zero-shot segmentation and 2D detection pipeline based on a learned approach.

We deploy our full pipeline on the Microsoft HoloLens 2 as an interactive application, enabling real-time pose estimation of objects in the user's environment. Our approach demonstrates the potential for using self-supervised deep learning methods to enable robust and accurate object pose estimation in a zero-shot setting.

A. Introduction

Object pose estimation, or the ability to determine the position and orientation of an object in 3D space, is a fundamental task in computer vision. It is particularly important in robotics, where it allows a robotic arm to grasp and manipulate objects, and in mixed reality applications, where it enables the overlay of virtual labels or instructions on real objects.

Traditionally, state-of-the-art methods for object pose estimation have relied on either a full CAD model [16] of

[1]All the authors are under ETH Zürich administration and within The Computer Vision and Geometry Group

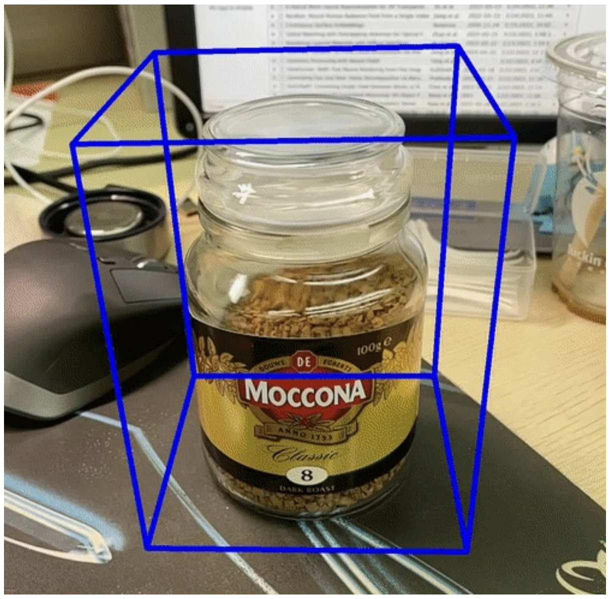

Figure 1. A pose estimation and 3D bounding box reprojection for a common household object

the object (instance-level object pose estimation) or the assumption that objects belong to predefined categories [**?**] (category-level object pose estimation). However, these approaches are not scalable and can easily fail in the real world, as it is not possible to obtain a CAD model of every object, and it is common to encounter objects that fall outside of the distributions of the predefined categories. Additionally, these approaches require the use supervision on deep neural networks that need to be fine-tuned or retrained when the objects and setting change, which can be time-consuming and resource-intensive.

In an effort to overcome these limitations, Sun et al. [12] have proposed the use of sparse 3D representations of objects for object pose estimation. While this approach allows for the automatic generation of 3D sparse representations from a video scan and camera poses, it still requires 3D bounding box annotations for each frame. In this project, we aim to remove this requirement by utilizing an unsupervised object detector, as presented in Deep Spectral Methods [7], and by estimating a 3D bounding box of the object obtained via [9] to enable zero-shot object pose estimation. In other words, our goal is to estimate the pose of novel objects of any category that have not been seen during training.

Our project relies on the code published in [12], [7] and [9], that we adapted and modified to suit our needs.

B. Contributions

The main contribution of this project is the development of a method for zero-shot object pose estimation that does not require any manual annotations during the object scan process as in OnePose [12]. Our pipeline makes exclusively use of RGB frames and camera poses to perform object pose estimation. As in OnePose, the resulting pose is expressed in the camera reference frame. This is accomplished by using the object DSM as an object detector on individual frames $\{\mathbf{I}^i\}$ extracted from the scan video. We then use the resulting bounding boxes $\{\mathbf{B}^i_{2D}\}$ and camera poses $\{\xi^i_W\}$ to estimate the ellipsoidal occupancy of the object with [9] and create a 3D bounding box $\mathbf{B}_{3D}$ as the parallelepiped encapsulating the ellipsoid. To meet the requirements of the OnePose pipeline, camera poses should be aligned with the object's origin. This is achieved by shifting all the camera poses, after estimating the 3D bounding box, such that the center of the 3D bounding box lies in the world origin. Since the 2D bounding boxes are obtained in unsupervised fashion and the 3D bounding box is estimated using purely geometrical methods, our object pose estimation can be applied to any instance of any object class in a zero-shot way.

