Category-level 6D Pose Estimation

Master thesis

10th Semester Project

Anna Maria Maj

Title:
Category-level 6D Pose Estimation

Theme:
Master thesis, Computer Vision

Project Period:
Spring Semester 2022

Project Group:
VGIS10 group 1040

Participant(s):
Anna Maria Maj

Supervisor(s):
Kamal Nasrollahi
Jacob Velling Dueholm
Raphael Dũrscheid

Page Numbers: 62

Date of Completion:
June 1, 2022

Abstract:

6D object pose estimation is a computer vision task popularly used in, among others, virtual reality or robotics applications. Currently, it is still an open problem with a lot of space for improvement, especially in the category-level field.
Moreover, a category-level 6D pose estimation algorithms could prove useful in applications developed by Aivero AS[31], which would use them for vision and robotics contexts in real-life scenarios.
This master thesis focuses on exploring the problem by researching the latest developed solutions, public datasets, as well as detection and evaluation methods. Based on the carried out research, FS-Net [8] had been identified as one of the most accurate open-source networks, and therefore was used throughout the project. First, this work attempted to reproduce the results by training the network on NOCS-Real [34] dataset. Next, it was attempted to improve the models produced by network with more focus put on ′mug′ category.
Although this work did not succeed in improving the results, the n°m cm and IoU metrics have been reproduced for ′laptop′ and ′bowl′ category, and to low extent for the ′mug′ category. Furthermore, estimating a 6D pose of non-symmetrical objects is still a challenging task, which could potentially be a good topic for future work.

Preface

The following report is a master's thesis written on the 4th semester of master's programme in Vision, Graphics and Interactive Systems at Aalborg University. This thesis has been written by Anna Maria Maj with collaboration with a company called Aivero AS. The purpose of this thesis is to research the category-level 6D object pose estimation topic and to reproduce and to improve upon the reported performance. Each section documents what have been done during the project.

Any material in this report, which is not cited, is composed by the author. All the code used in this project was ran using Python 3.8 and Anaconda.

I would like to express my deepest appreciation to Kamal Nasrollahi and Jacob Velling Dueholm for time and effort put into this project supervision. It was a pleasure to work with such great supervisors and thank you for all the help and guidance given to me on this thesis.

I would also like express my gratitude to the amazing Aivero team for giving me an opportunity for doing a collaboration project with them. I am very grateful of all the received help with sincere thanks to Raphael Dürscheid, Joao Moreira Alves and Guilherme Mateus.

Finally, I would also like to take this opportunity to thank my family and friends for all the words of encouragement and support which motivated me to finish my Master's degree.

Aalborg University, June 1, 2022

Contents

List of Figures

Chapter 1

Introduction

Estimating 6D poses of objects is an important task in many computer vision applications like augmented reality, virtual reality, or robotics. In Virtual Reality (VR) and Augmented Reality (AR) systems, being able to accurately estimate positions, poses, and geometric relations of objects in the surrounding environment plays an essential role in depicting a precise representation of the real world. On the other hand, in robotic systems 6D pose estimation can be used for pick and place applications as well as for improving the autonomy of the robotic system. [29]

The importance of these tasks in robotics field can be validated by Amazon Picking Challenge (APC)[15]. The goal of APC is to autonomously move items by robotic systems from a warehouse shelf into a tote. This challenge integrates many robotic and computer vision tasks, for instance mapping, motion planning, grasping, or object manipulation, in which estimating the 6D pose of the objects could improve the performance of the applied algorithms. [15, 29] For similar purposes precise 6D object pose estimation can be possible used in robotics applications developed by Aivero AS [31].

The 6D object pose estimation task can be divided into two levels: instance and category. While the instance-level tasks works on predefined object models, the category-level tasks have no specific models provided, which makes establishing the correspondence between an RGB or RGB-D image and the CAD model of the object more difficult. Category-level 6D object pose estimation is much more challenging than its instance-level counterpart, which makes it still a remaining problem in computer vision field. [37]

The aim of the following project is to explore the relatively new field of 6D pose object estimation task, which is the category-level estimation. This thesis will cover the currently available networks, datasets, evaluation and detection methods, as well as potential improvements.

Chapter 2

Background Research

The following chapter will cover a more detailed introduction to the 6D pose estimation problem, as well as will go more in deep into available solutions.

First, the problem of 6D object pose estimation task will be explained more in detail on both instance and category level. Next, the collaboration company will be introduced and their case for using 6D pose estimation task will be described.

After providing the general knowledge about the problem, the available solutions will be covered in separate sections, starting with NOCS[34] network which not only achieved a state-of-the-art performance at the time of publishing, but also created benchmark datasets called NOCS-Realcitenocspaper and CAMERAcitenocspaper. Subsequently the focus will be put on FS-Net[8] which was presented at The Conference on Computer Vision and Pattern Recognition back in 2021 with excellent evaluation and time performance. The last network covered in the Background Research will be 6D-Vit[37], which is one the newest networks found by the thesis's author which outperforms the FS-Net[8] in the metrics used for 6D pose estimation tasks.

Eventually the publicly available datasets used by just mentioned networks will be described and a final summary of the background research will be presented.

2.1 6D Object Pose Estimation

According to Sahin et.al [29], 6D object pose estimation is a task in realm of computer vision which is used to determine the 3D position and 3D orientation of an object in camera-centered coordinates. The visualization of the problem can be seen in figure 2.1. This problem can be addressed at two levels: instances and categories. The instance-level 6D object pose estimation requires the same statistical distribution for both the training data as well as for the testing data, while category-level 6D pose estimation continuously faces challenges such as distribution shift among training and testing data. For that reason instance-based methods cannot be used for category-based problems, as the would not be able to generalize the problem well enough. [29]

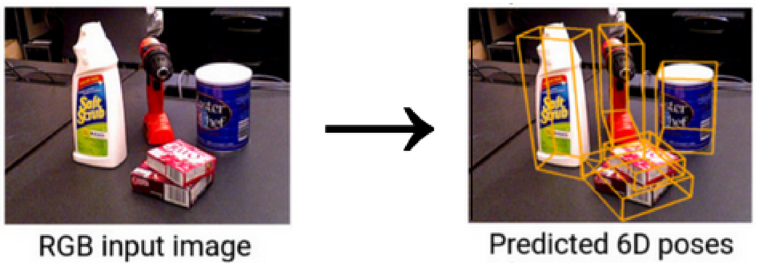


Figure 2.1: An example of how the 6D pose can be visualized. [5]

The 6D object pose estimation methods need to cope with common challenges in order to achieve a high performance. These challenges can be again divided into instance-based and category-based, however, it should be recalled that instances' challenges can also be observed when working on category-level pose estimation, but not the other way around. The common challenges of instance-based 6D object pose estimation are:

- *Viewpoint variability* - As the pose space gets wider, the amount of data required for training a 6D estimator increases, in order to capture reasonable viewpoint coverage of the target object.
- *Texture-less objects* - when objects are texture-less, the discriminative property disappears, thus making methods strongly dependent on depth channel in order to estimate 6D poses of objects.
- *Occlusion* - occurs when an object of interest is partly or completely blocked by other objects present in the scene. The existence of severe occlusion gives rise to false positive estimations, degrading methods' performance.
- *Clutter* - associated with complicated image backgrounds, in which existing objects of interest cannot even be detected by naked eye.
- *Similar-Looking Distractors and Similar-Looking Object Classes* - In case the similarity is in depth channel, 6D pose estimators are strongly confused because of the lack of discriminative selection of shape features. Features from the RGB space can then be used to attenuate this effect. [29]

Meanwhile the common challenges of category-based 6D object Pose Estimation are:

- *Intra-class variation* - Even though the instances from the same category can share similar properties (like texture and color variations in the RGB channel and geometry and shape discrepancies in depth channel), they are not exactly the same. This intra-class variation problem is handled during training by using instance data belonging to the category domain. An example of different intra-class variation can be seen in figure 2.2.

- *Distribution shift* - Objects in the target domain (e.g. the objects used for training) are different than that are of the source domain (e.g. the object used for testing), there is a shift between the marginal probability distributions of these two domains. This shift is also influenced by the fact that the instances in the target domain are unseen to the 6D pose estimator. [29]

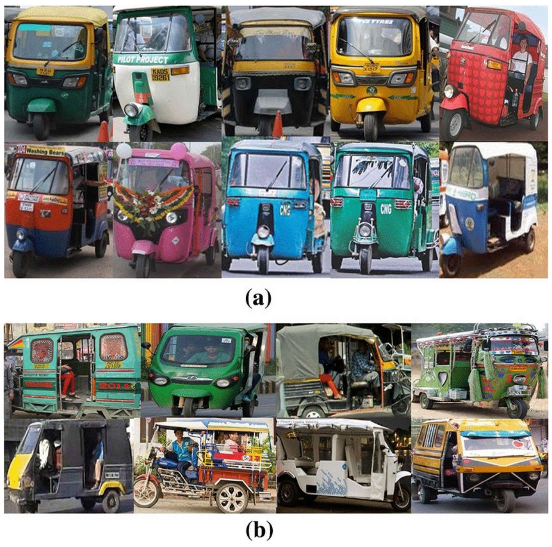


Figure 2.2: Examples of intra-class variation on vehicles from Auto-rickshaw dataset (a) Colour variation (b) Shape variation [3]

2.2 Aivero AS

Aivero AS is a Norwegian company with offices located in Aalborg (Denmark) and Stavanger (Norway) and was founded in 2017. It realizes RGB-D media management and robotics platforms by offering online 3D based cloud services using proprietary depth video compression and streaming technology. Their AI services can be divided into two sections: RGB-D media acquisition/streaming and robotic solutions. [31]

Aivero is interested in using 6D Object Pose estimation algorithms to instruct a collision avoiding path planning algorithm , and in the future for constructing a semantic world model for a robot. The precise estimation of the 6D poses would help determining the relationship between objects that would be picked up by a robot, creating a semantic model of the environment. An example scenario could be finding and estimating the 6D poses of objects like screws and boxes placed on a table, in order to find their relationship between each other (e.g. an empty box is on the table, screws are inside the box etc), so that the robot could perform certain commands such as empty the box or plan the trajectory.

