Underwater 6D Pose Estimation

Elena Govi [12], Paola Ardon [4], Ivan Crotti [1], Giorgia Franchini [1], Eric Pairet [4], Tobia Poppi [1], Eloy Roura [4], Davide Sapienza [12], Micaela Verucchi [3], Marko Bertogna [13]

[1]Università degli studi di Modena e Reggio Emilia, [2]Università di Parma, [3]Hipert S.r.l., [4]Technology Innovation Institute

Underwater 6D Pose Estimation-State of the art

- The '6d pose estimation' problem consists of both **detection** and **localization** (also orientation) of an object in a scene, only by using cameras.
- It is a crucial task for several computer vision applications, such as **autonomous object picking** in the Industry 4.0 era, just to mention one.
- While 2D object detectors reach great results, the 6d pose problem reveals to be more complex.
- The best performing methods that operated in RGB images (without the depth sensor) at the beginning were methods relying on local or global *gradient-based image features*. Nowadays, they have been outperformed by trainable *convolutional neural networks*.

Dataset

Since this work is on development, only a first-object pose-estimation is considered in this poster. However, this project should be inserted in a bigger one: **underwater robot manipulation**. For this reason, an underwater simulated dataset is created, starting from a *ply* model.

- The chosen object for first experiments is a hotstab, the correspondent *ply* model is represented in figure 1.
- Secondly, the object has been created with a 3D printer and real data have been collected.
- The RGB camera used for this data acquisition is the Real Sense D435. The depth is available but has been used only for refinement.

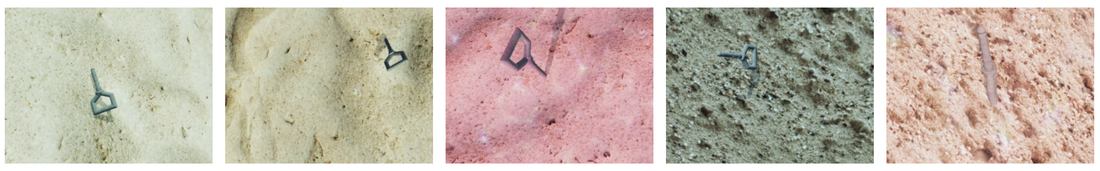

Figure 1. Examples of simulated underwater images with hotstab object

Pipeline

The first phase of pose estimation is about 2D detection, where one valid object detector should be chosen. Consequently, a 6D pose estimation network receives as input the cropped image and the class which the object belongs to. It is supposed to give as output the rotation and translation matrices. Our real-time 6D-pose estimation pipeline is described as follows:

1. **2D Detection**, with Yolo v4 [Bochkovskiy et al. 2020]
2. **Rotation and Translation estimation**, with Augmented Autoencoder [Sundermeyer et al. 2019]

YoloV4

Usually, object detectors are developed in two parts: a backbone, which is pre-trained on ImageNet, and a head, where each class and each bounding box are predicted. The chosen Object detector for our project is YoloV4. First, the reason for this choice lies on its efficiency in real time applications. Yolov4 consists of:

- Backbone: CSPDarknet53 [Bochkovskiy et al. 2020]
- Head: YOLOv3 [Redmon and Farhadi 2018]

Pose representation

- The Rotation $\mathbf{R}$ is represented by a 3×3 matrix
- The Translation $\mathbf{t}$ is represented by a 3×1 vector
- A pose of a 3D object is represented by the 4×4 matrix $\mathbf{P} = [\mathbf{R}, \mathbf{t}; \mathbf{0}, 1]$
- Matrix $\mathbf{P}$ transforms a 3D point x_m in the model coordinate system to a 3D point x_c in the camera coordinate system: $\mathbf{P}[x_m; \mathbf{1}] = \mathbf{R}x_m + \mathbf{t} = [x_c; \mathbf{1}]$

Results

Videos recorded at the Nemi Lake (RM) show real-time great results when the object is not partially occluded:

