MVA - Project Proposal for 6D Object Pose Estimation

Alexis GROSHENRY - alexis.groshenry@polytechnique.edu

January 28, 2022

Abstract

This project report studies the topic of 6D Object Pose Estimation from RGB images. Starting from the DenseFusion framework for RGB-D images [14], we propose two approaches to extend it to this more challenging setting. The first one bootstraps the depth estimations produced by TransDepth [17], a depth prediction network based on transformers, and the second approach directly adapts the framework to make predictions from RGB images. We notably report good results on the direct RGB adaptation for low cost modifications of the original pipeline.

1. Introduction

6D object pose estimation is an ancient topic at the core of many real-world applications, which aims to produce a robust estimate of object position and orientation from an input image. An important part of the recent literature focuses on predicting the pose of objects for RGB-D images. However, the acquisition of the depth channel requires dedicated devices whose measurements may eventually fail in some scenarios such as outdoor scenes. Therefore, it is an important topic to propose methods suited for 6D Pose Estimation from RGB images.

This project proposes to adapt the DenseFusion [14] method to tackle this problem. First, we try to replace the ground-truth depth channel by the predictions of a depth prediction network. Then, we propose a modified network to directly make predictions on RGB images. Finally, we analyse the results provided by these two methods, and compare them with those obtained from RGB-D images with the original framework.

2. Related Work

6D Pose Estimation A number of methods rely on the detection of keypoints in an image, using them to realize a matching with known objects with patch-based approaches [2, 12], or to predict the center of the object and then perform a regression to get the corresponding transformation matrices [10, 15, 8]. These methods are particularly well suited to make predictions from color images, and use the eventual depth channel for refinement purpose only. For example, [8] proposes an alternative to the classical ICP algorithm [1] by introducing a residual pose estimator network which efficiently performs local adjustments of the prediction.

However, keypoints methods come to fail on occluded or low-textured objects, while other methods show robustness against such data by using CNN to extract appearance and geometric features from the image. Xu *et al.* introduce the PointNet network [9] which creates meaningful features from a 3D point cloud for various applications. This network is often used as part of pose estimation pipelines, like in PointFusion [16] where the geometric features extracted by a PointNet architecture are fused with appearance features extracted by a CNN from the color image. DenseFusion [14] takes the best from previous approaches by making pixel-wise pose predictions from geometric and appearance features computed both at the pixel level, and at the global level. The final prediction is chosen according to a predicted confidence score, and is then iteratively refined in a way inspired from DeepIM [8].

Depth Estimation The recent literature mostly leverages CNN architectures and representation learning methods to predict the depth field of an image [3, 5, 4]. Following the recent success of Transformers [13] for computer vision tasks, Yang *et al.* proposed an interesting Transformers-based method called TransDepth [17] which achieved near state of the art results on the NYU Depth V2 dataset [11].

3. Method

3.1. Depth Estimation Network

As previously discussed, the goal of the project is to estimate the pose of an object from a RGB image. A first straight idea is to use a depth estimation network before the DenseFusion pipeline, and I chose the TransDepth network [17] which achieved state of the art results on the NYU Depth dataset [11] which contains indoor scenes just as the LINEMOD dataset [6].

To predict the pose of an object from an input RGB image, the depth channel is estimated by the TransDepth network, and the prediction is then bootstrapped to create a pseudo RGB-D image on which DenseFusion produces the final pose estimation.

3.2. DenseFusion adaptation to RGB images

The second approach consists in modifying the DenseFusion framework to directly estimate the pose of an object from RGB images. This approach aims to keep the same architectural principles of the RGB-D pipeline and to transpose them to a more challenging scenario where only RGB images are available. The absence of the depth channel brings two main differences compared to the initial architecture.

Firstly, the point cloud embedding cannot be computed anymore. However, we still want to keep pure geometric features which are complementary to appearance features. Thus, I replaced it by a 2D embedding, which might seem naive, but allows the integration of geometric features encoding the position of the target object in the 2D image.

The second main change is the final iterative refinement stage, which cannot be kept as well since the 3D point cloud is no longer available. A possible way to adapt it, is to project the target object from its estimated position predicted by the main network, onto the camera frame using the parameters of the camera. Then, the prediction is corrected by a residual pose estimator network iteratively.
Since the objective of this iterative refinement is to refine locally the prediction, I decided to keep the threshold of 1.3 cm of average distance between the ground truth and the prediction to start the training of the residual pose estimator. Removing or increasing this threshold would perturb the learning, making it less stable or potentially impossible.