Moreover, knowing the exact 6D pose of an object would help the robot with finding precise grasping points. Realizing this task on a category-level would make the algorithm better at generalizing the problem, as the objects that are supposed to be picked up by a robot could fall into the same category, but they do not necessarily need to represent the same instance. Following the example presented above, that could be, for instance, different types of screws for which the network would be able to estimate the 6D pose without being trained on all of them.

Furthermore, the solution proposed to Aivero should be able to work in real-time, as the robot would need to perform certain tasks in real-life scenarios.

2.3 Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation

The goal of the paper published by Wang et. al [34] is to estimate the 6D pose and dimensions of unseen object instances in an RGB-D image. In order to make the solution work on a category-level the authors propose an approach using Normalized Object Coordinate Space (NOCS) which is defined as a space "in which all objects are contained within a common normalized space, and all instances within a category are consistently oriented". [34]

Moreover, the authors of this method proposed two new datasets: a spatially context-aware mixed reality dataset called *CAMERA* (section 2.6.2) and fully annotated real-world dataset for training and testing called by the authors the *Real Dataset* (in this project for easier reference called *NOCS-Real dataset*, for more detailed description see section 2.6.3). [34]

It can be stated that this is one of the most important category-level 6D pose estimation solutions, as the dataset created along with the paper became a standard for training and evaluating this task. Furthermore, the authors shared their code and dataset publicly as a GitHub[16] repository, which makes it a potentially good starting point for further improvement. It is also worth noting that this solution was presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) back in 2019.

2.3.1 Normalized Object Coordinate Space (NOCS)

The NOCS can be defined as a 3D space contained within a unit cube i.e., x, y, z $\in [0, 1]$. The first step is to create a shape collection of known object CAD models for each category. Next, each model is normalized by their size by uniformly scaling it such that the diagonal of its tight bounding box has a length of 1 and is centered within the NOCS space (see Figure 2.3). Moreover, the object center and orientation is consistently aligned across the same category. In this solution models from ShapeNetCore[6] are being used which have been already normalised for scale, position, and orientation. To see an example of canonical shapes for camera category look at figure 2.3. [34]

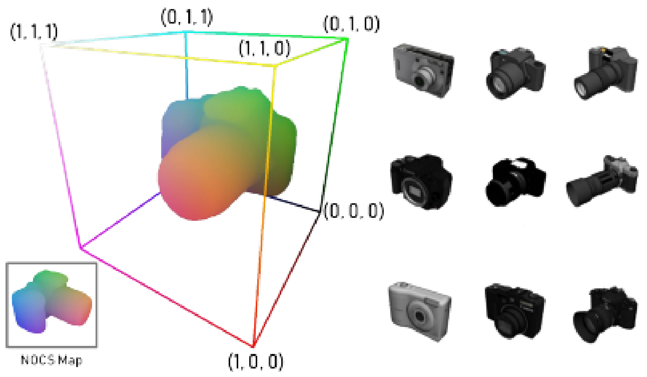


Figure 2.3: Example of NOCS. Each (x, y, z) position in the NOCS is visualized as an RGB color tuple. The network is trained on NOCS map which is a perspective projection of the NOCS on the RGB image (can be seen in the left bottom corner). At the test time, the network regresses the NOCS map which is then used together with the depth map for 6D pose and size estimation. [34]

2.3.2 Overall architecture

In figure 2.4 the architecture of the proposed network is presented with color and depth images as an input. The core of the proposed method is a convolutional neural network (CNN) based on Mask R-CNN[17] framework which estimates the object class, instance mask and a NOCS map of multiple objects from a single RGB image. The reason for not using depth data in the CNN is to exploit existing RGB datasets in order to improve the algorithm's performance. [34]

In this step the NOCS map captures the normalized shape of the visible parts of the object by predicting dense correspondences between object pixels and the NOCS. Then CNN is used to estimate the NOCS map by formulating it as either a pixel regression or classification problem. Finally, the generated map is used with depth map to find the full metric 6D pose and size of the objects using a pose fitting method. [34]

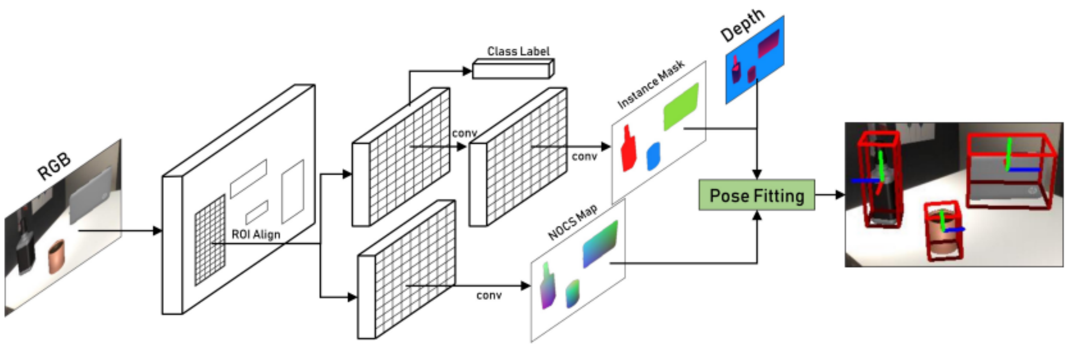


Figure 2.4: Architecture of the proposed network. [34]

2.3.3 Experiments and Results

Metrics

In order to evaluate both 3D object detection and 6D pose estimation different metrics have been used. To evaluate 3D detection and object dimension estimation, the authors used intersection over union (IoU) metric. To evaluate 6D pose estimation, the average precision of object instances is used for which the error is less than *m* cm for translation and *n* degrees for rotation (m°n cm). For more detailed explanation look at section 4.4.

In this evaluation symmetric object categories (like bottle, bowl or can) are allowed freely rotate around the object's vertical axis with no penalty. Moreover, the authors performed special processing for the mug category by making it symmetric when the handle is not visible. [34]

Experiments

At the time of publishing the paper the authors did not know any other methods for category-level 6D pose and size estimation, so they built their own baseline to compare the performance. In category-level 6D object pose and size estimation they ran two trainings:

- On CAMERA dataset only - this model achieved mean average precision (mAP) of 83.9% for 3D IoU at 50% and an mAP of 40.9% for the (5°5 cm) metric when evaluated on CAMERA test set
- On a combination of CAMERA, the real-world dataset (REAL) and with weak supervision from COCO [23] - this model achieved mAP of 76.4% for 3D IoU at 50%, an mAP 10.2% for the (5°5 cm) metric, and an mAP of 23.1% for (10○ , 5 cm) metric when evaluated on REAL test set. This experiment showed that

by learning to predict the dense NOCS map the developed algorithm can provide detailed information about the object's shape, parts and visibility, which are all essential for correct 6D pose and size estimation. The visualization of the evaluated model can be seen in figure 2.5. [34]

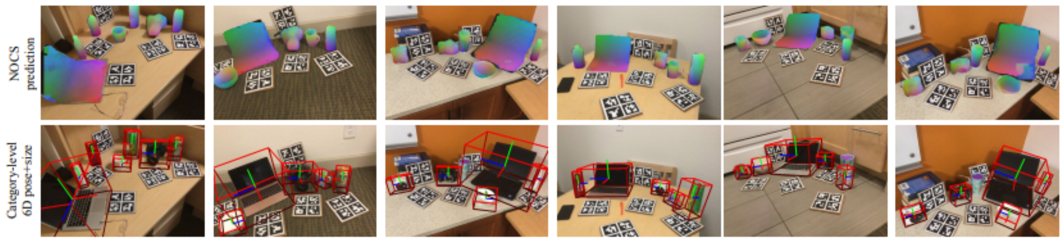


Figure 2.5: Qualitative result on REAL test set. Top row shows the predicted NOCS maps. Bottom row shows the predicted 6D pose (axis) and size estimation (red bounding box). [34]

2.4 FS-Net: Fast Shape-Based Network for Category-Level 6D Object Pose Estimation

FS-NET is a fast shape-based network proposed by Chen et. al [8] which can efficiently extract category-level features for 6D pose estimation. The following section will focus on the architecture of the network as well as on the evaluation results.

