Supplementary Material: 3D-Aware Ellipse Prediction for Object-Based Camera Pose Estimation

Matthieu Zins Gilles Simon Marie-Odile Berger
Inria, Université de Lorraine, LORIA, CNRS
matthieu.zins@inria.fr gilles.simon@loria.fr marie-odile.berger@inria.fr

1. Introduction

In this supplementary material, we first give additional details about how we implemented and trained our models. Then, more illustrations of our initial experiment for ellipse prediction on synthetic data are provided. Additional results for our real case scenario are also shown. Finally, we present some results obtained on a purely virtual dataset that we created. We also highlight that our method is designed for fast and model-free camera relocalization (without any refinement).

2. Training procedure

Regarding the object detection part, we used Faster R-CNN R50-FPN (ResNet+FPN backbone) from [7], pre-trained for several epochs on COCO dataset [4] (*train2017*). We fine-tuned it on our objects using a relatively limited number of images, around a hundred, depending on the experiment. Our ellipse prediction network uses a pre-trained VGG-19 as backbone network. For our experiments on real data, we trained one network per object, for approximately 2000 epochs. Again, only a reasonable number of images were used, depending on the experiment (around 200). We performed online data augmentation between each epoch, so that new images are given to the network throughout the training. We used PyTorch [5] for both networks and for the data augmentation.

3. Single-object ellipse prediction

We give here more details related to section 4.5 of the main paper, about the initial experiment for predicting the ellipsoid projection on synthetic data. This experiment was mainly used during the development of our network architecture and is focused on a single object.

We rendered images of a single object by placing the camera on a half-sphere around it. The camera is always held upright and pointing towards the object. The left part of Figure 2 shows the camera positions used for training and testing. We used different objects models taken from the YCB Benchmarks [1] (*cracker box*) or from LINEMOD [2] (*duck*) (see Fig. 1). Synthetic rendering was done with VTK [6] and the background images were randomly taken from COCO [4].

We also evaluated how this ellipse prediction behaves on much different camera positions, especially more distant cameras, and without any retraining. We have varied the half-sphere radius for positioning the test cameras between $50\,\mathrm{cm}$ (the initial one) and $2\,\mathrm{m}$ (see right part of Fig. 2). Regardless of the camera distance, the image passed to the network is always the square crop around the object, rescaled to meet the network requirements. Results are shown in Figures 3 and 4. The obtained IoU scores slightly drop as the camera moves farther away, but they remain quite satisfactory (always above 80%).

4. Real case application

We show more visual results obtained on our real case application (section 6.5 of the main paper) in Figure 5. The predicted ellipse (green) is inferred from the square crop (green). This crop is obtained by enforcing a square aspect ratio to the detection box (orange), provided by the object detection network. The object ellipsoid is projected in the image with the camera ground truth pose (blue). As explained in the main paper, the camera position computation is based on cones alignment, which means that its accuracy is directly linked to the alignment between the predicted ellipse and the projection of the object ellipsoid. We recall that predicting this ellipsoid projection is the goal of our ellipse prediction network.

5. Virtual Scene: different trajectories

In this last section, we show some results obtained on a purely virtual dataset that we generated. We decided to use a synthetic dataset to be able to generate new camera trajectories and evaluate our coarse pose estimation on very different viewpoints, which is not possible with the existing datasets (e.g. in WatchPose, only two different trajectories are available, *near* and *far*, and only one object is visible

Figure 1. Top row: Synthetic images used for training. Bottom row: Test images with the predicted (green) and the ground truth ellipse (red).

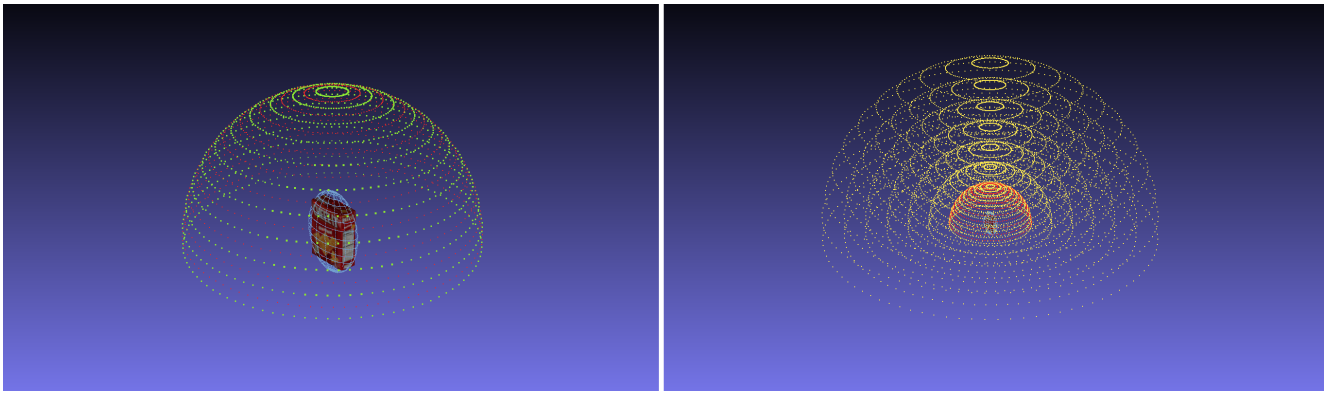

Figure 2. Left: Camera positions used for training (in red) and for validation (in green). Right: More distant cameras used for tests (in yellow).