4. Experiments

4.1. Reproduction of DenseFusion results

I used the implementation [1] of the authors, and retrained the model for 150 epochs. I also trained the iterative refinement network for 60 epochs, but it was not sufficient to improve the results. The corresponding training curves are available in Appendix C, and the table 4.1 compares the results on LINEMOD [6] of my trained model with those obtained with a pretrained checkpoint. The comparison is made using the Matching score [7], which validates a prediction if its ADD metric is lower than 10% of the diameter of the object. I did not manage to reproduce the results for symmetric objects because of a library using *Cython* that was triggering conflicts inside my environment.

[1] https://github.com/j96w/DenseFusion

Object	Author's checkpoint		My trained model
	per-pixel	iterative	per-pixel
ape	72.9	93.2	66.6
bench vi.	82.4	94.2	85.2
camera	77.0	96.6	77.1
can	82.2	94.0	86.7
cat	87.0	96.8	86.1
driller	69.3	87.9	77.0
duck	71.6	93.5	66.0
hole p.	67.3	92.9	69.1
iron	90.0	97.8	95.1
lamp	85.6	97.2	92.5
phone	76.2	95.7	86.8
MEAN	78.3	94.2	80.6

Table 1. Matching score of RGB-D DenseFusion on LineMOD The iterative approach includes the refinement stage, while the per-pixel does not

The table shows that we managed to reproduce the results of the authors for the per-pixel approach, and it also highlights the importance of the iterative refinement stage, which allows to gain 10 to 20 points of accuracy.

4.2. TransDepth + DenseFusion RGB-D

I used the code[2] of the authors and a checkpoint trained on the NYU dataset [11] and enriched it to predict the depth of the RGB images from the LINEMOD dataset. I also coded all visualization functions. The table 4.2 presents the Root Mean Squared error of the distances between the predicted depth d_i^{pred} and the ground truth depth d_i^{gt} over all pixels : $RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{n} d_i^{gt} - d_i^{pred}}$. The Masked LINEMOD dataset contains the images of the LINEMOD dataset where only the area corresponding to the target object is kept. Indeed, only the depth measurements associated to the segmentation masks are used in the model.

Dataset	RMS	Average distance (cm)
NYU Depth V2	0.365	13.3
LineMOD	0.842	70.9
Masked LineMOD	0.538	28.9

Table 2. TransDepth checkpoint results

We observe that there is an important gap between the results on the NYU Depth dataset and the LINEMOD dataset, even if there is an important improvement between the whole images and the masked ones.

With such an important gap between the ground truth and the prediction, the ADD falls without surprise to 0 when one

[2] https://github.com/ygjwd12345/TransDepth

directly bootstraps them as input depth channel for DenseFusion.

Inspecting the data and the predictions allows the observation of several pitfalls :

- sensitivity to contrast, important lightning and darkness which confuse the network. The figure 2 is an interesting example where the screen of the computer is predicted as a distant object, while the shadow of the table on the ground is predicted closer than the table itself. Likewise, the predictions of the black and white areas of the study support are very different while at the same level. This results in a global overestimation of the amplitude of the depth field.

- On the whole, the network struggles to predict the correct absolute distances, but performs pretty well in estimating the relative depth of objects between them. See Figure 3 where depth images look alike with their own depth scale, but are actually totally different when displayed with the same color scale.

If the contrast problematic is not relevant to our situation since we are only interested in the masked depth field where the contrast problems do not occur, there is an important scale problem that is likely to explain the poor performance of TransDepth on LineMOD. Since the NYU Depth and the LINEMOD datasets have been built using a Kinect camera sharing the same parameters, several other parameters like the content of the images, the point of view or the lightning conditions, could explain the impossibility to transpose the performance on a dataset to the other.

A *ad hoc* solution could be to build a corrective function to reduce the scale error by comparing the true depth field with the predicted one. However, such a supervised and *ad hoc* approach is not relevant regarding the scenario where it is impossible to measure the depth field, because if one cannot get the depth field, then one could not build such a corrective function.
For these reasons, I decided not to develop this approach.

4.3. RGB DenseFusion

I modified the authors' code according to the method described in 3.2 and trained the adapted RGB DenseFusion model for 170 epochs. The corresponding training curves are available in Appendix C. The table 4.3 summarizes the matching score and the ADD obtained by each method on the objects of LINEMOD.