C. Related Works

Instance-level object pose estimation. Instance-level object pose estimation refers to the problem of estimating the 6DoF pose of a specific, known object instance within an image. This is a challenging problem due to the wide variety of possible object poses, the variability in object appearance, and the presence of occlusions and clutter.

One approach to instance-level object pose estimation is to use a model-based method, which involves building a 3D CAD model of the object and fitting it to the 2D image data. Modern approaches use a learning-based model to predict object pose directly from image data, given the CAD model.

There have been a number of successful approaches to instance-level object pose estimation, such as PoseCNN [15] and DeepIM [6] that make use of image features to estimate the 6D pose of the object. Another approach Real-Time 6D Object Tracking Using RGB-D Data [8] uses instead a combination of images, optical flow estimation and Kalman filtering to track objects in real-time.

Category-level object pose estimation. There are some prior works that achieve CAD-model-free pose estimation. Neural Object Fitting [3] proposes an encoding category-level prior performed by a Variational Auto Encoder (VAE). However, these methods are limited by the image synthesizing networks, so the efficiency and accuracy make them not suitable for augmented reality applications.

Another work related is Objectron [1], where the authors propose an approach that fits the pixel coordinates of projected box corners for each category, so it's a data-driven model that requires a huge amount of annotated training data. The limits of this work lies on the amount of categories the model is trained on, because it is only category-specific, as well as the cost of obtaining the dataset and training the model.

D. Methods

Zero-Shot object pose estimation. Zero-shot pose estimation refers to the task in which a model is able to recognize and estimate the 6DoF pose ξ_o of an object from any class or instance without direct supervision during training or manual annotations. As shown in 2 and 3 we achieve Zero-shot pose estimation in two steps, that we refer to as back-end and front-end:

1. In the back-end of our pipeline 2 we use images to detect the object with [7], giving a collection of 2D bounding boxes for each frame. The 2D detections $\{\mathbf{B}^i_{2D}\}$ and camera poses $\{\xi^i_W\}$ are then used to estimate a 3D bounding box $\mathbf{B}_{3D}$ around the object with [9]. Camera poses and 2D features $\{\mathbf{F}^i\}$ obtained from the images are then used to construct the 3D sparse representation of the object $\mathbf{P}$, which is then filtered to keep points only inside the 3D bounding box. Correspondence graphs $\{\mathcal{G}^j\}$ are finally built between the 2D features and 3D points from the reconstructed model.

2. The front-end 3 makes use of 2D features extracted from novel images, to build directly 2D-3D matches to the SfM model. 2D object detections from [7] are used again to filter out features that do not belong to the object. Finally the PnP algorithm uses the 2D-3D correspondences to estimate the final object pose.

Category-Free 2D bounding boxes from images. The first block of our pipeline consists of using an off-the-shelf unsupervised object detector [7]. DSM builds on top of DINO [2] by using traditional spectral decomposition for images on the outputs of a Vision Transformer, to get their eigenvectors and eigenvalues. The extracted eigenvectors are then reshaped into images and used to localize objects within the image, independently of any specific category.

Standard object detection with DSM involves detecting a single object using always the eigenvector corresponding to the second smallest eigenvalue 4. This gives speed to the algorithm (instead of dynamically search for the best eigenvalue) at the cost of a lower precision on the detections.

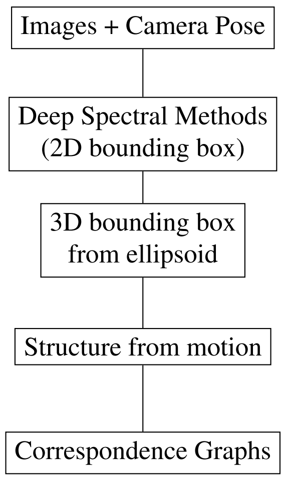
Images + Camera Pose
Deep Spectral Methods (2D bounding box)
3D bounding box from ellipsoid
Structure from motion
Correspondence Graphs

Figure 2. Back-end of pose estimation pipeline.

