Vision-based Localization Solution for a Team of Quadrotors in Formation

Théo Gieruc

Professor : Alcherio Martinoli
Assistant(s) : Kagan Erünsal

During my semester project (Vision-based Localization Solution for a Team of Quadrotors in Formation), I designed a two-part architecture with a frontend fitting a bounding box around the quadrotor and a backend estimating its 6DoF pose using only the provided bounding box. The performance was poor (30% error in depth estimation), so I decided to try a new architecture during this summer internship.

The objective is to find, train and implement as a ROS package an end-to-end neural network to estimate the 6DoF pose of the quadrotor. Several models have been considered, such as *PVNet* (S.Peng, 2018) or *RePOSE* (S.Iwase, 2021), but I decided to use *EfficientPose* (Y.Bukschat, 2020) because of its extensive documentation and debugging tools. The model requires for training RGB images, binary segmentation, rotation matrices and translation vectors, the intrinsic parameters of the camera and a 3D model of the quadrotor.

In order to train the neural network, I created a dataset with a static RGB-D camera, a quadrotor and a Motion Capture System (MCS). I used an Intel Realsense D415 recording with a 480p resolution and a ZED Mini with a 1080p resolution, in order to compare the upside or downside of having higher-resolution images.

I tried several 3D scanning algorithms but ended up merging by hand several clouds of points captured with the Realsense D415 camera in order to have a 3D model of the quadrotor. I also built a low-poly 3D model on Fusion 360 in order to observe how the complexity of the object influence the results.

Binary segmentation of the images has been achieved with the framework *Segmentation Models PyTorch* (P.Iakubovskii, 2019), that provides 9 architectures and over 100 pretrained encoders. I used both RGB and RGB-D images for segmentations and had better results models using only the RGB channels, probably due to the availability of pretrained weights.

Relative positions have been easily computed from the absolute measurements of the MCS using rotation matrices. Translation offsets and rotation pitch correction matrices have been tuned by hand in order to align the ground-truth 3D bounding boxes on the quadrotor images

No offset / pitch correction With offset / pitch correction

.

The training has been executed on a P100 GPU using my Google Colab Pro subscription, for over 60 hours per model, with pretrained weights on the LineMOD dataset. Using high-resolution images has greatly slowed down the training, resulting in an under-trained model after 100 hours. I decided to keep only the results on the lower-resolution dataset. Colorspace and spatial augmentation have been used in order to help the training and generalization.

Metrics	Scanned model	CAD model	MultiTracker MSL-RAPTOR
mAP	0.9905	0.9934	0.989
ADD	0	0.0954	-
ADD-S	0	0.7932	-
< 5cm 5degree Error	0	0.08	-
Mean Translation Error [mm]	41	27	451
Std Translation Error [mm]	49	37	331
Mean Rotation Error [degree]	120	86	-
Std Rotation Error [degree]	33	36	-

The CAD model has better results than the manually merged model, probably because of its simplicity, as it gives more importance to the global shape of the object rather than the smaller details. Globally, the CAD model gives 16 times better results than the MultiTracker + MSL-RAPTOR architecture but is 3 times slower.

The main issue is the poor generalization of the model, due to the static camera. When the background changes, the network is not able to detect the quadrotor anymore, as it has been trained with only one background. Training on a more diverse dataset would probably help with the generalization.