A first remark is that since the refinement threshold was not reached during the training, I eventually did not train the adaptation of the iterative refinement presented in section 3.2. We observe that the final matching scores are very low compared to the RGB-D pipelines, and the differences between objects are mainly due to the different diameters

Object	RGB		RGBD iter		RGBD per-pixel	
	Match	ADD	Match	ADD	Match	ADD
ape	1.5	5.1	72.9	0.9	93.2	0.5
bench vi.	8.3	6.3	82.4	1.8	94.2	0.8
camera	1.5	6.7	77.0	1.4	96.6	0.7
can	5.8	5.6	82.2	1.6	94.0	0.7
cat	3.5	5.3	87.0	1.1	96.8	0.5
driller	6.7	6.8	69.3	2.8	87.9	1.5
duck	2.3	3.9	71.6	1.1	93.5	0.6
hole p.	1.3	6.1	67.3	1.5	92.9	0.7
iron	14.8	5.8	90.0	1.7	97.8	0.7
lamp	21.2	4.9	85.6	2.0	97.2	0.6
phone	5.0	5.9	76.2	1.8	95.7	0.7
MEAN	6.5	5.7	78.3	1.6	94.2	0.7

Table 3. Performance of the DenseFusion pipelines

of the objects. Yet, the model reaches an average precision of 5.7 cm which is pretty good considering the challenging setting of RGB images. The Figure 1 allows to better realize the relatively good performance of the RGB pipeline by plotting the matching score against the diameter tolerance.

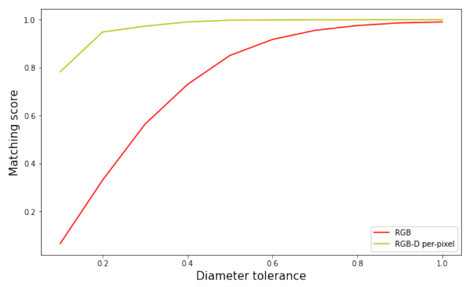

Figure 1. Matching score vs diameter tolerance

From a qualitative point of view, there is an important number of images where the alignment seems to be almost perfect but there is actually an error along the depth, see figure 4. From complex points of view, the model often fails to predict the depth or even the orientation, see figure 5, and such fail cases could not be solved by an iterative refinement stage, which is designed to perform local adjustments.

5. Conclusion

This project explored two extensions of the RGB-D DenseFusion pipeline to perform pose estimation from RGB images. If the first approach relying on a depth estimation network failed to provide interesting results in the absence of an *ad hoc* supervision, the second approach gave satisfying results by directly adapting the extraction of geometric features, even if they remain far from the state of the art results performed notably by keypoints methods.

References

[1] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. 1

[2] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *ECCV*, 2014. 1

[3] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation, 2018. 1

[4] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation, 2020. 1

[5] Vitor Guizilini, Rui Hou, Jie Li, Rares Ambrus, and Adrien Gaidon. Semantically-guided representation learning for self-supervised monocular depth, 2020. 1

[6] 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. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 128:858–865, 2011. 1, 2

[7] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. volume 7724, 10 2012. 2

[8] 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. 1

[9] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017. 1

[10] Mahdi Rad and Vincent Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth, 2018. 1

[11] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Computer Vision, ECCV 2012 - 12th European Conference on Computer Vision, Proceedings*, number PART 5 in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 746–760, 2012. 12th European Conference on Computer Vision, ECCV 2012 ; Conference date: 07-10-2012 Through 13-10-2012. 1, 2

[12] Alykhan Tejani, Danhang Tang, Rigas Kouskouridas, and Tae-Kyun Kim. Latent-class hough forests for 3d object detection and pose estimation. In *ECCV*, 2014. 1

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. 1

[14] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion, 2019. 1

[15] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes, 2018. 1

[16] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2018. 1

[17] Guanglei Yang, Hao Tang, Mingli Ding, Nicu Sebe, and Elisa Ricci. Transformer-based attention networks for continuous pixel-wise prediction, 2021. 1