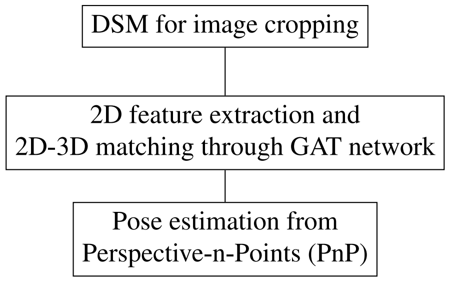
DSM for image cropping
2D feature extraction and 2D-3D matching through GAT network
Pose estimation from Perspective-n-Points (PnP)

Figure 3. Front-end of pose estimation pipeline.

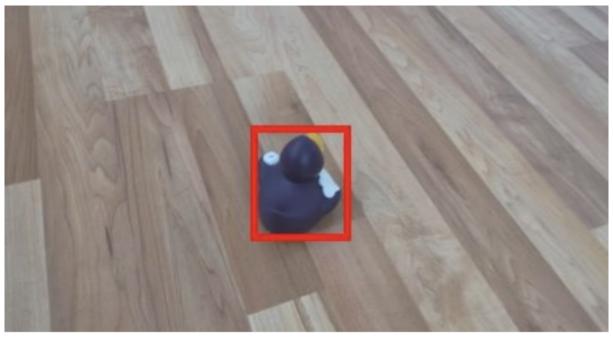

Figure 4. 2D detection over a single frame of a video recorded from Hololens2

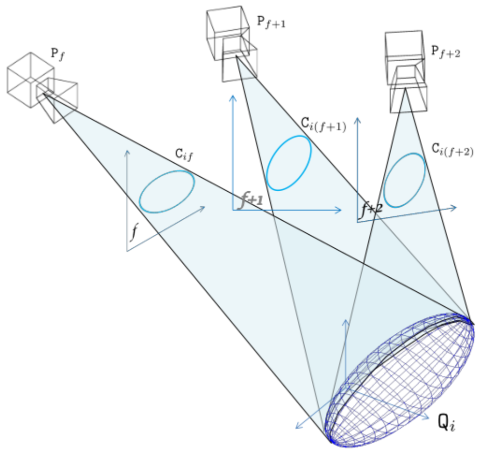

Figure 5. Example of a set of conics and the 3D quadric they represent

This is one of the bigger limitations of our work, as only one object can be detected at a time, sometimes leading to bad reconstructions or unstable pose estimates, and possible failures in cluttered scenes due to other objects being detected.

Since the bounding box is computed as the smallest rectangle containing all non zero values of the reshaped eigenvector, noise in the eigenvector can cause bad detections, much larger than the object itself. For this reason we also employed a filter that performs 2D Gaussian-like curve fitting to make the eigenvectors smoother and focus on the object of interest.

The purpose of 2D detections is twofold: they are used on one hand to create a crop of the images for the 2D feature extraction, and on the other hand to create a 3D bounding box around the object.

3D bounding box from 2D bounding boxes. 2D detections acquired in the previous step are used to create a 3D bounding box around the object. To get the 3D bounding box from the 2D detections the problem is reformulated as the estimation of an ellipsoid in 3D given a set of 2D ellipses (fitted from the 2D bounding boxes) in multiple views [9].

The estimation of the ellipsoid can be performed in closed form in the dual space from a set of at least 3 views, by using the relation between the dual of the 3D ellipsoid and its dual conics projections 5.

The process of detecting two-dimensional bounding boxes, even after filtering, is not always accurate, which can lead to problems when attempting to estimate the three-dimensional ellipsoid directly (I.E. using the relation between the dual space of the ellipse and the dual space of the ellipsoid). This can result in solutions that are nearly degenerate ellipsoids or even other quadric shapes, such as hyperboloids.

To overcome this issue, multiple images are used and a constraint is imposed such that the solution must be an ellipsoid by directly estimating its Rotation Matrix and its three semi-axes. To find the optimal solution in this case, since now the problem is no longer convex, a non-linear optimization algorithm is employed, which involves iteratively improving an initial guess of the solution, until it converges.

