Monocular markerless 6D pose estimation of ANYmal

Ivan Alberico
Department of Mechanical and Process Engineering (D-MAVT)
ETH Zürich, Switzerland
`ialberico@student.ethz.ch`

Kexin Shi
Department of Computer Science (D-INFK)
ETH Zürich, Switzerland
`kexshi@student.ethz.ch`

Abstract: Localization is an important task when it comes to tracking robots accurately in complicated and changeable environments. Previous methods generally rely on additional sources like depth cameras or QR codes placed in the surrounding environment. In this work, we propose to remove the dependency from these external sources by deploying state-of-the-art 6D pose estimation deep learning methods to achieve the same goal.

Keywords: 6D pose estimation, ANYmal, Deep Learning, Localization

1 Introduction

A core aspect of mobile robotics is to estimate the pose of a robot within a given environment in the most precise and reliable way. Traditionally, the most common approach to achieve this goal is to process cues present in the surrounding environment from which it is possible to localize the robot in the space. In this work, attention is shifted onto the user rather than the environment, as the goal is to localize the robot with respect to a camera held by an operator. The idea is that of being able to estimate the pose of a robot directly from a video-stream, hence relying on RGB frames only. This work aims at achieving the goal by means of deep learning based methods, trained in a supervised way on a collected dataset. The project lays the groundwork for interesting applications supporting the ongoing research at the RSL lab at ETH Zurich, offering a tool to directly infer the pose of ANYmal in a portable manner while deploying the robot in the real world environment. For example, an interesting application would be implementing an Augmented Reality mobile application that is able to accurately co-localize the robot with respect to the user and display all the available sensory information as AR features added in the scene.

Prior work addressing the task of localizing ANYmal in the space has been carried out only with the aid of external sources, making these methods not suitable for all situations. Moreover, solutions like placing QR codes in the environment are not always reliable, as it can happen that markers are occluded or that the detection system fails under certain environmental and lighting conditions. The way we aim at removing this dependency is to build the whole localization system on deep neural networks, having as only input requirement the video-stream in which we want to detect the robot. The project represents also an attempt to generalize and assert the validity of the state-of-the-art 6D pose estimation methods to more complex and dynamical shapes, like that of ANYmal. The way we planned to achieve the final result comprises the following two building blocks:

- Generation of an accurate dataset containing RGB images, ground truth 6D pose of the base of ANYmal, and the mask of the robot in each frame;
- Training of a 6D pose estimation network on the generated dataset;

2 Related Work

In this section we briefly review existing work on 6D pose estimation from classical hand-crafted features methods to newer end-to-end trainable CNN-based methods.

The vast majority of previous methods uses depth sensors or RGB-D cameras[1, 2, 3, 4, 5, 6, 7], leveraging depth information to disambiguate the object's scale, which is generally lost due to perspective projection in RGB images. That is also the reason why it is quite challenging to detect objects and estimate 6D poses from RGB images only. Some conventional work use hand-crafted features or direct image information, such as gradients and image pixel intensities. Recently, learning-based methods are proposed to learn features automatically.

In general, the most straightforward learning-based way to estimate the 6D pose of an object is to regress directly translation and rotation parts, like in the work of PoseCNN[8]. Another method is to discretize the continuous rotation space into several bins and classify them accordingly[9, 10]. However, results of these methods generally need additional refinement steps in order to provide an accurate 6D pose estimate.

To address this problem, pose estimation methods based on traditional 2D objection detection algorithms are proposed. Most recent state-of-the-art works with RGB images focus on first detecting 2D targets of the object in the given image and subsequently solving a Perspective-n-Point(PnP) problem with predicting 2D-3D correspondences for 6D poses[11, 12, 13, 14, 15, 16]. These dense 2D-3D correspondences are either obtained by UV maps[17] or regressing the coordinates in the object's 3D model space[12, 13]. These methods are often reported with high accuracy and fast inference times on Linemod dataset[18].