At the moment of publishing the paper the FS-Net achieved a state-of-the-art performance, being not only one of the best, but also one of the fastest 6D pose estimation methods. Moreover, this solution has been presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) in 2021. This network is also a good starting points for further investigation and improvement, as the authors have published their code as a GitHub[16] repository.

2.4.1 Proposed method

The architecture of the FS-Net can be divided into five parts: Object Detection, Shape-based network, Decoupled Rotation Estimation, Residual Prediction Network and 3D Deformation techniques. Each of these steps will will be described more in detail in the following section. The overall architecture can be seen in figure 2.6.

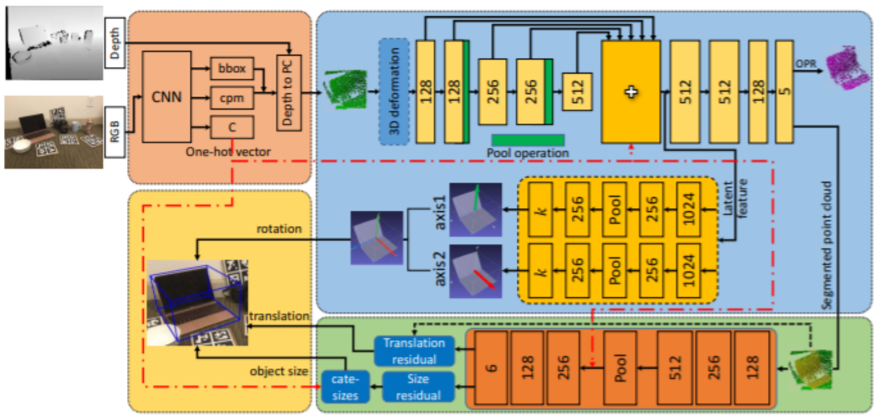


Figure 2.6: Architecture of FS-Net. The input of the network is an RGB-D image. The 2D detector is applied to the RGB channels where the target object is being found. Then using the found information and depth channel, the points in a compact 3D sphere are generated. Next, the 3D deformation mechanism are applied to the found points. After that, the shape-based 3DGC is being used for orientation latent feature learning. Afterward, the rotation information is decoded into vectors. Finally, a residual estimation network is used to predict the translation and size residuals. [8]

Object Detection

In order to detect the object location and find the category label in the RGB images a YOLOv3 [26] model has been trained. This detector detects the object 2D location, category label and class probability map (which generated the 3D sphere center via maximum probability location and camera parameters). Then a 3D sphere is being adopted which is used to locate the point cloud of the found object. This methods differ from others category-level 6D object pose estimations algorithms as it does not focus on finding a semantic segmentation masks of the objects, but instead is looking for their boundaries only. With this solution the object detection step is much faster than in the previous methods. [8]

Shape-based network

The output generated by the object detection step contains both object and background points, and in order to proceed to the next steps the points belonging to the target object need to be found. [8]

Most of the other network architectures operate on point coordinates directly, which can result in the point clouds being shifted or changed in size which would have a negative effect on the pose estimation accuracy. An estimated translation can be used to handle the point clouds shift problem by aligning the point clouds to a local coordinate space. Examples are Frustrum-PoinNet [27] and G2L-Net

[10]. Even though using this method will most likely prevent the point clouds from shifting, the intra-class size variation still remains a problem. [8]

Therefore, in order to solve the encountered problems, the authors propose a 3DGC (3D Graph Convolution) [24] auto-encoder to extract point cloud shape features for segmentation and rotation estimation. This method aims to deliver deformable 3D kernels, whose shape and weights are learnable during the training stage. In addition, similarly as in standard CNN, the network performs a graph max pooling operation in order to summarize the processed features across different scales. As a result, the model is able to observe and extract structural information of unordered 3D point clouds with arbitrary shape and size, which also, as confirmed by the later evaluation, is invariant to 3D point cloud shift and scaling changes. [8][24]

One of the main tasks of the mentioned autoencoder is to encode orientation information in the latent feature. In order to extract the latent rotation feature, the autoencoder is trained to reconstruct the observed points transformed from the observed depth map of the object. According to the authors there are several advantages of this method:

- The reconstruction of observed points is view-based and symmetry invariant
- The reconstruction of observed points is easier than that of a complete object model
- More representative orientation feature can be learned. [8]

In order to train the network the authors utilize Chamfer Distance and the reconstruction loss function (Ł$_{rec}$) is defined as follows:

$$L_{rec} = \sum_{x_i \in M_c} \min_{\hat{x}_i \in \hat{M}_c} ||x_i - \hat{x}_i||_2^2 + \sum_{\hat{x}_i \in \hat{M}_c} \min_{x_i \in M_c} ||x_i - \hat{x}_i||_2^2 \tag{2.1}$$

where:

- M_c is ground truth point cloud
- x_i are points in M_c
- $\hat{M}_c$ is reconstructed point cloud
- $\hat{x}_i$ are points in $\hat{M}_c$. [8]

With the help of 3D segmentation mask, only the features extracted from the observed object points are used for the reconstruction.

Finally, the encoder learned the rotation-aware latent feature as well as the scale and shift-invariant orientation feature under the corresponding rotation. [8]

Decoupled Rotation Estimation

Once the latent feature containing the rotation information is found, it is time to decode the category-level rotation features. In order to do that, two decoders are utilized in decoupling manner which decode the rotation information into two perpendicular vectors under the corresponding rotation. Each decoder extracts the orientation information along corresponding vector, so there is no need to estimate the complete rotation of the object. The representation of the object rotation can be seen in figure 2.7. [8]

The loss function of the decoders (L_{rot} is based on cosine similarity and is defined as follows:

$$L_{rot} = \frac{\langle \hat{v}_1 v_1 \rangle}{||\hat{v}_1|| \times ||v_1||} + \lambda_r \frac{\langle \hat{v}_2 v_2 \rangle}{||\hat{v}_2|| \times ||v_2||} \tag{2.2}$$

where:

- $\hat{v}_1$ and $\hat{v}_2$ are the predicted vectors
- v_1 and v_2 are ground truth vectors
- λ_r is the balance parameter. [8]

Thanks to the use of the balance parameter the network handles the circular symmetry objects well (e.g. bottle) and therefore these objects need only one vector which is their symmetry axis (the green vector in figure 2.7b). For that reason, the balance parameter is set to zero for circular symmetry objects, and for other symmetric objects the rotation mapping function is being used. [8]

In comparison to the other methods, this solution has a geometric meaning, as the rotation representation is based on the shape of the found object and aims to avoid the discontinuity issue which encounter in previous networks. [8]

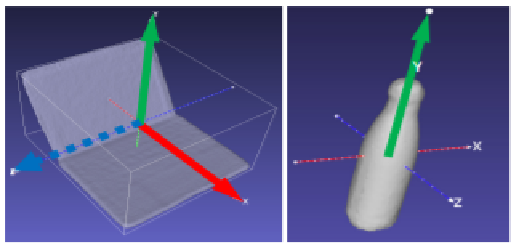


Figure 2.7: (a) Object rotation represented using two perpendicular vector
(b) Circular symmetry object rotation using one vector

Residual Prediction Network

The next step of the solution is to train a small PointNet [28] network which takes the previously segmented point cloud as an input. This step performs two tasks:

- Estimating the residual between the translation ground truth and the mean value of the segmented point cloud
- Estimating the residual between object size and the mean category size

Then the total loss function is found by adding the mean square error losses for translation and size residual. [8]

3D Deformation Mechanism

One of the biggest challenges in category-level 6D-pose estimation is intra-class shape variation. One of the most popular strategies to overcome this issue is to use large synthetic datasets to learn this variation. Even though the method might give satisfactory results, it requires extra hardware resources and increases both the pre-training and training time. [8]

Therefore, in order to avoid using big synthetic data, the authors propose an online box-cage based 3D deformation mechanism for training data augmentation which is based under an assumption that the objects which belong to the same category are similar in shape. Every rigid body has a box-cage predefined (figure 2.8), where each point is assigned to its nearest surface of the cage. [8]

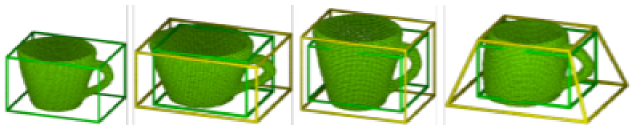


Figure 2.8: New data generated by the newly introduced mechanism. The most left mug is an original point cloud with original 3D box-cage. The three left one are deformed point clouds where the yellow boxes stands for deformed point clouds and green one stands for the initial box cage. [8]

Before performing the 3D deformation the points are first transformed to the canonical coordinate system and eventually they are transformed to the global scene using following following formula:

$$\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n\} = R(\mathbb{F}_{3D}(R^T(P - T))) + T \tag{2.3}$$

where:

- $\mathcal{P}$ are points generated after 2D detection step
- R and T are the pose ground truth
- $\mathbb{F}_{3D}$ is 3D deformation (e.g. case enlarging, shrinking)
- $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n\}$ are newly generated training examples. [8]

2.4.2 Experiments

Implementation

All of the experiments were run on PC with i7-4930K 3.4GHz CPU and GTX 1080Ti GPU. The first step is to locate the object in RGB images by fine-tuning the YOLOv3 [26] pre-trained on COCO dataset [23] with the training dataset. Next step is to train the 3DGC autencoder and residual estimation network. The total loss function looks as follows:

$$L_{Shape} = \lambda_{seg} L_{seg} + \lambda_{rec} L_{rec} + \lambda_{rot} L_{rot} + \lambda_{res} L_{res} \tag{2.4}$$

where λs is the balance parameters set empirically as 0.001, 1, 0.001 and 1 in order to keep different loss validation at the same magnitude, and Ls are loss functions. Moreover, there is a cross entropy used for 3D segmentation loss function L_{seg}. [8]

Moreover, the authors adopt Adam [20] to optimize the network and set the maximum epoch to 50. The initial learning rate is 0.001, and it is halved every 10 epochs.