Finally, once the 3D ellipsoid is estimated, we save

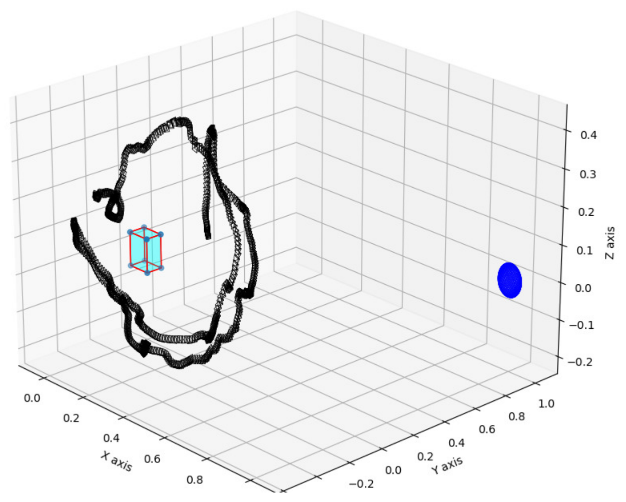

Figure 6. Example of a detected ellipsoid (in blue) and the corresponding parralelepiped shifted and rotated back in the center of the coordinate frame along with the camera poses

the coordinates of its corresponding parallelepiped (the 3D bounding box) and we shift and rotate all the camera poses such that the estimated bounding box lies in the origin of the coordinate frame. This step is very important since the HoloLens poses are estimated based on an arbitrary origin while [12] relies on the assumption that the object lies in the origin 6.

Building a 3D representation of the object. OnePose makes use of COLMAP to perform Structure from Motion (SfM) reconstruction of a sparse 3D point cloud of the object. To remove background clutter and retain only what corresponds to our object in the scene, we make use of Deep Spectral Methods [7] to detect the object and obtain a 2D bounding box surrounding the object, then we crop the original images and save the crops and intrinsics parameters. 2D Image Features are extracted using SuperPoint [4] from the modified images and matched with SuperGlue [10]. We then feed the series of cropped images, intrinsics, keypoint matches and poses to COLMAP [11] for triangulation and refinement.

Pose estimation from RGB images. Once the model of the object is obtained, at run-time the pipeline estimates the 6D pose of the object in the camera frame from a given image of the environment.

The pose estimation is done as in OnePose using 2D keypoints and the 3D object point cloud. The query images are cropped using [7], and 2D keypoints are again extracted with [4]. The 2D keypoints are matched with the 3D point cloud from the back-end, using multiple Graph Attention layers (GAT).

The Graph Attention layers select the best descriptor of a 3D point from the correspondence graph. This aggregation operation is fundamental for the 2D-3D matching, where a dual-softmax operator extracts with the 2D-3D descriptors the permutation matrix M_{3D}, which represents the 2D-3D matches. Finally with M_{3D}, the object pose in the camera coordinate can be computed by the Perspective-n-Point (PnP) algorithm with RANSAC.

E. Results

E.1. Data acquisition

We evaluated the performance of our zero-shot pose estimation method on samples from the OnePose dataset, which contains images of different objects without specific categories. The objects in the dataset have a wide range of poses, including rotations and translations in all three dimensions.

This dataset was captured using Apple ARtoolkit as described in [12], together with annotated 3D bounding boxes. Since the objective is to perform pose estimation without user annotations, in this work we used only the colored images and poses. However, these poses were all written with respect to the object reference frame, so we manually applied a shift to all camera poses to verify our approach is fully independent from all the OnePose assumptions.

The data captured from the HoloLens device is by default independent from the OnePose assumptions.

E.2. Evaluations

We compare the performance of our method on the OnePose datasets. We focus on evaluation of our pipeline in three different scenarios: easy, medium and hard datasets, where the difficulty of the dataset is given by the amount of clutter in the background. Additional objects or patterns in the background can create wrong detections, corrupting the 2D and 3D bounding box construction.