which does not allow us to recover the full pose). It represents a minimalist scene of ten objects placed on a table. Again, the objects were taken from the YCB Benchmarks and the rendering was done with Blender. We generated three camera trajectories used for training and rendered a total of 192 images (Fig. 6). For each trajectory, the camera is looking at a different fixed point randomly placed on the table. We then applied the strategy presented in section 5.1 of our main paper. In details, we reconstructed an ellipsoid for each object from manual object annotations in five images (Fig. 7). We then obtained objects annotations (BB and ellipse) in all the training images and used them to fine-tune Faster R-CNN and train the ellipse prediction network for each object.

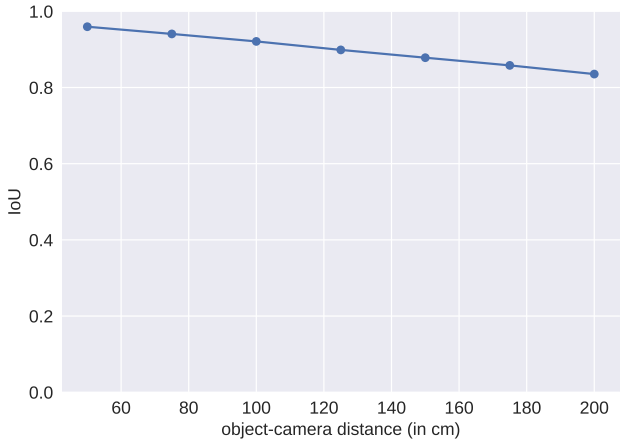

Figure 3. Ellipse prediction accuracies wrt. to the camera-object distance in the test images. Training was done at $50\,\mathrm{cm}$.

Finally, we are able to compute the full 6D pose of any camera, as long as enough objects are visible (and detected) in the image. Figure 8 shows the detected objects used for pose computation.

We generated three different camera trajectories for evaluation. The first one, presented in Figure 9, is the most similar to the training trajectories in terms of camera distance. The second trajectory (Figure 11) provides more diverse viewpoints, and, finally Figure 13 shows much more distant trajectories.

We also tested a global method, PoseNet [3], on each trajectory. The obtained camera poses are shown in Figures 10, 12 and 14. We used the PoseNet version available on GitHub[1] and trained it on our three training trajectories for 6000 epochs using the default parameters.

The obtained results indicates that PoseNet has a lot of difficulties to generalize to new camera positions, whereas our method seems to handle this much better.

The attached videos present the results obtained on a larger trajectory. The first video (*video_localization_1.mp4*) shows both the relocalized camera and the image captured by the camera. On the left, the white trajectory is the ground truth. On the right, the object detections and ellipse predictions are in green. The ellipsoids of the objects used to compute the pose are projected in blue and those that were not used are in red. The second video (*video_localization_2.mp4*) shows what the camera sees augmented with the ellipsoidal abstractions of the objects projected with the estimated camera pose. It should be noted that the frames are processed completely independently, i.e. no tracking is performed and no temporal consistency is enforced.

[1] `https://github.com/GrumpyZhou/visloc-apr`

References

[1] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar. Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. *IEEE Robotics Automation Magazine*, 22(3):36–52, 2015.

[2] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. R. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In K. M. Lee, Y. Matsushita, J. M. Rehg, and Z. Hu, editors, *Computer Vision - ACCV 2012 - 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part I*, volume 7724 of *Lecture Notes in Computer Science*, pages 548–562. Springer, 2012.

[3] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*, 2015.

[4] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. In D. J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, 2014.

[5] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019.

[6] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit*. Kitware, 2006.

[7] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[8] C. Yang, G. Simon, J. See, M.-O. Berger, and W. Wang. WatchPose: A View-Aware Approach for Camera Pose Data Collection in Industrial Environments. *Sensors*, 20(11), May 2020.

Figure 4. Predicted ellipses (green) for more distant camera positions (indicated on the left). Ground truth ellipses are in red. All the images have the same size because only the rescaled crop around the object is passed to the network. The bottom images would appear much smaller in the original full-size images.