FS-Net Evaluation

The FS-Net was evaluated on two datasets: NOCS [34] (see section 2.6.3) for category-based 6D pose object estimation using 3D $IoU_{25,50,75}$ and $n°m$ cm metrics and LINEMOD [18] (see section 2.6.1) for instance-level 6D object pose estimation using ADD-(S) metric. [8]

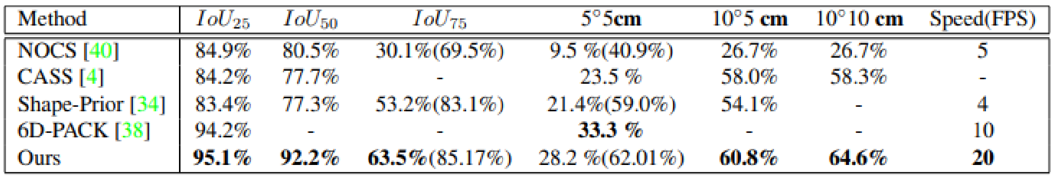
The ADD is a metric used to measure averaged distance between points transformed using the predicted and ground truth pose. ADD-S is a modified ADD metric used to evaluate symmetrical objects.[21] For more detailed explanation of 3D $IoU_{25,50,75}$ and $n°m$ look at section 4.4.

2.4.3 Results

According to the paper, FS-Net achieved a state-of-the-art performance. The authors claim that:

- The network shows robustness to the size of the training dataset. The tests ran by the authors shows that even when only 20 % dataset is being used the network is still able to achieve high performance.
- The network has good category-level extraction ability, as the 3D deformation mechanisms created by the authors significantly improves the results.
- The experiments showed that the FS-Net achieved better pose estimation results using simpler reconstruction tasks when compared to the other reconstruction types used by other methods. [8]

The results generated by FS-NET were then compared to four other networks: NOCS [34], CASS [7], Shape-Prior[32] and 6D-PACK [33] and all of them were evaluaed on NOCS-REAL dataset. According to the tests, the FS-NET outperforms other networks on both accuracy and speed while being trained on only real-world data, as compared to the other method which used both real-world and synthetic data for training. The quantitative results of the training can be seen in figure 2.9. [8]



Method	IoU_{25}	IoU_{50}	IoU_{75}	5°5**cm**	10°5 **cm**	10°10 **cm**	Speed(FPS)
NOCS [40]	84.9%	80.5%	30.1%(69.5%)	9.5 %(40.9%)	26.7%	26.7%	5
CASS [4]	84.2%	77.7%	-	23.5 %	58.0%	58.3%	-
Shape-Prior [34]	83.4%	77.3%	53.2%(83.1%)	21.4%(59.0%)	54.1%	-	4
6D-PACK [38]	94.2%	-	-	**33.3 %**	-	-	10
Ours	**95.1%**	**92.2%**	**63.5%**(85.17%)	28.2 %(62.01%)	**60.8%**	**64.6%**	**20**

Figure 2.9: Category-level performance evaluation on NOCS-REAL dataset using different methods. '-' means no results have been reported for this metric, and value in the brackets stands for performance on synthetic NOCS data. [8]

2.5 6D-ViT: Category-Level 6D Object Pose Estimation via Transformer-based Instance Representation Learning

6D-ViT is a two-stream transformer-based encoder-decoder architecture used for 6D category-level object pose estimation developed by Lu Zou and Zhangjin Huang. The proposed network is based on SPD [32] network and is inspired by the newest transformer architectures which shows to be very efficient in computer vision tasks. [37]

2.5.1 Overall architecture

The 6D-ViT network is is built on three sub-networks:

- An encoder-decoder architecture called Pixelformer, used for dense instance representation learning on RGB images. It is used to explore the appearance of an object instance.
- A transformer-based 3D point cloud learning network called Pointformer for instance representation learning on point clouds. Used to explore the geometric characteristics of an object instance
- Multi-source aggregation (MSA) network for joint and dense representation generation and pose estimation

As the final step, the category-level 6D pose is found by establishing the correspondence among dense representations, categorical shape priors, and the observed point clouds. The overall architecture can be seen in figure 2.10. [37]

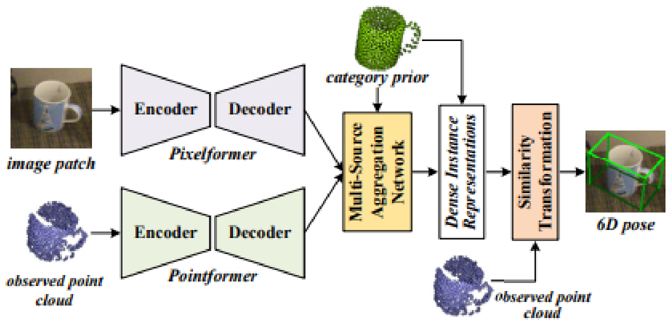


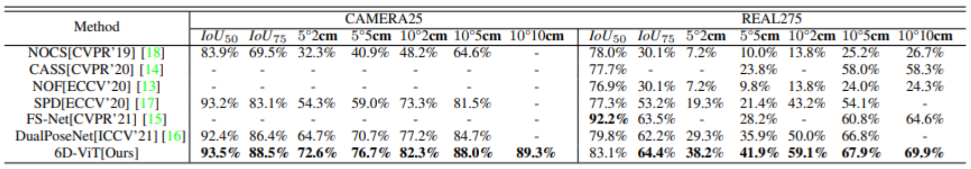
Figure 2.10: The base architecture of 6D-ViT. [37]

2.5.2 Experiments and results

The 6D-ViT network has been trained and evaluated on the two most widely used datasets: CAMERA and REAL proposed by NOCS [34] (for more detailed description see section 2.6.2 and 2.6.3). The results are evaluated on two levels: 3D object detection and 6D object pose estimation. For 3D object detection, the results are reported as average precision at different Intersection-Over-Union (IoU) thresholds. For 6D object pose estimation, the average precision computed as n°m cm is being used. [37]

The performance of the network is compared to with the baseline method: SPD [32], and five other state-ofthe-art methods, including NOCS [34] (section 2.3), CASS [7], NOF [11], FS-Net [8] (section 2.4), and DualPoseNet [22]. [37]

Compared to the results obtained by SPD, the 6D-Vit achieved a remarkable improvement in each category of both datasets. When evaluated on the CAMERA dataset the proposed solution outperfoms other methods for 3D object detection IoU_{50} and IoU_{75} metrics. On the other hand, when evaluated on the REAL dataset, the solution achieved the second best performance score under IoU_{50} metric, which is 9.1% lower than that of FS-Net [8]. However, the 6D-ViT achieved better results under a more difficult metricIoU_{75}, where it outperforms FS-Net[8] by 0.9%. For the 6D pose evaluation metrics the 6D-ViT significantly outperforms the previous best method, DualPoseNet [22]. Moreover, 6D-ViT achieves an average accuracy of 69.9% under the translation/rotation error metric, which is 5.3% higher than the one achieved by FS-Net [8]. The performance results can be seen in the figure 2.11. [37]



Method	CAMERA25							REAL275						
	IoU_{50}	IoU_{75}	5°2**cm**	5°5**cm**	10°2**cm**	10°5**cm**	10°10**cm**	IoU_{50}	IoU_{75}	5°2**cm**	5°5**cm**	10°2**cm**	10°5**cm**	10°10**cm**
NOCS[CVPR'19] [18]	83.9%	69.5%	32.3%	40.9%	48.2%	64.6%	-	78.0%	30.1%	7.2%	10.0%	13.8%	25.2%	26.7%
CASS[CVPR'20] [14]	-	-	-	-	-	-	-	77.7%	-	-	23.8%	-	58.0%	58.3%
NOF[ECCV'20] [13]	-	-	-	-	-	-	-	76.9%	30.1%	7.2%	9.8%	13.8%	24.0%	24.3%
SPD[ECCV'20] [17]	93.2%	83.1%	54.3%	59.0%	73.3%	81.5%	-	77.3%	53.2%	19.3%	21.4%	43.2%	54.1%	-
FS-Net[CVPR'21] [15]	-	-	-	-	-	-	-	**92.2%**	63.5%	-	28.2%	-	60.8%	64.6%
DualPoseNet[ICCV'21] [16]	92.4%	86.4%	64.7%	70.7%	77.2%	84.7%	-	79.8%	62.2%	29.3%	35.9%	50.0%	66.8%	-
6D-ViT[Ours]	**93.5%**	**88.5%**	**72.6%**	**76.7%**	**82.3%**	**88.0%**	**89.3%**	83.1%	**64.4%**	**38.2%**	**41.9%**	**59.1%**	**67.9%**	**69.9%**

Figure 2.11: Quantitative comparisons of 6D-ViT evaluation. [37]

2.6 Available datasets

In this section the most widely used RGB-D datasets in instance and category level 6D pose estimation algorithms will be described. The focus will be put on the datasets used by already covered networks (NOCS [34], FS-Net [8] and 6D-ViT [37]). All of the below covered datasets are publicly available.