We first test the ability of our pipeline to estimate the correct 3d bounding box starting from unsupervised 2D detection with [7]. Moreover, our objective is to show that shifting poses such that the object is not centered, yields a correct estimation of the 3D bounding box. For this evaluation we compute IoU of our 3D bounding box with respect to the ground truth, annotated bounding box after applying a constant shift of 1 meter in the x and y directions, as well as a $\pi/2$ rotation along the y axis.

IoU	easy	medium	hard
Shifted poses	0.667	0.485	0.472

We additionally compare the position error of the estimated pose of the object with a simple $L2$ loss

$$e_p = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \quad (1)$$

and the rotation error by first calculating the difference rotation matrix between ground truth and estimated rotation

$$R_{diff} = (R_{gt}, R_{est}^T) \tag{2}$$

and then computing the rotation difference as the angle of the difference rotation, retrieved throught the trace of R_{diff}:

$$\theta_{diff} = \arccos(\frac{tr(R_{diff}) - 1}{2}) \tag{3}$$

Image 7 and 8 shows translational and rotational error respectively of three increasingly harder datasets. We compared the results using the same datasets and Vanilla OnePose aganist our pipeline. The poses are detected with filtering of 2D bounding boxes and as the plot shows the clutter of the environment does not influence the prediction error. On average the translational error is near $10^{-2}m$. As expected, the rotational error is very high: this is due to the fact that in OnePose the default orientation of an object is hand-labelled in an arbitrary way, whereas in this work it's obtained automatically through unsupervised methods.

We also compared both 2D bounding boxes from DSM with and without the Gaussian filter for three different scenes from the OnePose validation dataset. The ground truth 2D bounding boxes used to perform the IOU are obtained from Onepose dataset itself, by projecting the ground truth 3D bounding box on to a 2D bounding box on each frame of the scene.

Mean IoU	easy	medium	hard
DSM without filter	0.49	**0.40**	0.28
DSM with filter	**0.52**	0.38	**0.38**

Table 1. Comparison of IOU for bounding boxes computed from DSM with and without filter.

Table 1 shows that the filter can improve the IoU when the scene is really cluttered (hard case), but it does not have a significant impact for the easier cases.

F. Deploying on HoloLens

One of the goals of the project was also to deploy the pipeline on the Microsoft Hololens 2 device. Since the pipeline relies heavily on large neural networks that require big computations and benefit from CUDA architecture, we decided to build the application in a client-server fashion, with the HoloLens acting as a client and sending images and poses to a server pc where the heavy processing happens, and receiving directly the pose and size of the detected objects. To enable the client-server communication, we employ the Unity-Robotics-Hub package: it allowed us to easily define custom messages for passing data between the HoloLens and the pc, with a easy to use ROS-like interface.

F.1. On the HoloLens: Unity application

On the HoloLens side, we deployed a Unity application with the main goal of capturing and sending images+poses to the server and displaying to the user the 3D bounding box resulting from the pipeline in the augmented reality world.

The HoloLens is constantly streaming images with the corresponding camera pose to the server though the Unity-ROS TCP interface. This is achieved through a custom defined message containing the raw image as an array of unsigned integers together with image details. At first, the image acquisition was done using the LocatableCamera API. However, we found this API to be very slow and cumbersome to use, with a maximum framerate of about 1 Hz with a 720p image. We then moved to the WebcamTexture API which was much easier to use and resulted in a more reasonable 4-5 Hz transfer rate at a 896x504 pixels resolution. Further optimizations might be done using different APIs or by sending compressed images instead of raw pixel data. The HoloLens camera intrinsics were obtained by calibrating the camera with opencv and through COLMAP, which resulted in very similar estimates.

A floating button follows the user, and can be pressed to start and stop the recording for the object Structure from Motion scan. To display the bounding boxes, we employed a slightly modified version of the BoundBoxes package from the Unity Asset Store. A custom message was also defined to transfer bounding boxes information, containing 6D pose and size (height, width, length). Embedded in the bounding box message received from the server is a unique object identifier: if the identifier is new, a new bounding box object is created and placed in the scene. Otherwise, if the identifier has already been previously received, the old bounding box is moved to the new object position.