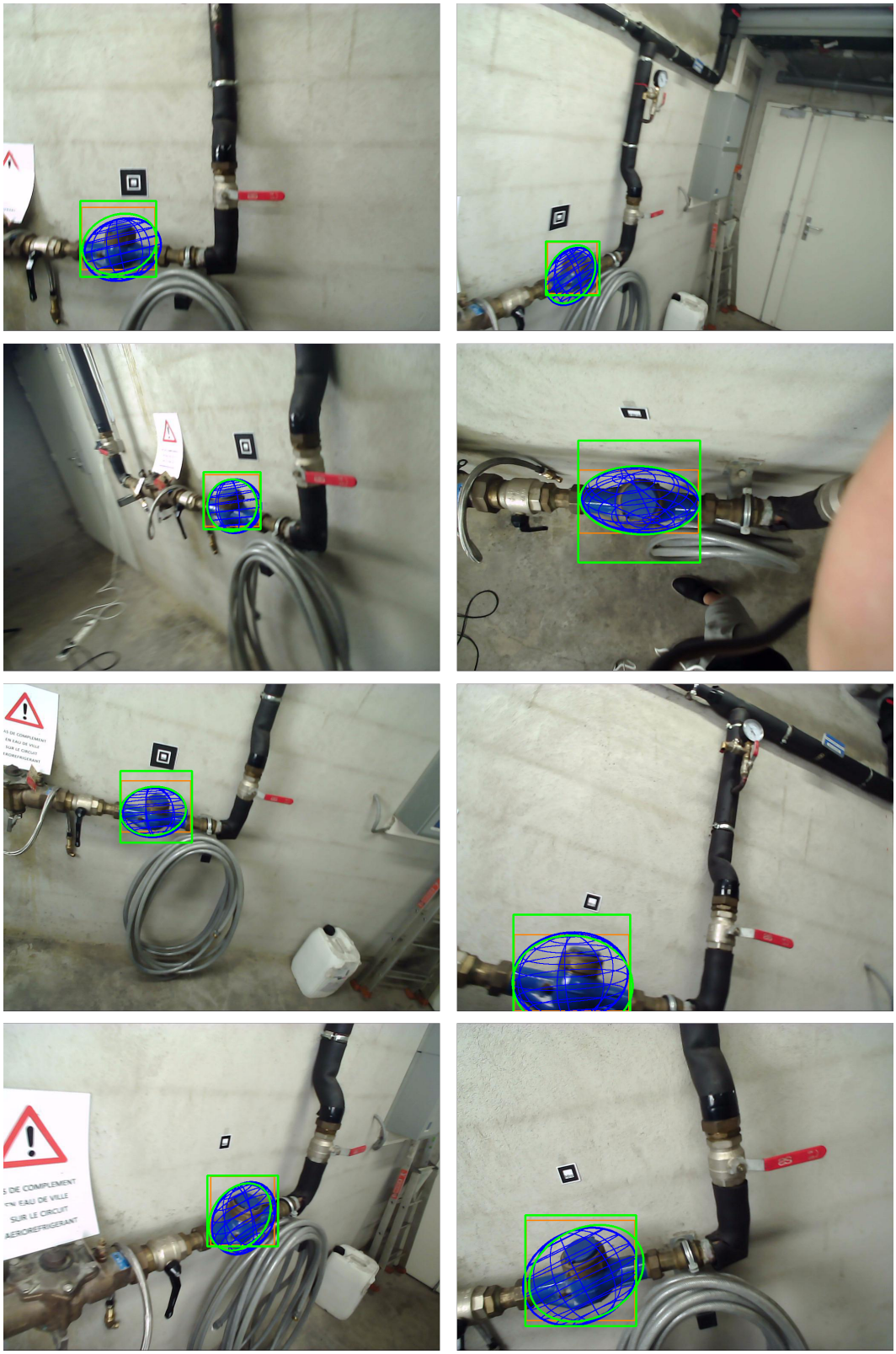

Figure 5. Additional results obtained on WatchPose [8]. Orange: object detection box. Green: square crop passed to the network and predicted ellipse. Blue: ellipsoid projection with the ground truth pose. Good matching between the predicted ellipse and the projected ellipsoid enables a precise pose estimation.