Most objection detection algorithms are two-stage approaches[19, 20, 21] which perform a region proposal step in the first stage and then make a final object detection step in the second stage based on the region proposals. Nevertheless, two-stage methods increase the model complexity. One-stage detectors gained more attention due to their simplicity and efficiency, like the work in EfficientDet[22]. In this paper, the authors focused not only on accuracy but also on efficiency based on the scalable backbone architecture EfficientNet[23]. To introduce those advantages also in 6D pose estimation field, they extended EfficientDet[22] to EfficientPose[24] by adding two more sub-heads for rotation regression and translation regression. In our project, we adopted the EfficientPose[24] model, which proposes an efficient way to predict poses without any post-processing procedures like PnP and RANSAC[25], resulting in a more straightforward and accurate result.

3 Method

In this section we first discuss the training data generation process, including all the pre-processing steps required to come up with accurate ground truth poses of ANYmal. Then, we briefly describe the deep learning network deployed for the 6D pose estimation part and how it is integrated in the whole pipeline.

3.1 Dataset generation

The crucial part for the dataset generation process is to estimate precise ground truth poses from the recorded videos, in order to provide a solid supervision for the future training. The ground truth poses of the base of ANYmal are generated by detecting AprilTags attached to a mounting fixed on top of the robot. The mounting consists of a vertical shaft attached on the upper surface body of ANYmal, with a cube placed on one of the far ends, on which different AprilTags are glued. The mounting was designed and placed such that we are able to detect at least one QR code from different viewpoints in every frame. The whole ground truth pose estimation environment is set up with ROS Melodic on Ubuntu 18.04 and the videos are recorded with a GoPro HERO 10.

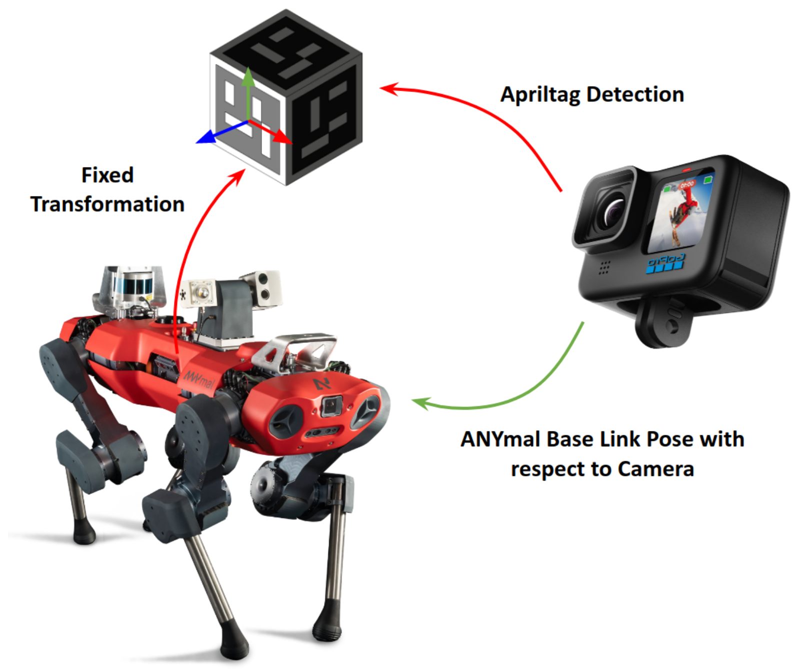

Figure 1: Diagram of the dataset generation module, showing how the relative transformations between camera, cube and base link of ANYmal are related one another.

3.1.1 Camera Calibration

First of all, camera calibration is performed in order to estimate the intrinsic parameters of the device. The camera calibration step is performed with the publicly available package Kalibr [1], and it represents a core part for the early stage of the project, as it significantly affects the performance of the AprilTags estimation module. Pinhole camera model and radial-tangential distortion model are chosen as reference models, and a 6×6 checkerboard with April tags of size 8.3 cm and spaced 2.49 cm from each other was used to calibrate the camera. Different recording modalities offered by the GoPro HERO 10 are investigated, namely *NARROW*, *LINEAR* and *WIDE* modalities, in order to analyze how they affect the calibration performance, as well as different input video resolutions (i.e. 1080p and 4K). Results of the camera calibration will be shown in Appendix .

3.1.2 AprilTag detection and post-processing

In order to estimate the pose of the individual AprilTags we leveraged the already existing ROS package *apriltag_ros* [2], which directly provides position and orientation of each tag with respect to the camera. Knowing the fixed transformations between each marker and the base link of the robot, estimated by visual inspection of the 3D model of ANYmal on Blender[3], we are able to extract the 6D pose of the base link. The complete 6D pose is composed of two parts - the 3D rotation $R \in SO(3)$ and the 3D translation $t \in R^3$. This 6D pose represents the rigid transformation from the ANYmal base link coordinate system into the camera coordinate system.