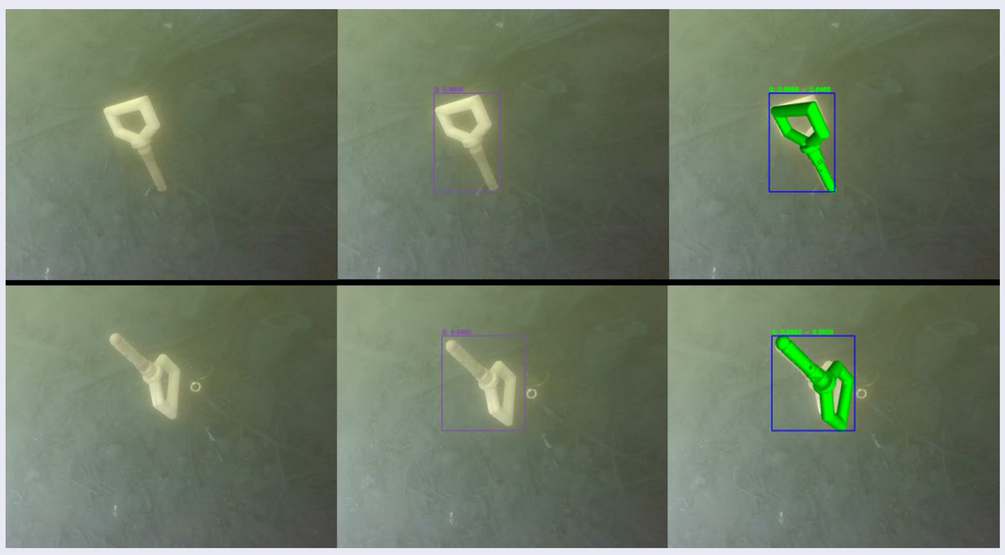

Figure 2. Real-time underwater pose estimation: 2D detection on the centre, AAE estimation on the left.

	YOLO inf.	YOLO post-proc.	AAE proc.pose	AAE drawing t	TOT
t_min	18 ms	29 ms	6 ms	45 ms	98 ms
t_max	18 ms	44 ms	7 ms	50 ms	119 ms
t_avg	18 ms	34 ms	4 ms	48 ms	104 ms
FPS_{avg}	/	/	/	/	9,6

Table 1. Time-table of YOLOv4+AAE training. Inpute size: 1280x720, GPU: 3070

Next steps: metrics for quantitative analysis [Hodan et al. 2018]

- Compute the *error* between $\overline{\mathbf{P}}$ (ground-truth) and $\hat{\mathbf{P}}$ (estimated pose) with $e_{ADD} = avg_{x \in \mathbb{M}} \|\hat{\mathbf{P}}x - \overline{\mathbf{P}}x\|$ or others (err_{VSD}, err_{ADI}, ...)
- Define a criterion of correctness with a threshold θ: $err < \theta$;
- Compute accuracy, recall, precision and $AUC = \int_0^1 recall(err) derr$

Augmented Autoencoder

- Augmented Autoencoder (AAE) applies a random augmentation f_{aug} to input x and reconstructs the original image (figure 3).
- An encoder-decoder training reconstructs the original input.
- The per-sample loss is $l_2 = \sum_{i \in D} \|x_i - \hat{x}_i\|_2$.

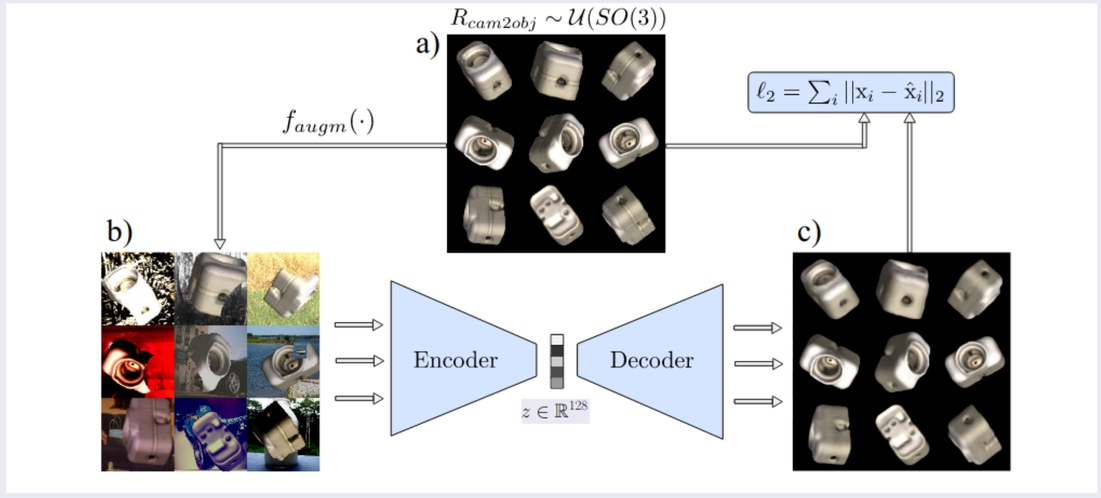

Figure 3. Training strategy: a) reconstruction target batch x of uniformly sampled SO(3) object views; b) geometric and color augmented input; c) reconstruction $\hat{x}$ after 40000 iterations