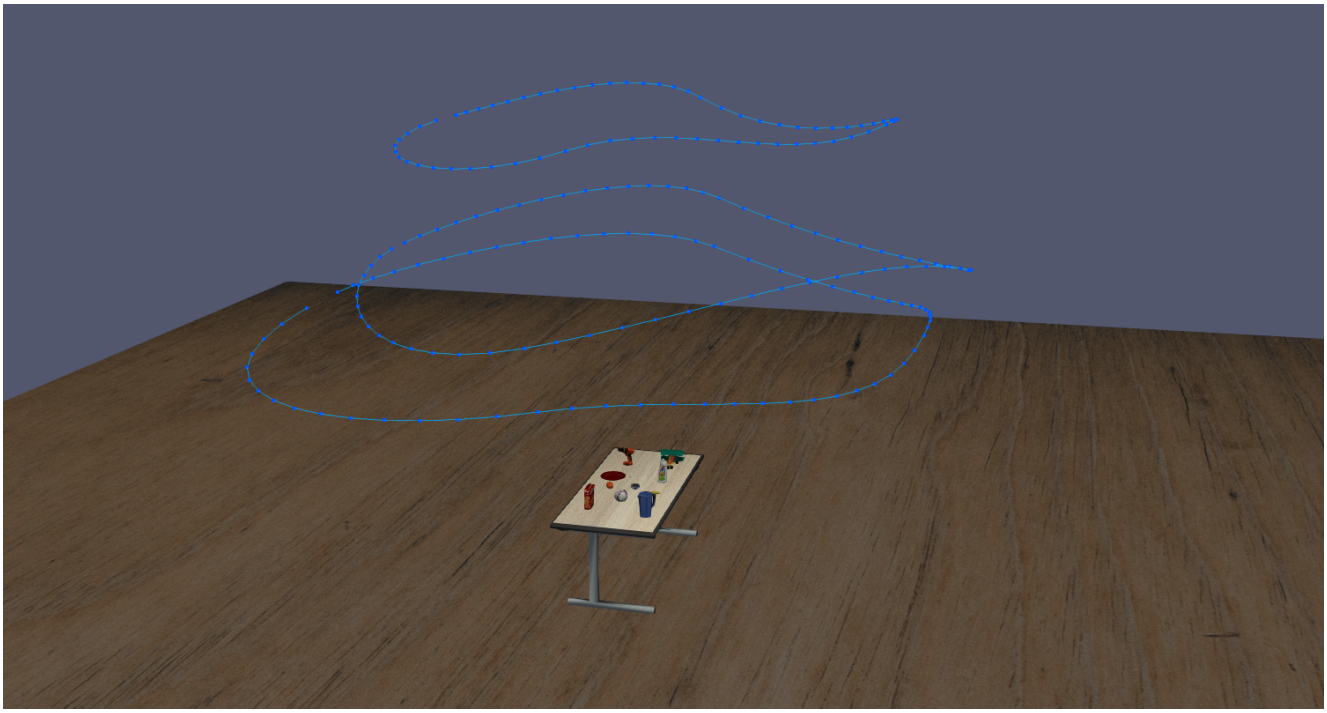

Figure 6. Camera trajectories used to generate the training data (192 images).

Figure 7. Ellipsoidal abstractions of each object (reconstructed from 5 images).

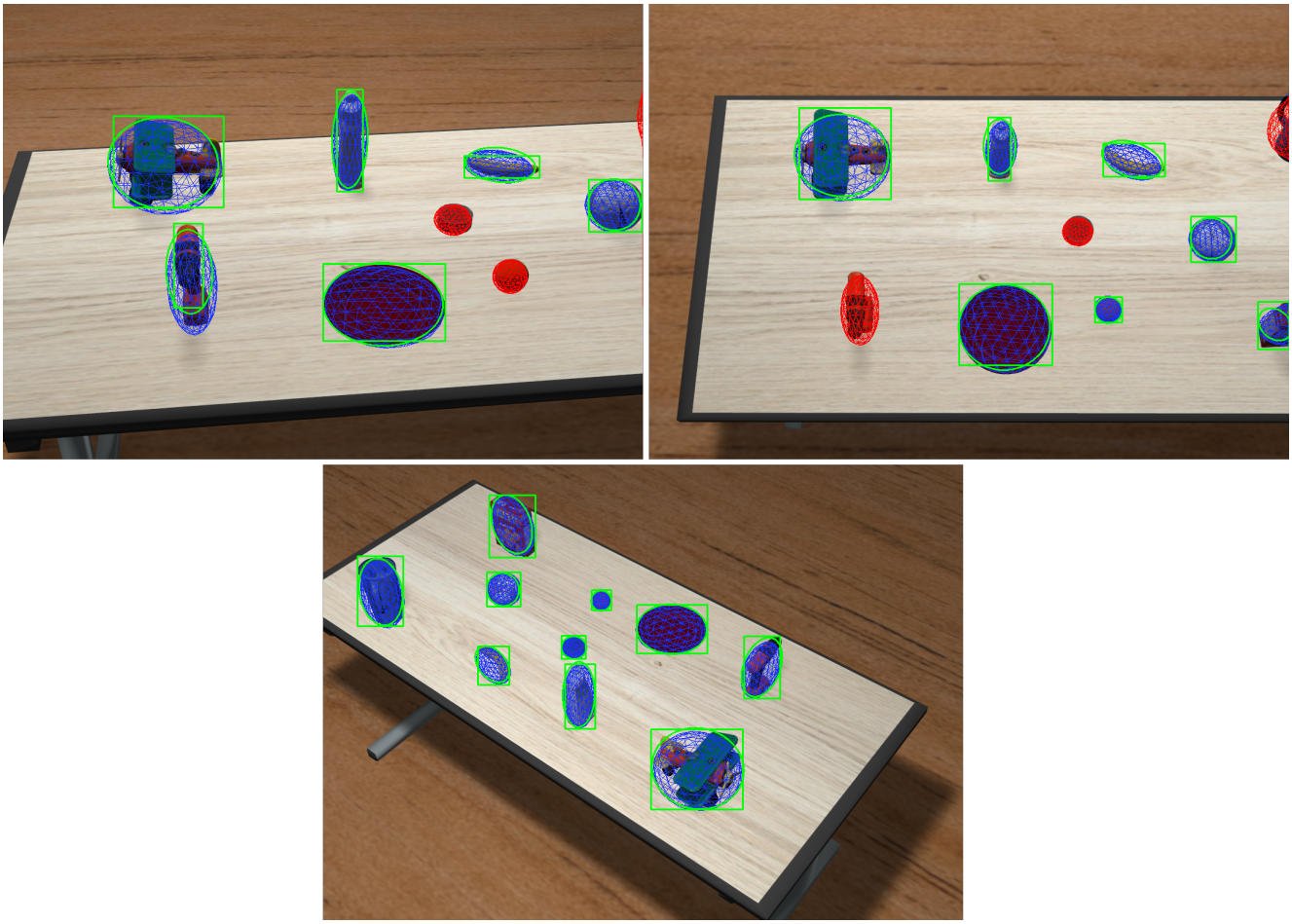

Figure 8. Detected objects (BB) and predicted ellipses are in green. The objects ellipsoids are projected in the image with the estimated pose (in blue). Those in red could not be used for pose computation because the objects were not detected in the image.