One final challenge to overcome in order to guarantee better estimates of the ground truth poses was to attenuate the jittery behaviour caused by detecting multiple tags in each frame. This jitter is caused by irreducible mismatches among the poses estimated from the different faces of the cube. Whilst the effect on the position is diminished by averaging the position vectors of each detected

[1] https://github.com/ethz-asl/kalibr
[2] http://wiki.ros.org/apriltag_ros
[3] https://www.blender.org/

face, as far as orientation is concerned, we implemented an efficient solution following [26], which solves the averaging as an optimisation problem based on its eigenvector decomposition, as shown in the following:

$$\bar{q} = \underset{q \in \mathcal{S}^3}{\operatorname{argmax}}\, q^T M q \qquad\qquad M = \sum_{i=1}^{n} w_i \; q_i \; q_i^T \tag{1}$$

where w_i refers to the weight associated to each quaternion q_i in the averaging process. The averaged quaternion is the eigenvector of M corresponding to the maximum eigenvalue, and it has unit norm by construction.

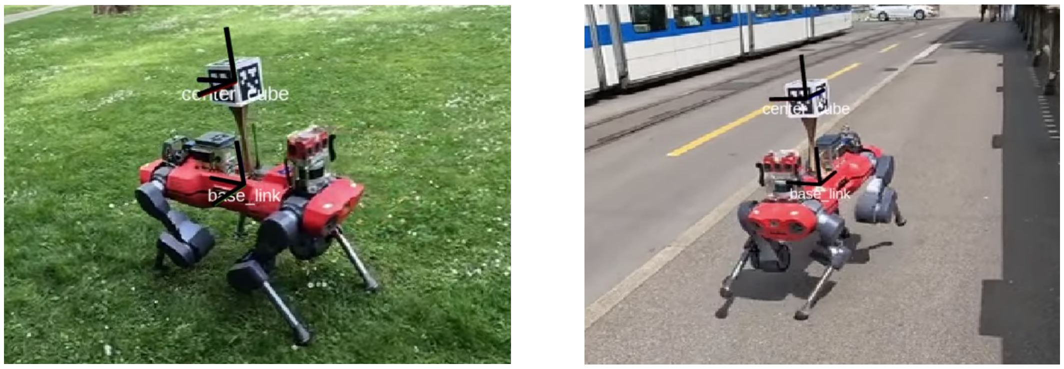

Figure 2: Visual inspection of estimated ground truth poses on different sequences.

3.2 6D pose estimation network model

The model we deployed for the learning part is called EfficientPose[27], one of the state-of-the-art method for 6D pose estimation. It is based on the one-stage object detection algorithm called EfficientDet[22], and extends functionality to 6D pose estimation in a simple and intuitive way.

As it can be seen from the schematic representation in Figure 3, the network first extracts image features with the scalable backbone architecture EfficientNet[23]. After that, the features are fed into a bidirectional feature pyramid network (BiFPN) to extract and fuse features at multiple scales. Finally, all features are fed into 4 sub-networks. Each block consists of a classification network, a 2D bounding box regression network and additionally a rotation and translation regression network. Since these last 2 subnets are relatively small and share part of the computation of the input feature maps with the already existing networks, we are able to get the full 6D pose very inexpensive without much additional computational cost.

What is required in order to train the network are the RGB images of ANYmal, the corresponding binary mask of the object of each frame, the 3D object model and the ground truth poses. From the URDF model of ANYmal, we first generate the mesh of the base link, excluding legs and feet of the robot that constantly change their configuration in time, and then we sample points from this mesh. The binary mask is then computed by projecting these points onto the image plane, knowing the relative transformation with respect to the camera, as well as the intrinsics parameters. The binary mask is used for the groundtruth of 2D bounding boxes. In test, we only need RGB images.

In fact, this model is more efficient than other methods that need post-processing steps like RANSAC or PnP, because in this case the 6D pose is regressed directly. The rotation network regresses rotations in the form of axis angle representation $r \in R^3$ for each anchor box, as it needs fewer parameters with respect to quaternion or rotation matrix representations. Additionally, the architecture of the rotation network is defined by a scaling hyperparameter ϕ, which defines the number of layers of each refinement module and also how many times it needs to be applied at each step.