2.6.1 LINEMOD

LINEMOD is a RGBD dataset published by Hinterstoisser et. al [18] which is widely used in instance-level 6D object pose estimation challenges. It consists of 15 registered video sequences of 15 texture-less 3D objects. Each object was stuck to the center of a planar board with markers attached to it, which were then used to provide the corresponding ground truth poses. Each sequence contains more than 1,100 real images from different view points. [18]

Do et. al [14] comments, that the objects used in this dataset are poorly textured and are placed in a cluttered scene. The images in these sequences contain multiple objects, but only one object at the time is annotated in the ground truth data. Example images from the sequence can be seen in figure 2.12. [14]

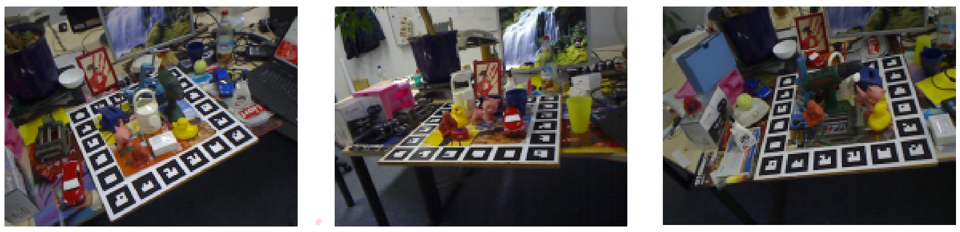


Figure 2.12: Original sample frames from LINEMOD dataset. [14]

2.6.2 Context-Aware MixEd ReAlity (CAMERA)

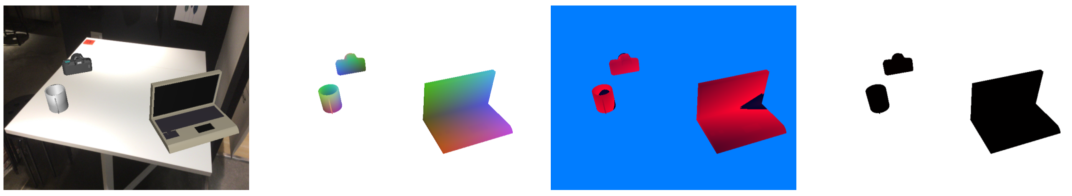
The Context-Aware MixEd ReAlity (CAMERA) is a dataset created by Wang et. al [34] which combines real background images with synthetically rendered foreground objects in a context-aware manner. The synthetic objects are rendered and composited into real scenes with believable physical locations, lighting, and scale. [34]

This dataset consists of real RGB-D images of varying indoor scenes which are then used as a background for the synthetic objects. The authors focused on capturing tabletop scenes with hand-scale objects. In total they collected 553 images for 31 scenes, from which 4 were used for validation. [34]

In order to render only realistic looking objects the authors manually chose the hand-scale objects from ShapeNetCore [6] and eventually ended up using objects from six categories: bottle, bowl, camera, can, laptop and mug. In total the dataset consists of 1085 object instances from which 184 were used for the validation. [34]

The generated objects are placed where they would naturally occur with realistic lighting that mimics indoor conditions. For that purpose the authors used detection algorithm which obtained the pixel-level plane segmentation in real-images. With usage of that algorithm the solution generates random locations and orientations on the segmented plane on which the synthetic object would be placed. [34]

As the results, the dataset consists of 300K composite images with perfect ground truth NOCS maps, masks, and class labels (for explanation look at section 2.3), from which 25K images are set aside for validation. An example of the image taken from this dataset can be seen in figure 2.14. [34]



(a) Color **(b)** Coord **(c)** Depth **(d)** Mask

Figure 2.13: Frame number 0003 from folder 1123 from CAMERA validation set. [34]

2.6.3 NOCS-REAL

This real-world dataset often referenced as NOCS-REAL dataset was created by Wang et. al, which are the same authors as of the CAMERA dataset described in subsection 2.6.2.

This dataset consists of total of 18 real indoor scenes from which 7 for training, 5 for validation, and 6 for testing. All of the scenes were captured by a Structure Sensor. In total the authors collected 8K RGB-D frames, from which 4300 for training, 950 for validation and 2750 for testing. Each scene has at least 5 object instances recorded in order to mimic real-world clutter. [34]

For each of the training and testing subsets there are 6 categories (bottle, mug, bowl, camera, can and laptop) and 3 unique instances per category. For each instance there is a clean and accurate 3D mesh generated using an RGB-D reconstruction algorithm developed by the authors. [34]

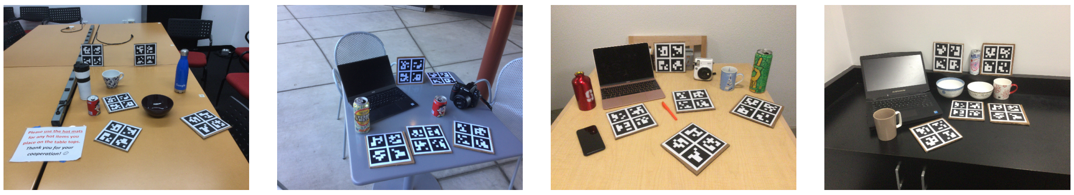

(a) Train 2_0000 **(b)** Train 6_0000 **(c)** Test 2_0000 **(d)** Test 6_0000

Figure 2.14: Sample images from NOCS-Real dataset. As it can be seen, some of the scenes consist of two instances of the same category (image (b) and (d)). [34]

2.7 Summary

One of the newest networks which achieved the highest performance when evaluated on NOCS-REAL dataset[34] is 6D-ViT network[37] described in section 2.5, which was published in October 2021. However, in comparison to NOCS[34] or FS-Net[8], this paper has not been presented at the conference on Computer Vision and Pattern Recognition (CVPR), widely considered one of the leading Computer Vision conferences in the world [12, 30, 13]. Beside this, the authors of the 6D-ViT[37] network state that the performance gap between their and FS-Net[8] method is due to FS-Net[8] using YOLO[19] detection for cropping object instances instead of Mask-RCNN[17]. Finding semantic segmentation masks instead of object bounding boxes leads to a higher IoU_{75} metric, at the cost of slower execution [8]. The authors of 6D-Vit[37] do not state the absolute speed of their method. However, the FS-Net[8] authors claim that due to not using Mask-RCNN[17] as the 2D detection method, FS-Net[8] works four time faster than the NOCS[34] network.

FS-Net[8] is not much older than 6D-Vit[37] (published in June 2021), and also achieved a state-of-the-art performance at the time of publishing. Moreover, the authors of NOCS[34] and of FS-Net[8] networks publicly shared their solutions on Github [16].

The datasets which are the most commonly used in category-level 6D pose estimation tasks are NOCS-Real and Camera dataset described in sections 2.6.3 and 2.6.2[34]. According to Leonard Bruns and Patric Jensfelt, these datasets mostly contain a heavily constrained set of orientations. This simplifies the 6D pose estimation task by removing pose and shape ambiguities. Therefore, there is a need for new larger, high-quality datasets that could be applied to this problem [4].

Nevertheless, NOCS-Real[34] can be considered as a commonly used baseline non-synthetic dataset for category-level 6D pose estimation problems. It is often being combined with CAMERA dataset[34] for training purposes (as in NOCS[34] network), but it can also be used alone, as in FS-Net[8] case.

Chapter 3

Problem Formulation

Based on the research shown above, it can be stated that category-level 6D object pose estimation is still an open problem with lot of space for improvement.

New solutions are being constantly developed and published at high-level conferences. Many authors are very open about their work and share their implementations publicly as Github [16] repositories. Using this kind of open-source work is a one of the possible ways of researching the topic more in depth, and it can be a great starting point for future improvements.

Therefore, for this project it was decided to choose one of the publicly available networks in order to research the topic more. The first objective of this work will be to try to reproduce the authors' paper results and second, to improve them by applying changes to the network or to the dataset used for training.

Based on the summary done for the background research, it can be concluded that one of the best networks that can be chosen for this thesis is FS-Net[8]. It is not only one of the fastest solutions publicly available, but also one of the best in its field. Additionally, the dataset that the solution was trained on has been also shared publicly, which makes it an excellent candidate for a 6D pose estimation research topic.

Chapter 4

Solution overview

The following chapter will describe how FS-Net[8] network works with focus put on dataset used, training, detection and evaluation. Following that, the 2D detection part of the solution will be evaluated.

4.1 Dataset

The following section will focus on describing the used dataset NOCS-REAL([34] more in detail. The authors of FS-Net [8] did not use a validation set for training, and instead only used train and test sets. All of the scenes contained objects placed either on the table or on the floor. The objects are static throughout the scene. The cameraman recorded the scenes by moving to different sides and by changing the angles.