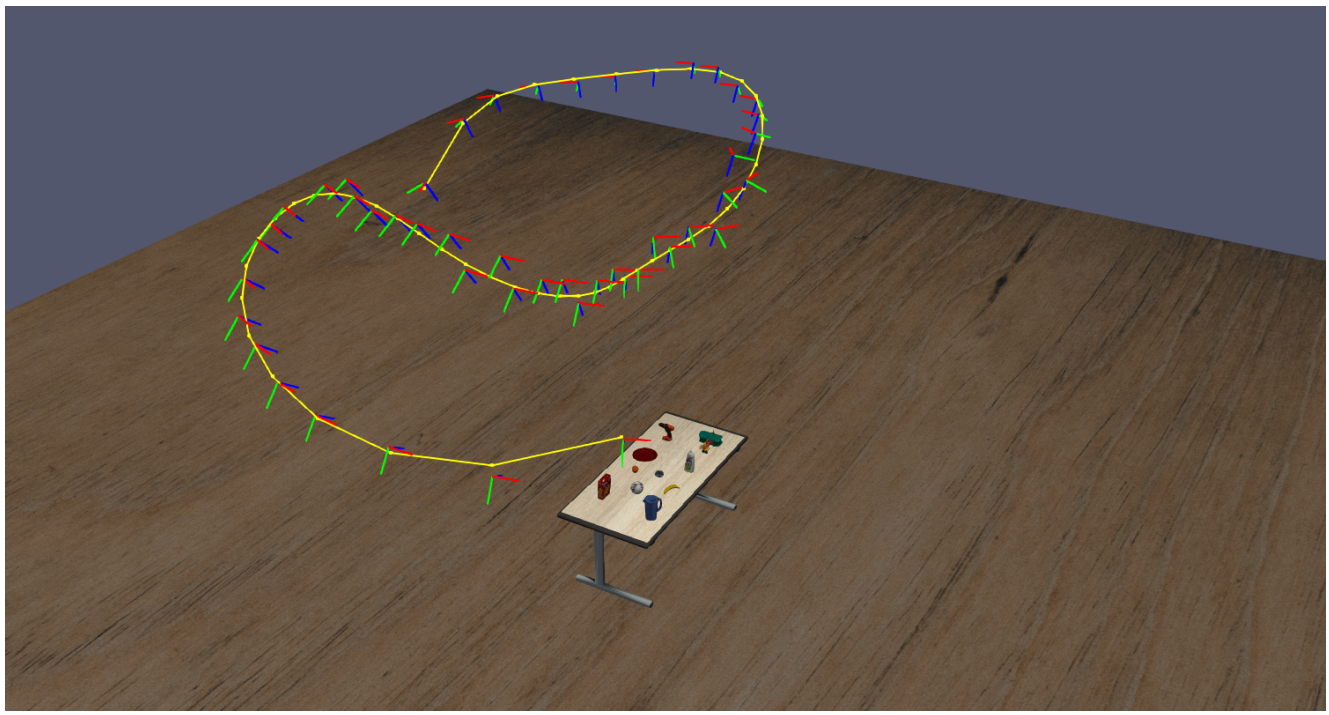

Figure 9. Our method: Estimated cameras poses and ground truth trajectory (yellow). Median position error: $7.63\,\mathrm{cm}$. Median orientation error: 0.79°.