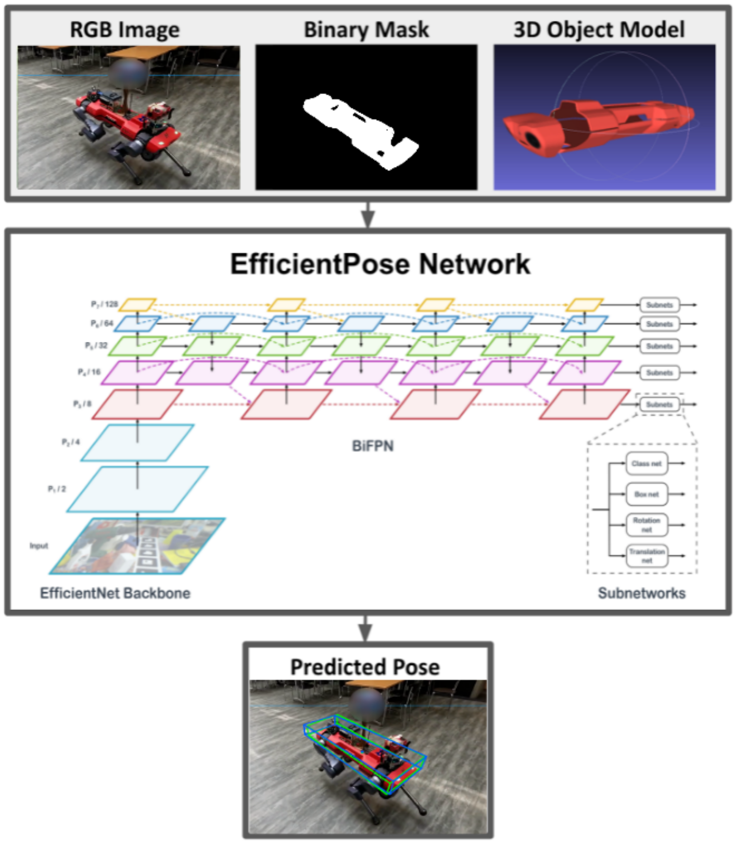

Figure 3: Flowchart of how the 6D pose estimation network is deployed starting from the generated dataset.

4 Experimental Results

In this section we evaluate our algorithm in terms of both ground truth pose generation and training performance of the 6D pose estimation network.

4.1 Pose generation results

The videos used to generate the dataset were recorded in 4K, with the cube attached to the robot having size 15 cm with AprilTags of 13 cm glued on each face, leaving 1 cm of white border for contrast. The validity of the estimated poses has been ascertained with a laser distance meter, so that we could measure the real distance of the cube from the camera and compare it with the estimated one. As far as single AprilTag detection is concerned, results are rather accurate. In the range 0-2 m from the camera, the accuracy of the detection module is significantly high with an error of approximately $\pm$ 3 mm. The tags are detected up to a distance of 5-6 meters. Naturally, the further the robot moves away from the camera and the more error-prone the estimates will be.

On the other hand, less accurate results are observed when it comes to bring the estimates of each individual AprilTag together, in order to compute the pose of the center of the cube. By projecting the estimated reference frames onto the scene, as shown in Figure 5, it is observed that there is an error of roughly $\pm$ 2-3 cm around the real center of the cube, which is more emphasized when moving from one AprilTag to the other. We believe that this is caused by small mismatches in the way the markers were attached onto the cube, which are then summed up to the inherent errors coming from the detection module.

4.2 Neural network training results

All experiments were conducted on selected sequences of the dataset, which were manually inspected in order to filter out all the corner cases that might have affected the training procedure in a negative way. We mainly focused our experimental analysis on the training of 3 sequences, of approximately 2-3 minutes each, recorded at 30FPS in 4K. Before feeding the RGB sequences to the network, each frame is first down-sampled to a resolution 640×480 in order to comply with the Linemod dataset[18] which represented the benchmark for EfficientPose. All experiments are run on the ETH Euler cluster using NVIDIA GeForce RTX 2080 Ti GPUs with Tensorflow 1.15.0, CUDA 10.0 and CuDNN 7.6.5. We train on each sequence independently, splitting training and validation set between the first 70% and the last 30% of the frames. The network is initialized with weights of EfficientDet [22] pre-trained on COCO dataset [28], and every training process runs for 500 epochs, with a batch size of 4 and a learning rate of $1e^{-4}$.