A. TransDepth pitfalls

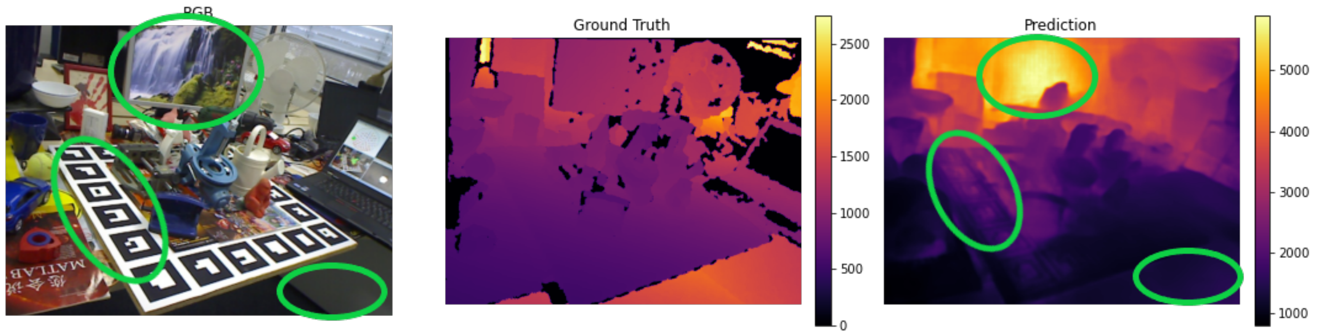

Figure 2. Contrast and lightning sensitivity

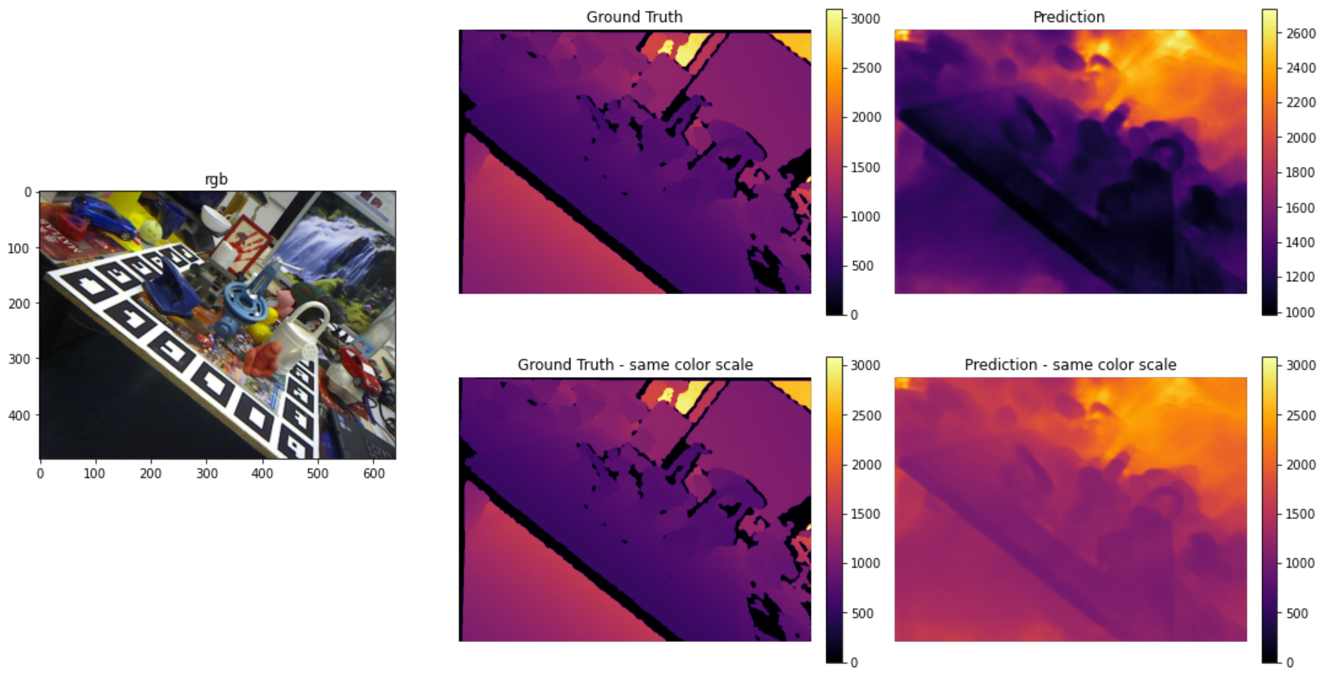

Figure 3. Scale error
The relative depth of objects is well predicted but the absolute comparison with the ground truth shows a scale shift