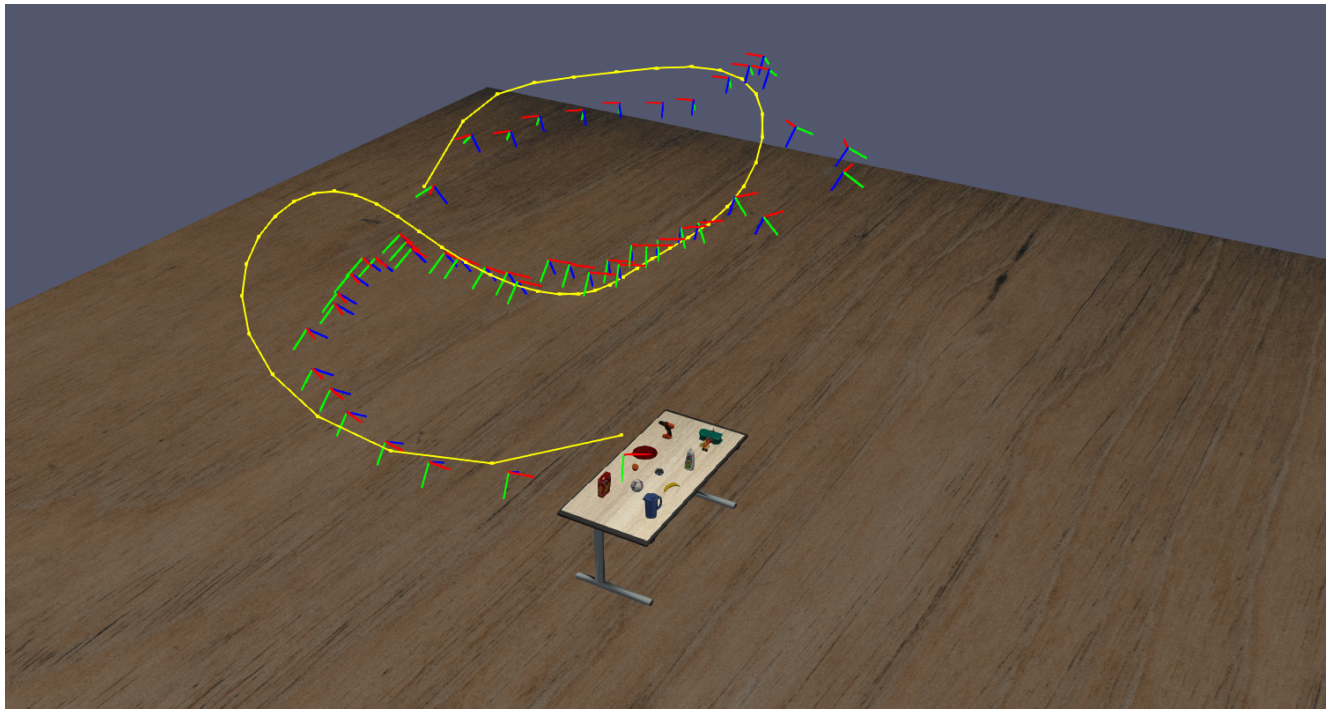

Figure 10. PoseNet: Estimated cameras poses and ground truth trajectory (yellow). Median position error: $40.13\,\mathrm{cm}$. Median orientation error: 3.04°.

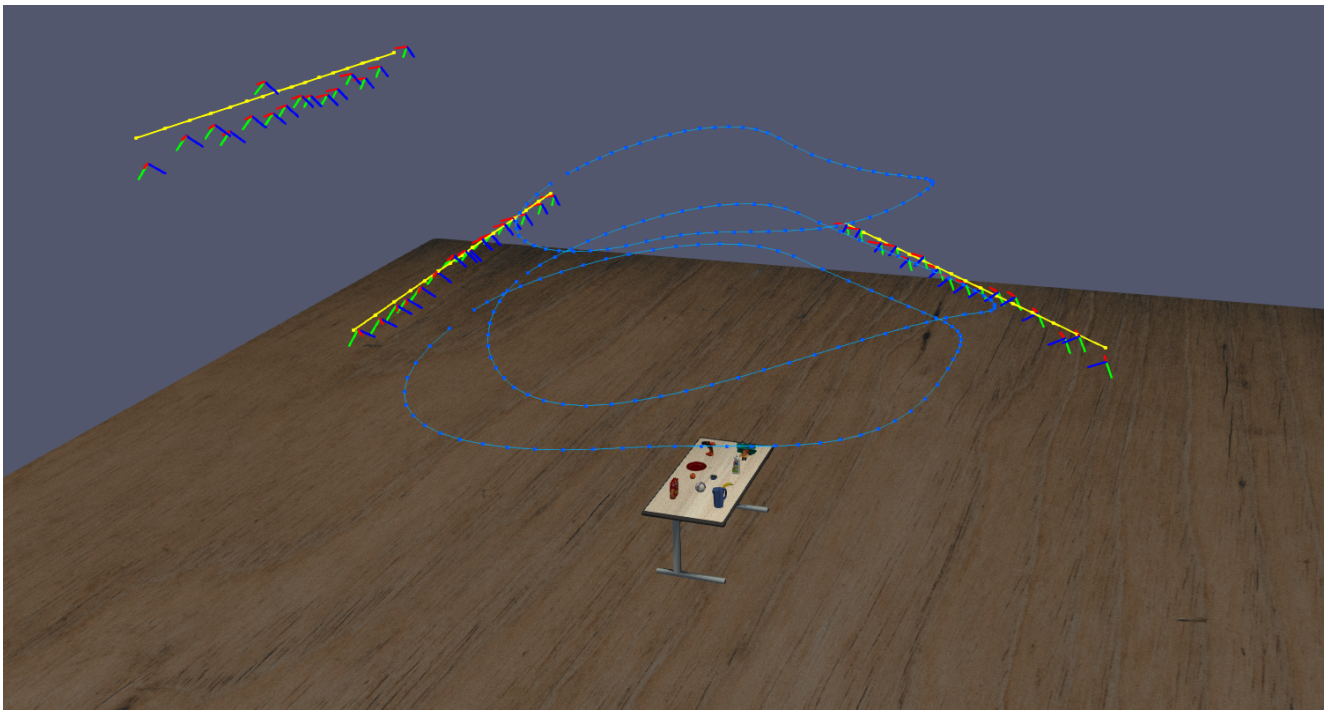

Figure 11. Our method: Estimated cameras poses and ground truth trajectory (yellow). The trajectories used for training are shown in blue. Median position error: $8.16\,\mathrm{cm}$. Median orientation error: 0.75°.