In order to convert the models to a different format as well as to visualize them the PyMesh[35] library has been used. All of the below images which visualize a 3D object were opened using this library.

4.1.1 Train set

The train set consists of 7 scenes which in total is 4372 frames. The sample images taken from each of the scenes can be seen below in figure 4.1

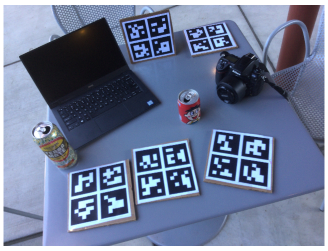
(a) Scene 1, frame 0266

(b) Scene 2, frame 0101

(c) Scene 3, frame 0476

(d) Scene 4, frame 0499

(e) Scene 5, frame 0515


(f) Scene 6, frame 0569

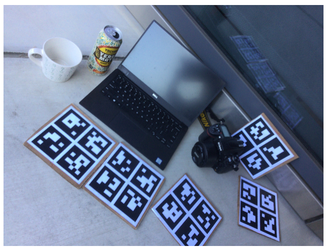

(g) Scene 6, frame 0569

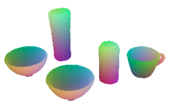
Figure 4.1: Randomly chosen images from each of the train scenes.

For each frame the dataset provides a color image, depth image, masked image and an image consisting of objects in Normalized Object Coordinate Space (NOCS). In addition, there is a meta text file containing the instance index, class index and model name and a pickle file consisting of object labels (an example of how the images look like can be seen in 4.2). The mentioned pickle file consists of following labels:

- *'class_ids'* - array consisting of category class IDs: 1 - bottle; 2 - bowl; 3 - camera; 4 - can; 5 - laptop; 6 - mug
- *'bboxes'* - array consisting of starting and end points of 2D bounding boxes
- *'scales'* - array consisting of 1x3 scales arrays of the 3D bounding boxes
- *'rotations'* - array consisting of 3X3 rotation matrices of the 3D bounding boxes
- *'translations'* - array consisting of 1X3 translation arrays of the 3D bounding boxes

- *'model_list'* - array consisting of names of the models in the scene e.g. *'mug2_scene3_norm'*

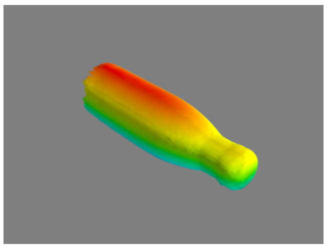
(a) Color image

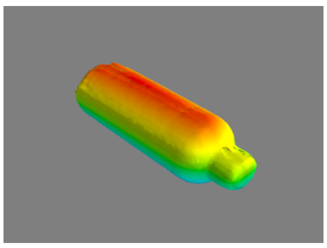

(b) NOCS image

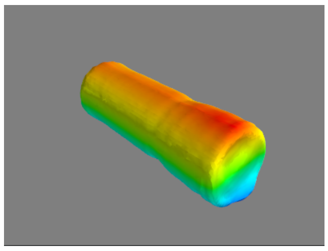
(c) Masked image

Figure 4.2: Three types of images that can be found in the dataset. There is also a depth image for each of the frames. The following example shows frame number 1 from the first scene.

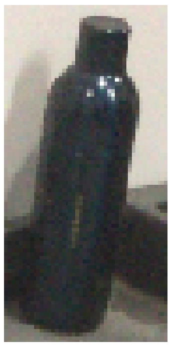
Each scene consists of different models of objects (however, some models repeat in other scenes). There is a total of 18 models, 3 per each category (bottle, bowl, laptop, can, mug and camera). The 3D models along with the example picture of the object in real life can be seen in below figures (4.3, 4.4). Below only bottle and mug categories can be seen, to see all of the models from all categories look at Appendix .1.


(a) *"bottle blue google"* model


(b) *"bottle starbuck"* model


(c) *"bottle3 scene5"* model

(d) *"bottle blue google"* color image


(e) *"bottle starbuck"* color image

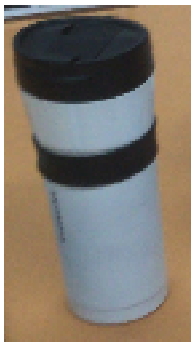
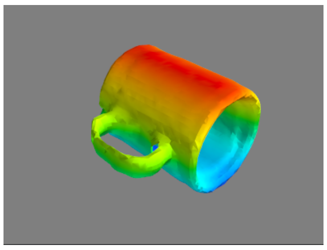
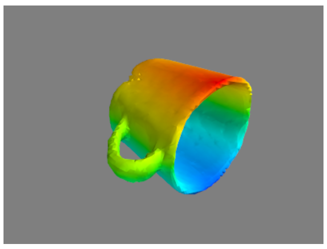
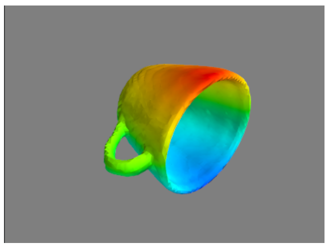

(f) *"bottle3 scene5"* color image

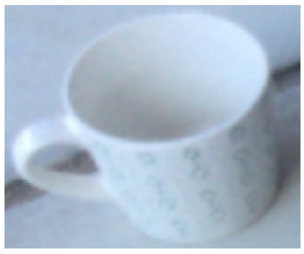
Figure 4.3: Training bottle models and sample color pictures of them.


(a) *"mug vignesh"* model


(b) *"mug white green"* model


(c) *"mug2 scene3"* model

(d) *"mug vignesh"* color image


(e) *"mug white green"* color image

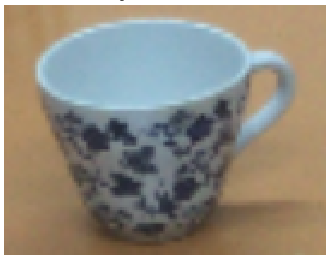
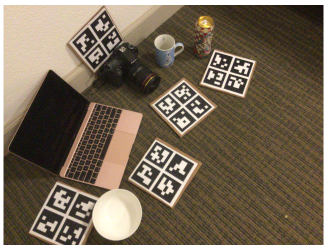

(f) *"mug2 scene3"* color image

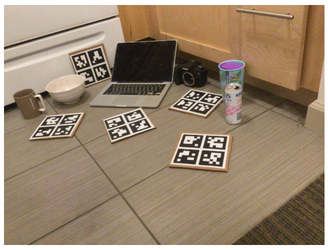
Figure 4.4: Training mug models and sample color pictures of them.

4.1.2 Test set

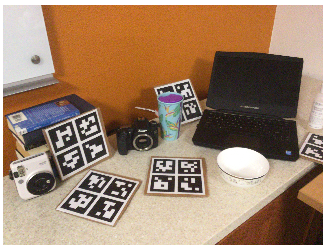
The test set consists of 6 scenes with a total of 2754 images. Samples images from the scenes can be seen in figure 4.5.


(a) Scene 1, frame 0266

(b) Scene 2, frame 0101


(c) Scene 33, frame 0476

(d) Scene 4, frame 0499


(e) Scene 5, frame 0515

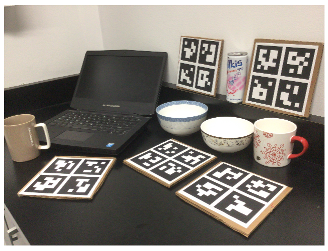
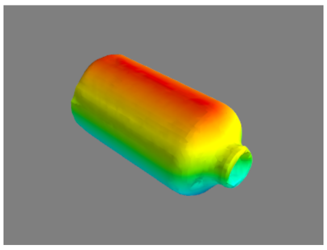
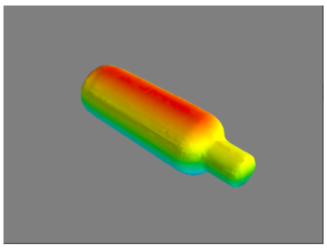
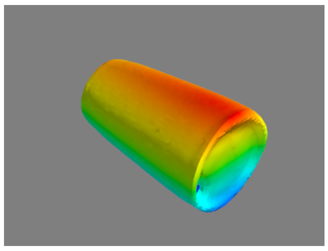

(f) Scene 6, frame 0569

Figure 4.5: Randomly chosen images from each of the test scenes.

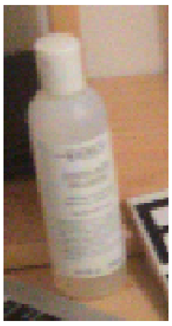
Same as in the train set, there is color image, depth image, masked image, NOCS image, meta file and pickle file for each of the frames. There is a total of 18 models, 3 per each category (bottle, bowl, laptop, can, mug and camera), and none of these models were used for training. The 3D models along with the example picture of the object in real life can be seen in below figures (4.6 and 4.7). Below only bottle and mug categories can be seen, to see all of the models from all categories look at Appendix .2.