We evaluate our approach with the commonly used ADD metric [29], which computes the average point distances between the 3D model point set M transformed with the ground truth and the predicted poses, as shown below:

$$\text{ADD} = \frac{1}{m} \sum_{x \in M} \|(Rx + t) - (\hat{R}x + \hat{t})\|_2 \tag{2}$$

In addition to the ADD metric, we also evaluate the model with the ADD accuracy score, which represents the rate of the ADD being smaller than 10% of the object diameter within each sequence. In our specific case, as the diameter of ANYmal is roughly of 1.04m, a predicted pose is considered to be correct and it is included in the ADD accuracy score whenever it stays within a range of $\pm$ 10 cm from the ground truth pose. In Figure 4 we show the result of training the network on a sequence of the dataset in terms of the ADD metric and ADD accuracy score.

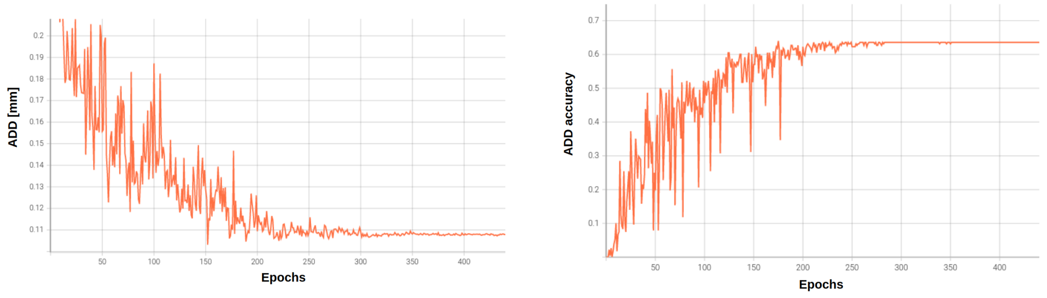

Figure 4: Metrics performance after training the network on a sequence of the dataset.

In order to guarantee that the presence of markers in the scene would not bring meaningful information to the network while predicting the pose, we blur the cubes in the validation set and test the model on the blurred frames. Visual inspection of the results by means of 3D bounding boxes centered and oriented according to the predicted poses in each frame, as displayed in Figure 5, shows that the network generalizes well to unseen frames within the same sequence. Cross-validating the trained networks among different sequences still does not lead to successful results in all the cases, but we believe that this is due to the limited amount of data we are currently training and testing our models on. As far as run-time performance is concerned, the average execution time of the network to evaluate a single frame is 39.26146 ms, which makes it perfectly suitable to run in real-time.

In Table 4.2 we present the values of the maximum ADD accuracy score for each sequence, while training the network with different values of ϕ. The observed performances of the model with $\phi = 0$, $\phi = 1$ and $\phi = 3$ are almost comparable, although the different network capacities. This suggests that the capacity of the network with $\phi = 0$ is already enough for the single object 6D pose estimation task on ANYmal and that the bottleneck seems to be caused more from the small

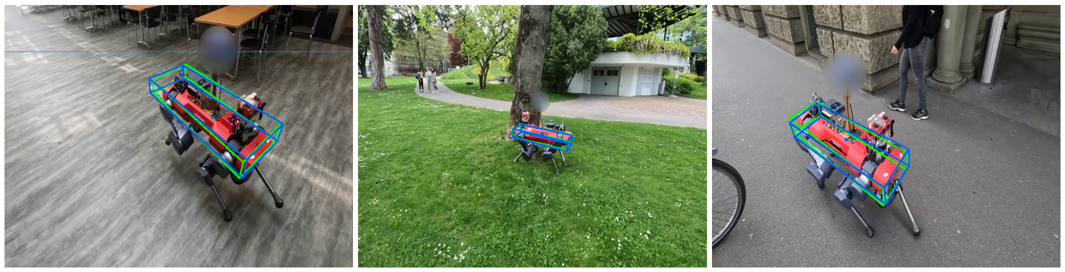

Figure 5: Visual inspection of predicted poses on validation set from different training sequences. The green bounding box represents the ground truth pose, while the blue one the predicted pose from the network.