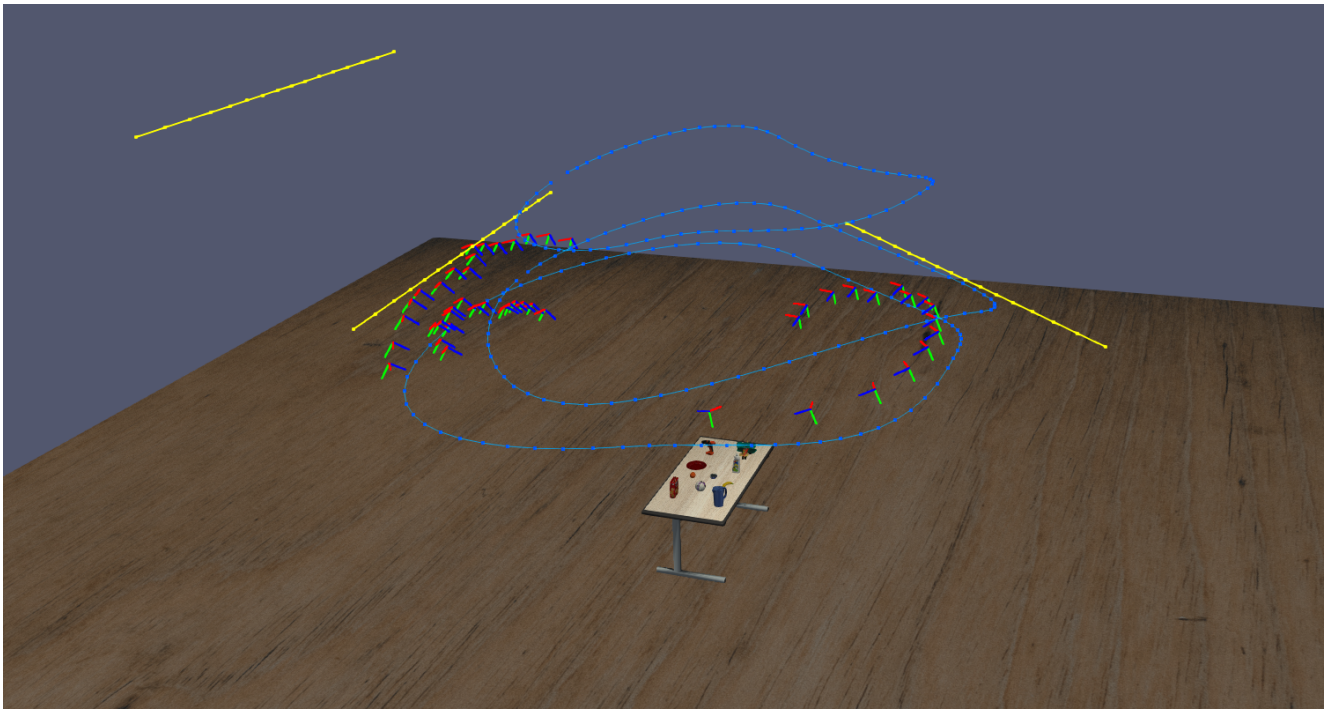

Figure 12. PoseNet: Estimated cameras poses and ground truth trajectory (yellow). The trajectories used for training are shown in blue. Median position error: $111.65\,\mathrm{cm}$. Median orientation error: 6.72°.