(a) *"bottle red stanford"* model


(b) *"bottle shampoo"* model


(c) *"bottle shengjun"* model

(d) *"bottle red stanford"* color image


(e) *"bottle shampoo"* color image

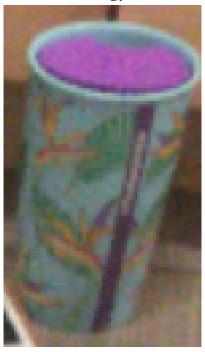
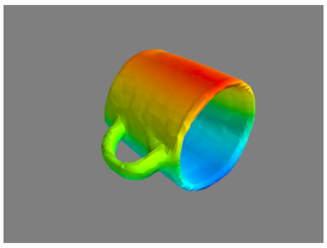
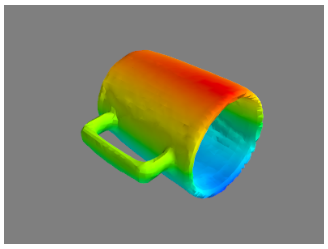
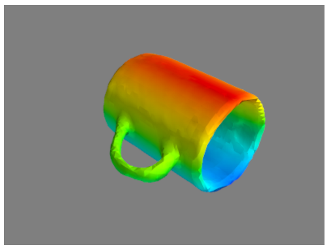

(f) *"bottle shengjun"* color image

Figure 4.6: Test bottle models and sample color pictures of them.


(a) *"mug anastasia"* model


(b) *"mug brown starbucks"* model


(c) *"mug daniel"* model

(d) *"mug anastasia"* color image

(e) *"mug brown starbucks"* color image

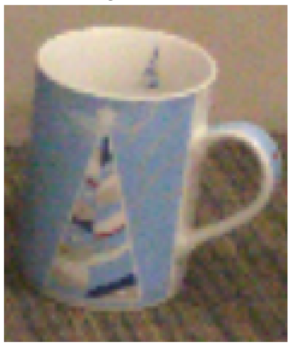

(f) *"mug daniel"* color image

Figure 4.7: Test mug models and sample color pictures of them.

4.2 Training

All of the trainings were run by the thesis's author on ASUS ROG Zephyrus S GX531 with processor i7-8750H, 24GB ram and with a NVIDIA GeForce RTX2080 mobile graphics card. The code used for trainings the models was taken from the publicly available FS-Net[8] code repository on GitHub [1]. The only changes applied to the training itself is changing the batch size from 12 to 8 due to CUDA[25] memory allocation. For various training sessions, the number of epochs was also changed and will be mentioned in the relevant sections.

The FS-Net[8] trains each model separately and generates four PTH files for each of the object categories:

- Rot_g - the weights contained in the estimation of the green vector of the rotation (the green arrow in figure 2.7)
- Rot_r - the weights contained in the estimation of the red vector of the rotation (the red arrow in figure 2.7)
- Seg3D - the weights contained in the 3D graph convolution autoencoder
- Tres - the weights contained translation estimation

[1]Link to the FS-Net Github repository: `https://github.com/DC1991/FS_Net`

These four model files are then combined to achieve 6D pose estimation. For analyzing the training, it was decided to generate graphs which would plot different metrics using TensorBoard [1]. As mentioned before, no validation data was originally used while training. The only data being evaluated that could be plotted in a graph and analyzed were loss functions of: green rotation, red rotation, size, 3D segmentation, reconstruction, residual prediction (sum of size loss and translation loss) and total loss (for equation look at 2.4). Mentioned loss functions are calculated after each step, which is a lot information to be plotted in a graph. Therefore, to analyze the loss graphs easier, the new information was sent to the graph after every epoch. For that purpose the following formula was used after each trained epoch:

$$EpochLoss = \frac{Loss_0 + Loss_1 + ... + Loss_i}{i} \tag{4.1}$$

where i stands for number of iterations in an epoch.

In order to analyze the training more in detail additional metrics were evaluated at the end of each epoch: after each epoch the model generated by the network was used to detect the results using the method described in 4.3. Next, the detected boxes were used for evaluation using the method and metrics described in 4.4. Lastly, the generated metrics were plotted in separate graphs.

4.3 2D Detection

The authors of FS-Net[8] provided an example code of how detection could be applied on the laptop category using pretrained YOLOv5l weights [19] and a provided, pretrained FS-Net model for this category. On the GitHub [16] repository, the FS-Net authors mention that in order to detect different categories, the 2D detect model needs to be retrained for other objects [8]. Although this logic makes sense, the 2D detect model used in this demo should still be able to detect bottle and bowl categories as they are part of the classes that YOLOv5[19] should be able to detect. The same applies to a mug category, if it is assumed that detection of a 'cup' class is good enough to detect a mug.

Therefore, in order to be able to detect other categories the object detector should be fine tuned for the specific object categories. However, training a new 2D detect model would be time-consuming and would not fall into scope of the main topic of 6D pose estimation. Thus, it was decided to focus on detecting the classes which both the YOLOv5[19] model and FS-Net[8] models have been trained for: a laptop (which is already working due to the provided demo), a bowl, a bottle and a mug which would be detected as a cup.

In order to be able to detect other objects as well, the 2D detection part of the code has been modified in such a way that now it detect the objects the same way as in the demo code provided in YOLOv5[19] repository [19]. Once the predictions are found, the code will loop through them and find the detection which match the chosen category. After that, the 6D pose with be estimated and a bounding box will be plotted.

4.3.1 Challenges

During improving the detection part a few challenges have been met. None of them were critical enough to stop the solution from working, however, they might have influenced the results. In order to deal with the challenges the authors of the FS-Net[8] have been contacted on email and on GitHub [16] issue section, but no response was received.

No detection of a camera and a can

As mentioned before, the YOLOv5l [19] model which was trained on COCO dataset [23] is not able to detect all of the categories. The classes that the model is able to find are, among other, a bottle, a laptop, a bowl and a cup (a mug), but it cannot predict that an object is either a camera or a can. Even though they are not necessary to be detected, it would be better if the solution was not limited to just a few objects and was able to detect and evaluate all of the given categories.

Since not all objects can be detected, it is not possible to run a full evaluation. The performance of the FS-Net[8] is compared to the other networks by analyzing the results outputted by all the categories and by taking an average perfomance of them. Therefore, this project will focus on reproducing the results provided for each of the categories, and not on the overall performance of the algorithm.

Mug as a cup detection

Using the cup-based 2D prediction for 6D pose estimation of a mug most often returns results with a high confidence score. However, since the 2D detection is not trained for other categories, sometimes it results in ′can′ models being detected as a ′cup′. This issue was solved to some extent by setting a higher confidence level for 2D detection. Throughout the whole project, the confidence level was set to 0.6 for all of the categories.

Moreover, the model sometimes detect a ′bottle′ as a ′cup′ with a high confidence level. This problem occurs most of the time when the object called *′bottle_shengjun′* is being detected (for reference image look at figure 7f). This however, would be a task difficult to fix even if a new 2D detection model was trained due to the object looking very much like a cup. An example image of the detection of a mug and a bottle being detected as a cup can be seen in figure 4.8. This issue can have a negative outcome, as a bottle is a symmetrical model, and a mug is not, which might influence the rotation performance.

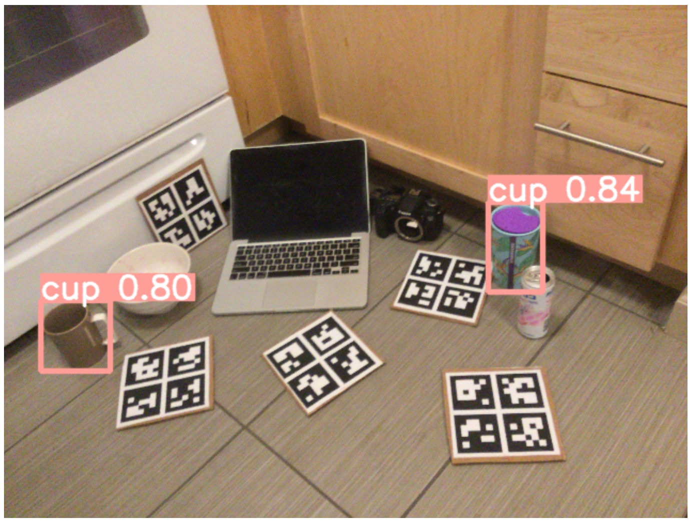
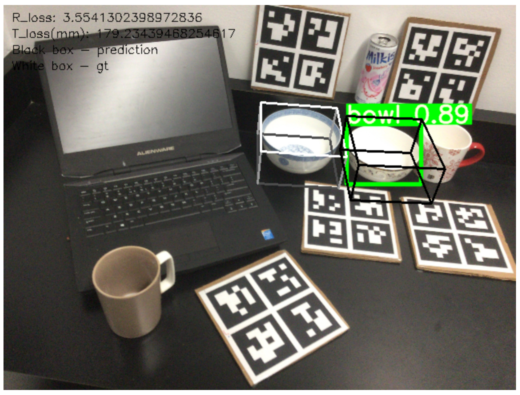


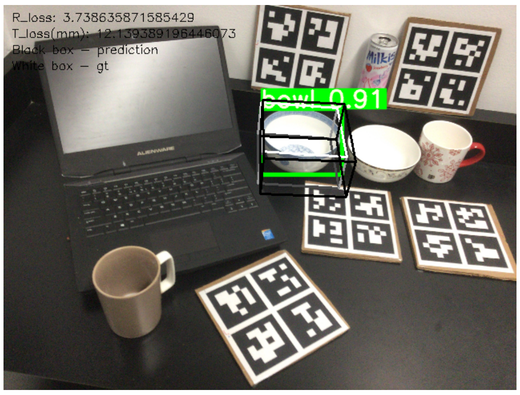
Figure 4.8: A 2D detection applied to frame number 0310 from test scene number 3. As it can be seen, the object which has a 'bottle' category assigned to it (the one on the right) is detected as a cup with a higher confidence level than the mug.