- The **augmentation phase** could be personalized with background images, artificial occlusion, light, brightness, blurred effects and other available options (figure 4).
- After **training** (figure 6), a *codebook* is created by generating a **latent representation** $z \in \mathbb{R}^{128}$ of each possible object view, and its correspondent **P** matrix (figure 5).
- At **test time**, first the object is detected and cropped. Secondly, the encoder gives its latent space features. Then, cosine similarity is computed between the input latent representation code $z_{test} \in \mathbb{R}^{128}$ and all codes $z_i \in \mathbb{R}^{128}$ from the codebook:

$$cos_i = \frac{z_i z_{test}}{\|z_i\| \|z_{test}\|}$$

Figure 4. Different Data Augmentation with VOC dataset and underwater background images.

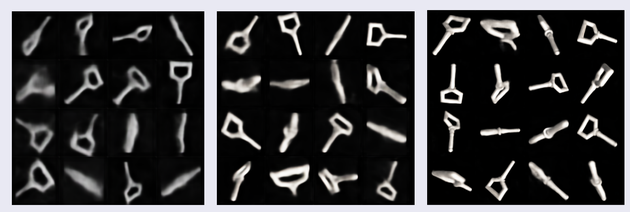

Figure 5. Images reconstructed respectively at iteration 5000, 10000, 50000 of the training

The highest similarity is chosen and the corresponding rotation matrix from the codebook is returned as 3D object orientation.

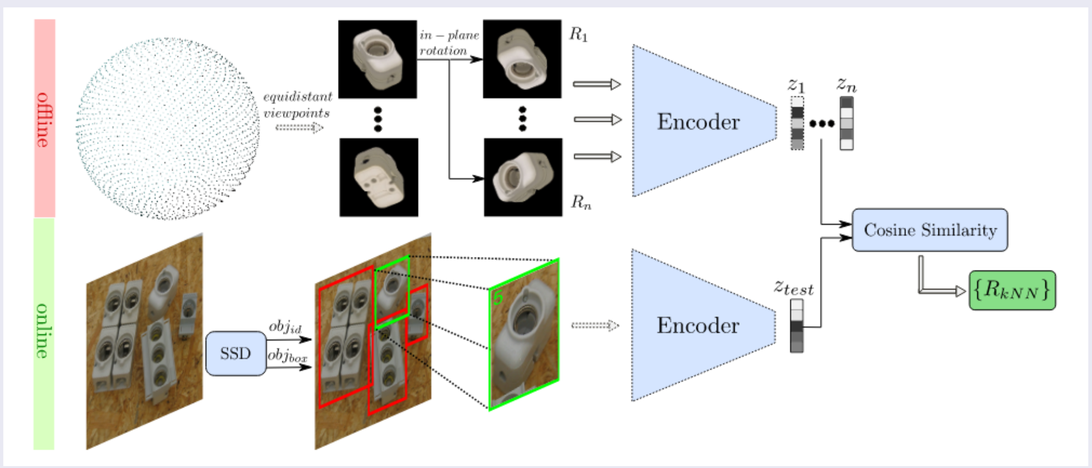

Figure 6. Offline: codebook creation. Online: inference phase.

Future Purposes

To sum up, a real time pose estimator has been created. Our first purpose is to compute metrics in order to try and compare this method with others. Secondly, this project aims to generalize the methods to different objects and scenarios. Some challenging open issues should be studied and solved:

- Multiple objects
- Partially occluded objects

References

- Bochkovskiy, A., C. Wang, and H. M. Liao (2020). "YOLOv4: Optimal Speed and Accuracy of Object Detection". In: *CoRR*.
- Sundermeyer, M., Z. Marton, M. Durner, M. Brucker, and R. Triebel (2019). "Implicit 3D Orientation Learning for 6D Object Detection from RGB Images". In: *CoRR*.
- Hodan, T., E. Brachmann, W. Kehl, A. G. Buch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, C. Sahin, F. Manhardt, F. Tombari, T. Kim, J. Matas, and C. Rother (2018). "BOP: Benchmark for 6D Object Pose Estimation". In: *CoRR*.
- Redmon, J. and A. Farhadi (2018). "YOLOv3: An Incremental Improvement". In: *CoRR*.