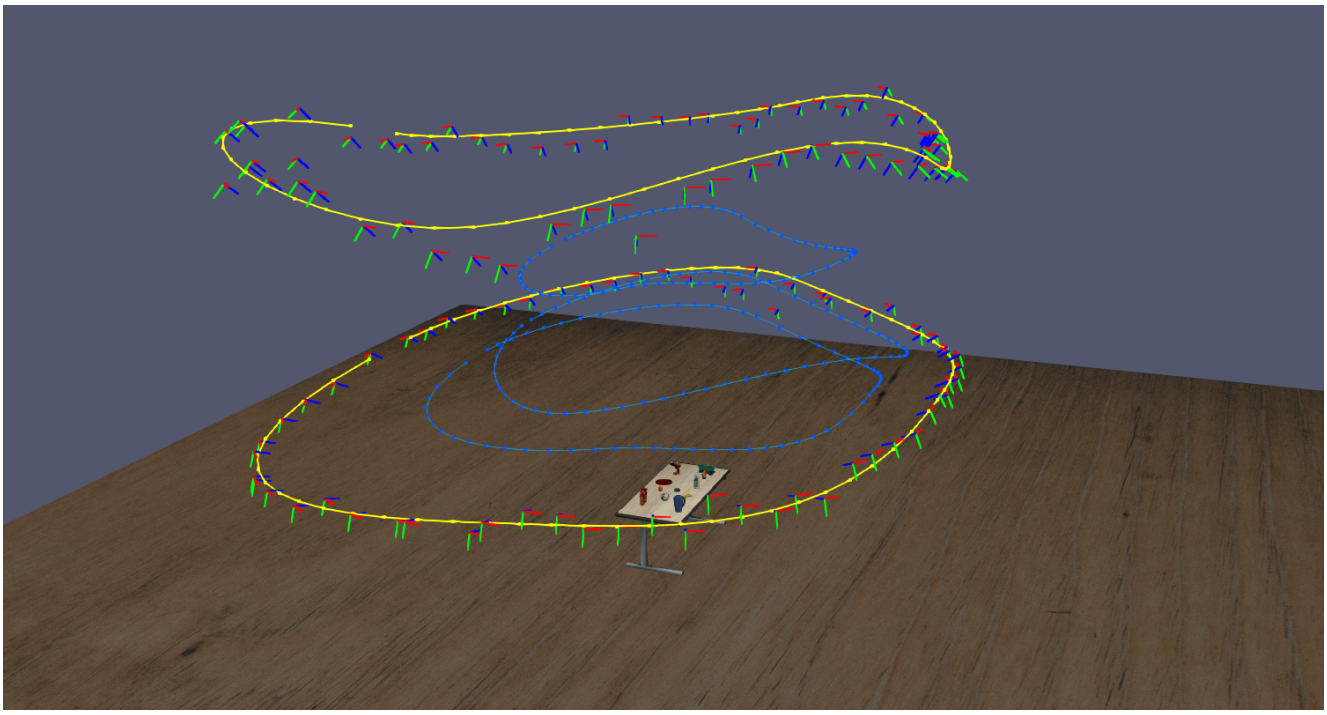

Figure 13. Our method: Estimated cameras poses and ground truth trajectory (yellow) for much more distant cameras. The trajectories used for training are shown in blue. Median position error: $14.5\,\mathrm{cm}$. Median orientation error: $1.09°$.

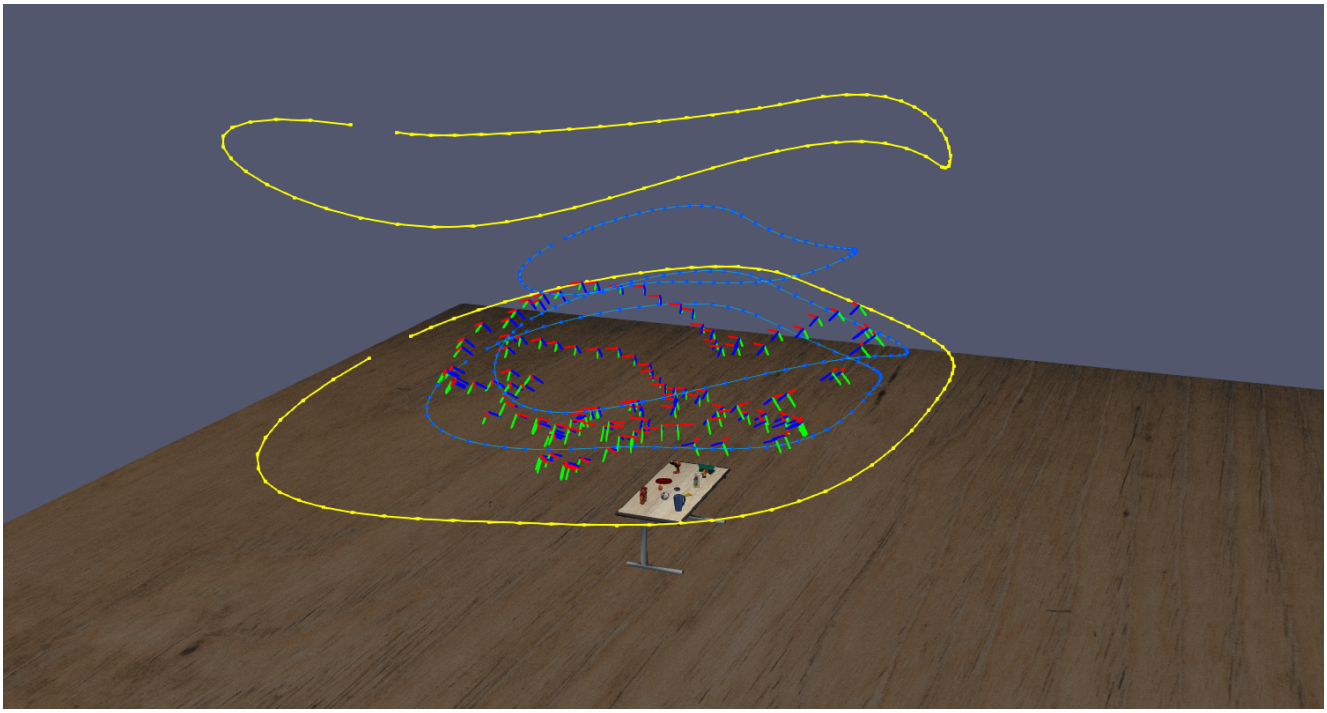

Figure 14. PoseNet: Estimated cameras poses and ground truth trajectory (yellow) for much more distant cameras. The trajectories used for training are shown in blue. Median position error: $422.30\,\mathrm{cm}$. Median orientation error: $15.82°$.