F.2. Server Side: ROS

The two separate steps for the pipeline lend themselves very well to be operated through ROS as an interface. The start / stop scan button on the Unity side acts as a switch on the server side between the two steps. During scanning time, images are saved locally to disk. Whenever the *end scan* signal is received, we run the Structure from Motion part of OnePose, modified as previously discussed to generate the 3D sparse representation and correspondences graph of the object in an unsupervised fashion.

Whenever the server application is not in scan mode, the image subscriber callback executes the standard inference step of the OnePose pipeline, which returns the relative pose of the detected object with respect to the camera T_{CO}. By using the pose information from the HoloLens T_{WC}, we can calculate the actual position and orientation of the object T_{WO} in the Unity reference frame so that it can be published over the ROS topic and displayed inside the HoloLens application.

$$T_{WO} = T_{WC} \times T_{CO} \quad (4)$$

F.3. The full pipeline

Unfortunately, we were not able to get the full pipeline working on the HoloLens device. In particular, we were not able to create a 3D reconstruction of the object with COLMAP using the images and poses captured directly from the HoloLens, as very few points are triangulated (and do not create a coherent object). The main suspect lies in the Hololens poses, since we were able to verify that every other component of the Structure from Motion step works correctly: feature extraction and matching through superpoint + superglue produces good results, and without embedding the poses in COLMAP a meaningful object can be reconstructed. This excludes the possibility of blurry images coming from the Hololens or not enough features present on the objects being the culprit.

Interestingly, the main difference with the OnePose dataset is that poses as acquired from Unity are in the Unity reference frame, which is a y-up left-handed reference frame. Nevertheless, the Unity-Robotics-Hub package provides methods to automatically convert to a right-handed frame, which were employed in the Unity application before the SfM step. To test whether the poses provided by the Hololens in this way could be noisy, we used Umeyama's algorithm [13] to align the real Hololens poses to the camera poses estimated by COLMAP and verify they are coherent. As a baseline, we also employed the same method on images + poses from the OnePose dataset. The alignment procedure produces we get the following results:

Average position error after alignment	
Hololens	2.34 $\pm$ 1.55 cm
OnePose dataset	0.25 $\pm$ 0.172 cm

The error with Hololens poses is an order of magnitude higher, but still within what could be considered acceptable noise (especially considering the Onepose dataset's poses are further refined through bundle adjustment, contrary to the raw HoloLens poses). We are therefore still uncertain as to what is causing the failure in the reconstruction phase.

F.4. User Studies

Given that we were not able to get the full application to work, we also could not conduct proper user studies. Nevertheless, while acquiring data from the HoloLens to conduct our tests, we noticed the following things:

- The app itself is very straightforward to use, as interaction is limited to the click of one button to switch between the scanning and pose estimation phases. We considered also using a voice command to switch between the two phases, but proper user studies would be required to determine which mode would be best. Another small improvement would include adding a visual cue whenever the object scan and reconstruction are in progress, so that the user is aware of what is happening in the backend.

- The biggest flaw in the user experience lies in the scanning phase: it is very challenging to get close detail of objects with a head mounted device, while also moving slowly enough to not acquire blurry pictures that would degrade the quality of both reconstruction and pose estimation. Furthermore, it is sometimes impossible to get a full 360 degrees view of the object without moving it, e.g. when the object is placed on a table with one of the sides leaning against a wall.

G. Future work

In this section we explore further possible improvements to the proposed approach:

- OnePose++
- Parallel Reconstruction and Pose Estimation
- Improvements to the HoloLens app

G.1. OnePose++

A natural extension to this work would be to substitute the original OnePose with its follow-up work OnePose++. In particular, in OnePose++ [5] the authors move from a sparse 3D object reconstruction to a dense one, which results in higher reconstruction detail. This enables more robust pose estimation, especially in the case of objects lacking texture where the sparse 3D model construction fails to capture enough points to produce consistent pose estimates.

G.2. Parallel Reconstruction and Pose Estimation

One big limitation of this work is that, even though the object scan and capture are done automatically through self-supervised methods in a zero-shot fashion, a full scan of the object is still required before the actual pose estimation step. To overcome this, a possible approach would be to bootstrap an initial 3D representation of the object from the first recorded frames, then use the partial 3D model for pose estimation (assuming subsequent images to be taken close in time and space) while simultaneously adding newly triangulated points and features to the sparse 3D model.