B. Pose Estimations

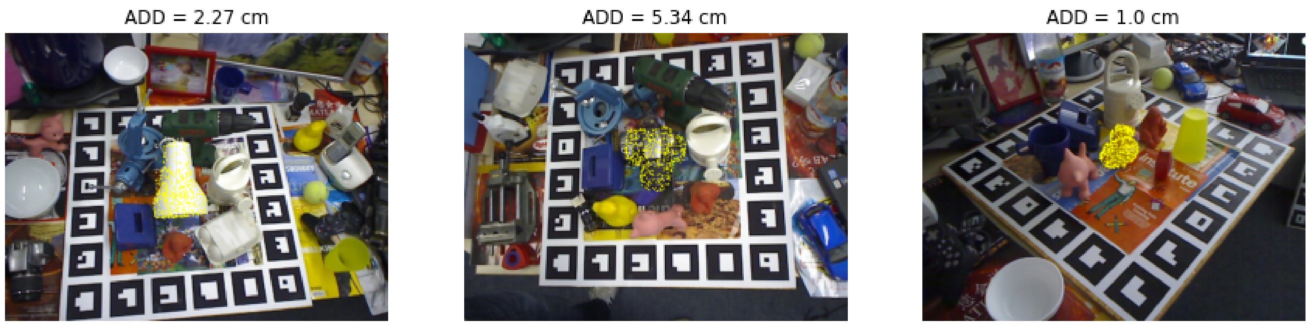

Figure 4. Good 2D estimates

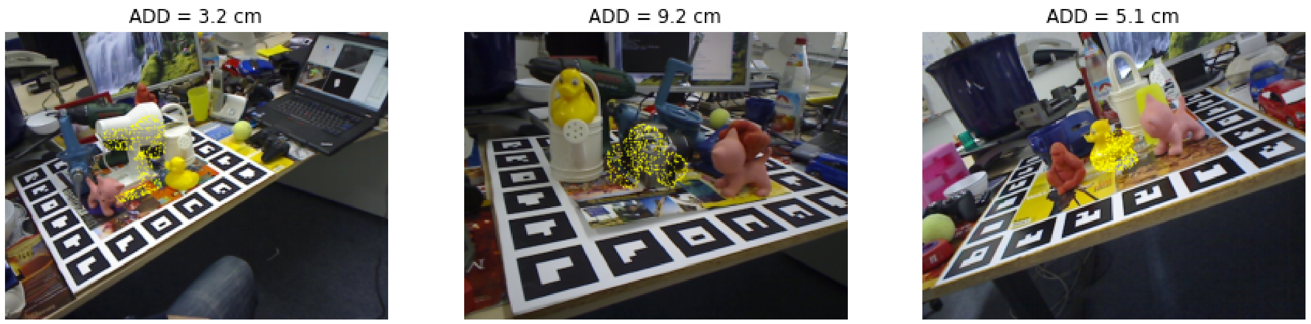

Figure 5. Bad estimates

C. Training curves

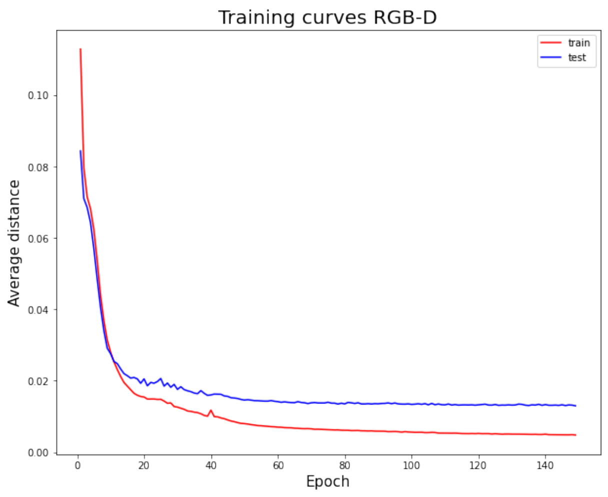

Figure 6. Training curves of RGB-D DenseFusion PoseNet

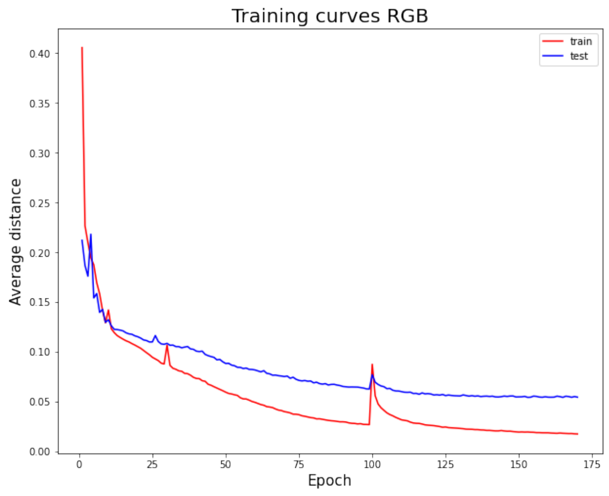

Figure 7. Training curves of RGB DenseFusion PoseNet
NB : The two peaks correspond to resumptions of the training