Training sequence	$\phi = 0$	$\phi = 1$	$\phi = 3$
sequence 1	**0.643**	0.627	0.631
sequence 2	0.623	**0.628**	0.616
sequence 3	**0.589**	0.568	0.582

Table 1: Quantitative evaluation in terms of the ADD(-S) accuracy metric for 6D pose estimation of ANYmal on 3 different sequences of the generated dataset.

amount of data. Additionally, the reason why the small $\phi = 0$ actually performs slightly better then the other cases might be that the shallower network may not suffer from overfitting as much as its larger counterparts.

5 Conclusion

In this project, our contributions include the generation of an accurate dataset 6D pose estimation of ANYmal, and training the EfficientPose[24] algorithm on this generated dataset with an ADD accuracy score of over 60%. Furthermore, there is still a lot of future work to improve results:

- The original EfficientPose[24] algorithm is trained and tested on the Linemod dataset, which does not take into account the relations between adjacent frames. Since our dataset is made in sequence order, we could combine 6D pose estimation with tracking methods such as Kalman filters.
- For simplification, our current object model only includes the base of ANYmal (front face, rear face, top shell and bottom shell), which is fixed while the robot is moving. It will be more accurate if we extend the 3D object model to legs and feet. However, if we want to include these additional parts, we may need to integrate knowledge from robot dynamics.
- For a better generalization ability, we can train this algorithm on a larger dataset (with different lighting conditions, background environments and viewpoints). Another way is to augment the dataset with synthetic data from simulators or use 6D data augmentation method from EfficientPose paper[24].
- EfficientPose[24] model is designed to estimate 6D poses for multiple objects in one scene. To this end, the model contains not only object detection head and pose estimation heads, but also head to classify different objects. Based on it, we can also try to generalize our model under conditions where multiple objects or multiple ANYmal robots are visible in the same frame.
- For future validation, it is better to perform more benchmarking experiments with other 6D pose estimation algorithms applied on ANYmal which are mentioned in related work part.

References

[1] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6d object pose estimation using 3d object coordinates. In *European conference on computer vision*, pages 536–551. Springer, 2014.

[2] C. Choi and H. I. Christensen. Rgb-d object pose estimation in unstructured environments. *Robotics and Autonomous Systems*, 75:595–613, 2016.

[3] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab. Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation. In *European conference on computer vision*, pages 205–220. Springer, 2016.

[4] K. Lai, L. Bo, X. Ren, and D. Fox. A scalable tree-based approach for joint object and pose recognition. In *Twenty-fifth aaai conference on artificial intelligence*, 2011.

[5] F. Manhardt, W. Kehl, N. Navab, and F. Tombari. Deep model-based 6d pose refinement in rgb. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 800–815, 2018.

[6] J. Sock, S. Hamidreza Kasaei, L. Seabra Lopes, and T.-K. Kim. Multi-view 6d object pose estimation and camera motion planning using rgbd images. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2228–2235, 2017.

[7] H. Zhang and Q. Cao. Combined holistic and local patches for recovering 6d object pose. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2219–2227, 2017.

[8] G. Chéron, I. Laptev, and C. Schmid. P-cnn: Pose-based cnn features for action recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 3218–3226, 2015.

[9] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE international conference on computer vision*, pages 1521–1529, 2017.

[10] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the european conference on computer vision (ECCV)*, pages 699–715, 2018.

[11] B. Chen, A. Parra, J. Cao, N. Li, and T.-J. Chin. End-to-end learnable geometric vision by backpropagating pnp optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8100–8109, 2020.

[12] Z. Li, G. Wang, and X. Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7678–7687, 2019.

[13] K. Park, T. Patten, and M. Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7668–7677, 2019.

[14] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4561–4570, 2019.

[15] C. Song, J. Song, and Q. Huang. Hybridpose: 6d object pose estimation under hybrid representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 431–440, 2020.

[16] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[17] S. Zakharov, I. Shugurov, and S. Ilic. Dpod: 6d pose object detector and refiner. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1941–1950, 2019.

[18] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *2011 international conference on computer vision*, pages 858–865. IEEE, 2011.