This would require a major rework of the whole infrastructure, especially given that COLMAP cannot incrementally add new images to existing models but requires recomputing everything from scratch, which is computationally heavy and could not work in real time. A possible solution would be to adapt the work of Wen et al. [14].

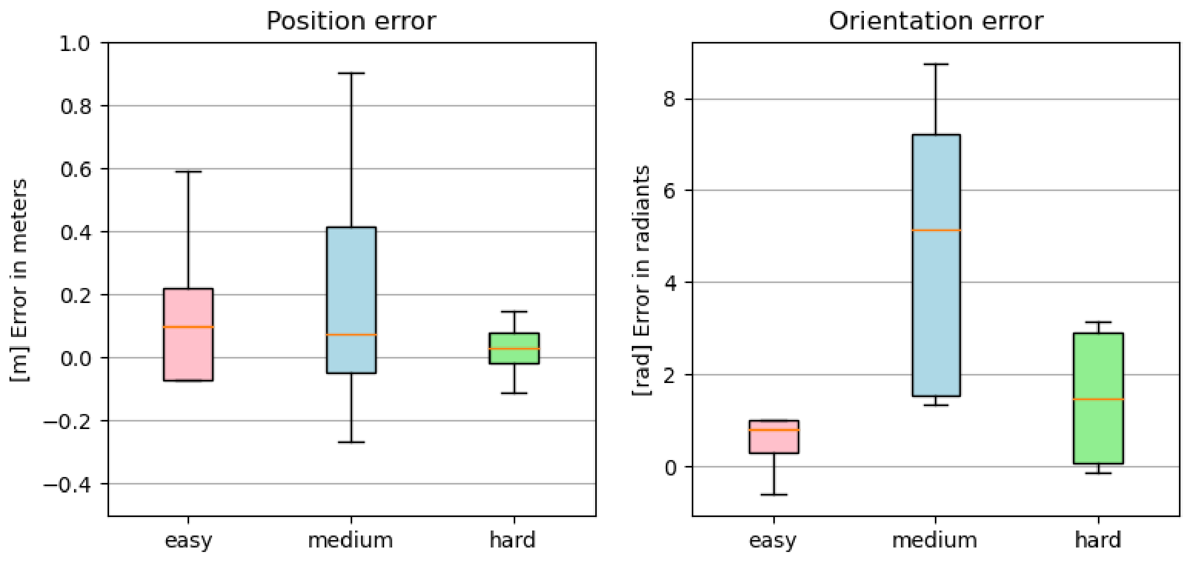

Figure 7. Bar plot of pose error estimated by our pipeline for three datasets of the estimated objects displacement with respect to a ground truth pose of the object obtained from ARtoolkit. Thin lines show confidence interval at 3σ over the value of the mean of the considered error.

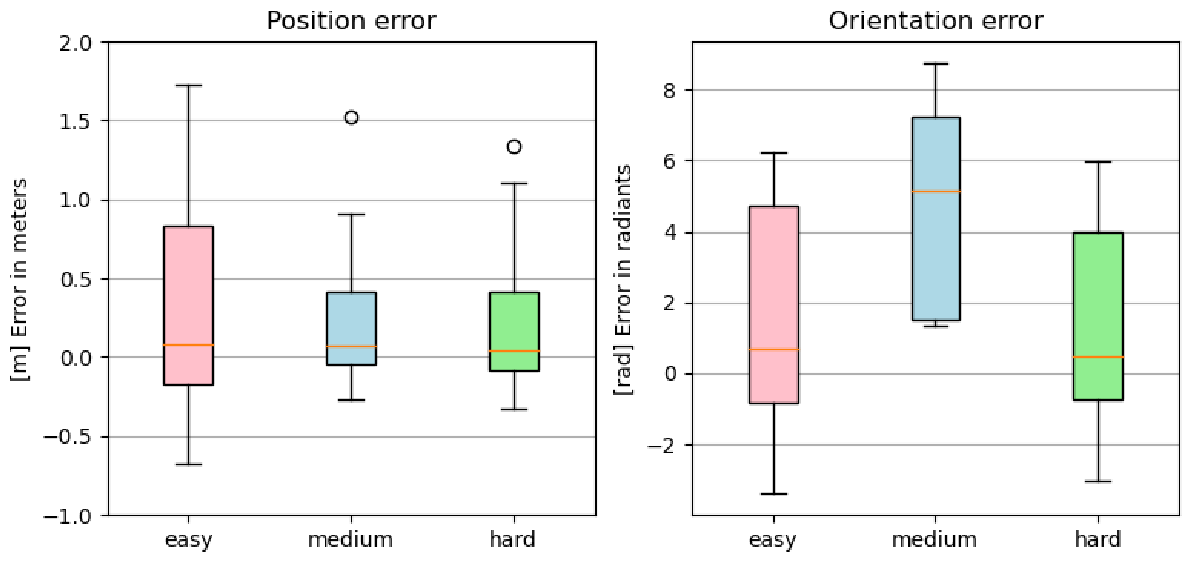

Figure 8. Bar plot of pose error estimate by vanilla OnePose for three datasets of the estimated objects displacement with respect to a ground truth pose of the object obtained from ARtoolkit. Thin lines show confidence interval at 3σ over the value of the mean of the considered error.

G.3. Improvements to the HoloLens app

The app could be extended in a number of ways, e.g. by letting the user visualize the reconstructed 3D point cloud of the object, or properly handling multiple objects detections by doing further work on the object detector. The issue of difficult scanning could also be addressed by changing the scanning approach from a video of a still object to an in-hand scanning, or in general an interactive approach where the user would be able to move the object during the scan to cover all sides.

References

[1] Adel Ahmadyan, Liangkai Zhang, Artsiom Ablavatski, Jianing Wei, and Matthias Grundmann. Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021. 2

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021. 2

[3] Dong Z. Song J. Geiger A. Hilliges O. Chen, X. Category level object pose estimation via neural analysis-by-synthesis. *European Conference on Computer Vision*, Aug 2020. 2

[4] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description, 2017. 4

[5] Xingyi He, Jiaming Sun, Yuang Wang, Di Huang, Hujun Bao, and Xiaowei Zhou. Onepose++: Keypoint-free one-shot object pose estimation without CAD models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 6

[6] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. DeepIM: Deep iterative matching for 6d pose estimation. *International Journal of Computer Vision*, 128(3):657–678, nov 2019. 2

[7] Luke Melas-Kyriazi, Christian Rupprecht, Iro Laina, and Andrea Vedaldi. Deep spectral methods: A surprisingly strong baseline for unsupervised semantic segmentation and localization, 2022. 1, 2, 4

[8] Nicola A. Piga, Yuriy Onyshchuk, Giulia Pasquale, Ugo Pattacini, and Lorenzo Natale. ROFT: Real-time optical flow-aided 6d object pose and velocity tracking. *IEEE Robotics and Automation Letters*, 7(1):159–166, jan 2022. 2

[9] Cosimo Rubino, Marco Crocco, and Alessio Del Bue. 3d object localization from multi-view image detections. In *Pattern Analysis and Machine Intelligence (TPAMI), 2017 IEEE Transactions on*, 2017. 1, 2, 3

[10] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks, 2019. 4

[11] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4

[12] Jiaming Sun, Zihao Wang, Siyu Zhang, Xingyi He, Hongcheng Zhao, Guofeng Zhang, and Xiaowei Zhou. Onepose: One-shot object pose estimation without cad models, 2022. 1, 2, 4

[13] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04):376–380, 1991. 6

[14] Bowen Wen and Kostas Bekris. Bundletrack: 6d pose tracking for novel objects without instance or category-level 3d models, 2021. 6

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

[16] Yan Xu, Kwan-Yee Lin, Guofeng Zhang, Xiaogang Wang, and Hongsheng Li. Rnnpose: Recurrent 6-dof object pose refinement with robust correspondence field estimation and pose optimization, 2022. 1