[19] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[20] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[21] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[22] M. Tan, R. Pang, and Q. V. Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[23] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[24] Y. Bukschat and M. Vetter. Efficientpose: An efficient, accurate and scalable end-to-end 6d multi object pose estimation approach. *arXiv preprint arXiv:2011.04307*, 2020.

[25] M. A. Fischler and R. C. B. R. S. Consensus. A paradigm for model fitting with applications to image analysis and automated cartography. *SRI International*.

[26] J. L. C. F. Landis Markley, Yang Cheng and Y. Oshman. Averaging quaternions. *Journal of Guidance, Control and Dynamics, Vol. 30, No. 4*, July-August 2007. doi:10.2514/1.28949.

[27] Y. Bukschat and M. Vetter. Efficientpose: An efficient, accurate and scalable end-to-end 6d multi object pose estimation approach, 2020. URL `https://arxiv.org/abs/2011.04307`.

[28] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft coco: Common objects in context, 2014. URL `https://arxiv.org/abs/1405.0312`.

[29] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. 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*, pages 548–562, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

A Appendix - Camera calibration results

The best performance in terms of reprojection error was found by calibrating the camera with a video recorded in *WIDE* mode with 4K resolution in 30fps. As shown in Figure 6, the reprojection error resulting from calibrating the camera under these settings stays bounded withing 1 pixel in most of the cases, with a mean of ± 0.356050 along the x-axis and ± 0.331232 along the y-axis.

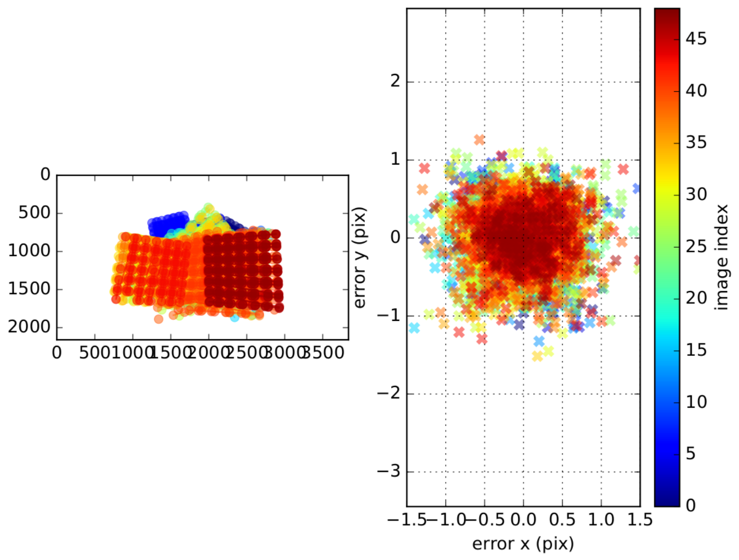

Figure 6: Reprojection error of the GoPro calibration with Kalibr.

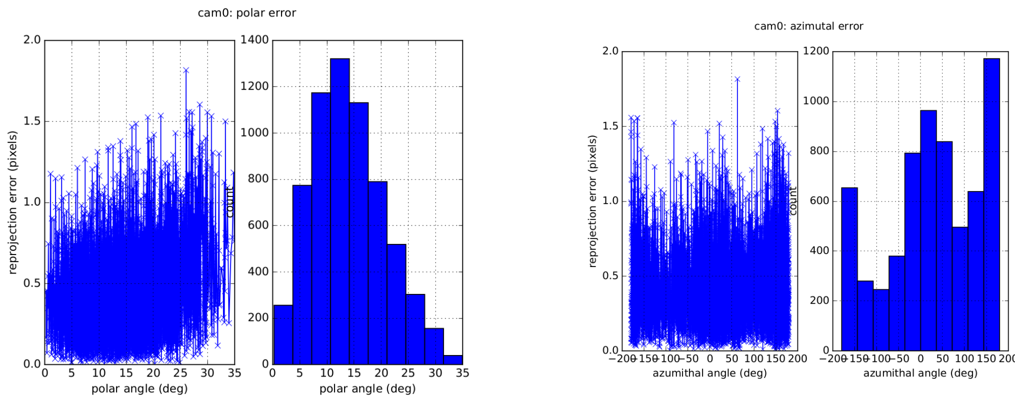

Figure 7: Polar and azimutal error of the GoPro calibration with Kalibr.