Multi-object detection within the same category

Another issue was a detection of a few objects of the same category in a scene. Even though the solution was able to detect the objects and estimate their 6D pose, it was not able to compare them to the right ground truth data. An example can be seen below, where both of the bowls are detected properly but only one ground truth box is being plotted.



(a) Prediction of one of the bowl object.



(b) Prediction of the other bowl object.

Figure 4.9: Frame number 0169 from test scene number 6. The green square is a 2D detection, white box is the ground truth and black box is the prediction made by FS-Net model.

This problem was solved by writing both ground truth data models and both predicted boxes to the results pickle files, but in such order where the boxes would match each other (e.g. the first written prediction matches the first written ground truth data). This was inspired by the way it was done in Shape Prior Deformation [32], where the authors compare the IoU and transformation matrices of the 3D boxes and pair the best matches together.

To find out which ground truth data matches which prediction an additional code was added which calculated translation loss between each ground truth and prediction boxes. Once the losses were calculated, the pairs with the lowest translations losses were chosen and matched together. An example of the added multi detection can be seen in figure 4.10, which shows the same image as in figure 12 before multi detection was applied. In this project, as compared to just mentioned Shape Prior Deformation network [32], only the translation loss was used as it handles the problem well enough and requires less computation. That way multiple bounding boxes will be evaluated the same as a single one, as it will be described more in detail in section 4.4.

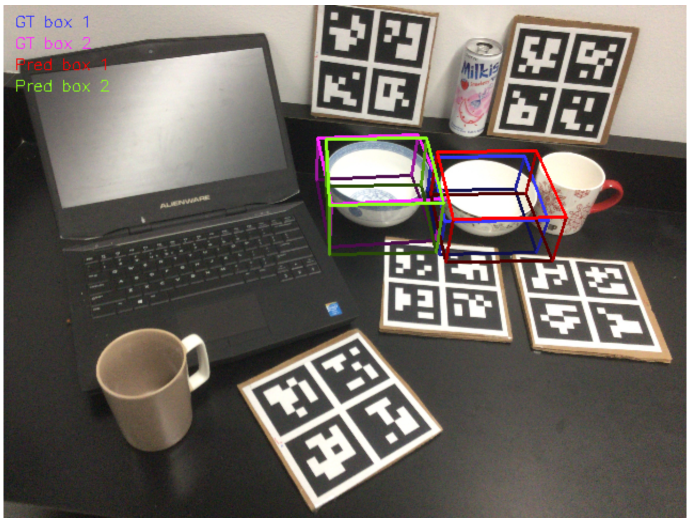


Figure 4.10: Multi detection of *bowl* model run on frame number 0169 from 6th test scene. By looking at the legend it can be seen if the right ground truth boxes are being compared to the right predictions, meaning that the green box should overlap with the pink one, and the red one should overlap with the blue one.

No general model generated

Another challenge is the detection of all the models/objects at the same time. As mentioned in 4.2, the network generates four models for each of the categories, and there is no general model generated which would be able to estimate the 6D poses of all of the categories. This however is not necessary to have at the moment, as the focus of this project is put on specific categories detections, although it would be useful to have in more real-world scenarios.

Not having one model generated was also what caused problems with evaluating the results, as the borrowed evaluation code from Shape Prior Deformation Network [32] uses one model for estimating the 6D pose of all of the categories. This however was managed to be fixed, as it will be described in section 4.4.

4.3.2 2D detection evaluation

In order for the evaluation of the 6D pose estimation not to be influenced by any wrongly made detections, it was decided to evaluate the 2D detection part of the solution separately.

For the evaluation of the 2D detector, precision and recall have been used. To calculate these metrics the script runs through all of the scenes and labels each detection as either:

- True Positives (TP) - correctly predicted objects with at least 0.6 confidence level

- False Positives (FP) - falsely detected objects
- False Negatives (FN) - not detected objects [2]

Once the code was run on all the scenes, it was necessary to go through all of the images plotted with detected bounding boxes and handpick False Positives detections. This step was essential as there is no ground truth data provided for 2D detection, therefore not everything could have been evaluated automatically.

The last step is to calculate the precision and recall based on the labeled detections. The formulas for finding these metrics are as follows [2]:

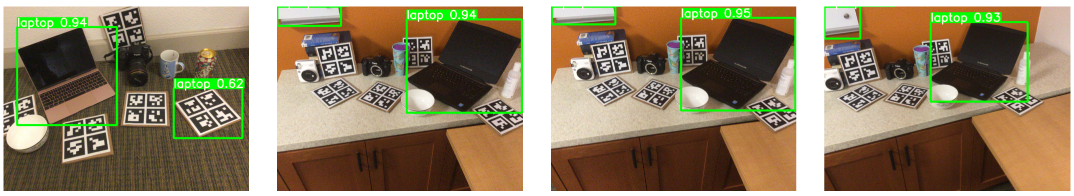
$$Precision = \frac{TP}{TP + FP} \qquad (4.2) \qquad\qquad Recall = \frac{TP}{TP + FN} \qquad (4.3)$$

The evaluation of the performance can be seen in the below table 4.1. It needs to be noted that the evaluation of the 2D detector was run on only three categories (laptop, mug and bowl) since only these categories will have the 6D pose estimated in the next steps (look at section 4.2.

Table 4.1: 2D detection evaluation

Category	**TP**	**FP**	**FN**	**Precision**	**Recall**
Laptop	2752	4	2	0.99854862	0.99927378
Mug	2783	946	56	0.74631268	0.98027474
Bowl	2061	0	309	1.0	0.86962025

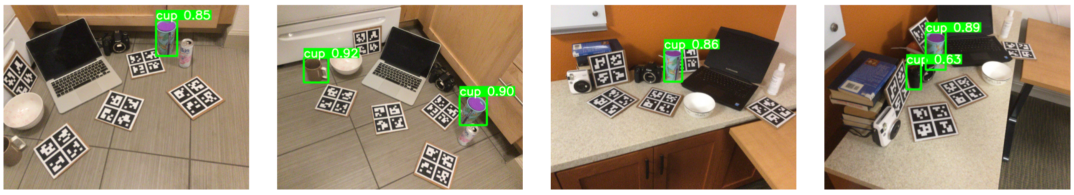
As it can be read from the table 4.1, the category with the highest precision is a bowl category which outputted zero false positive predictions. The second best is laptop category with four false positive detetections, these detections can be seen in figure 4.11. The false negatives detections which appeared in the laptop and bowl categories were caused by too low confidence level which was lower than 0.6 (most of the time the confidence level of these detections was around 0.5).



(a) Scene 1, Frame 0355 **(b)** Scene 5, Frame 0100 **(c)** Scene 5, Frame 0101 **(d)** Scene 5, Frame 0116

Figure 4.11: Images from 'laptop' category with false positive and true positive predictions.

In the 'mug' category most of the false positive detections come from scene number 3 (444 FP) and number 5 (407 FP) and most of the these detections happened when the 2D detector recognized a *'bottle_shengjun'* as a potential mug match. Example detections can be seen in figure 4.12.



(a) Scene 3, Frame 0215 **(b)** Scene 3, Frame 0399 **(c)** Scene 5, Frame 0081 **(d)** Scene 5, Frame 0373

Figure 4.12: Images from 'mug' category with false positive predictions.

Summary

In overall it can be stated that using pre-trained YOLOv5 weights is good enough for detecting the laptop, bowl and mug categories. The outputted recall metric is high for all of the categories, being the lowest for the bowl category where it is equal to 0.86. Moreover, the outputted precision is high for both laptop and bowl category, but it is significantly lower for the mug category where the metric is equal to 0.75.

This is due to the 2D detector recognizing the *'bottle_shengjun'* model as a cup, which according to the ground truth data should be labelled as a bottle. This issue occurred in both test scene number 3 and 5, which are the only scenes in the test set consisting of this model. In order to check if the 2D detector fails to detect other 'mug' objects, the precision and recall metrics has been calculated again, this time without using scenes number 3 and 5. As the result, the evaluation returned the precision equaled to 0.96 and the recall equal to 0.98, which shows that the 2D detector can handle the 'mug' detection as a 'cup' recognition well enough as long as the *'bottle_shengjun'* is not present in the scene.

4.4 Evaluation

Since there is no evaluation code provided in the FS-Net GitHub repository1, it was necessary to find a way to evaluate the results. For that purpose, the evaluation code from Shape Prior Deformation Network [2] has been modified and used [32].

The evaluation code provided in just mentioned network repository detects and evaluates all of the images and all of the objects and categories at once, meaning that there was no code provided to evaluate just one category/object at a time. An example produced by this detection and evaluation code using model provided

[2]Link to the Shape Prior Deformation Network Github repository: `https://github.com/mentian/object-deformnet`

by the authors of the Shape Prior Deformation Network [32] can be seen in figure 4.13.

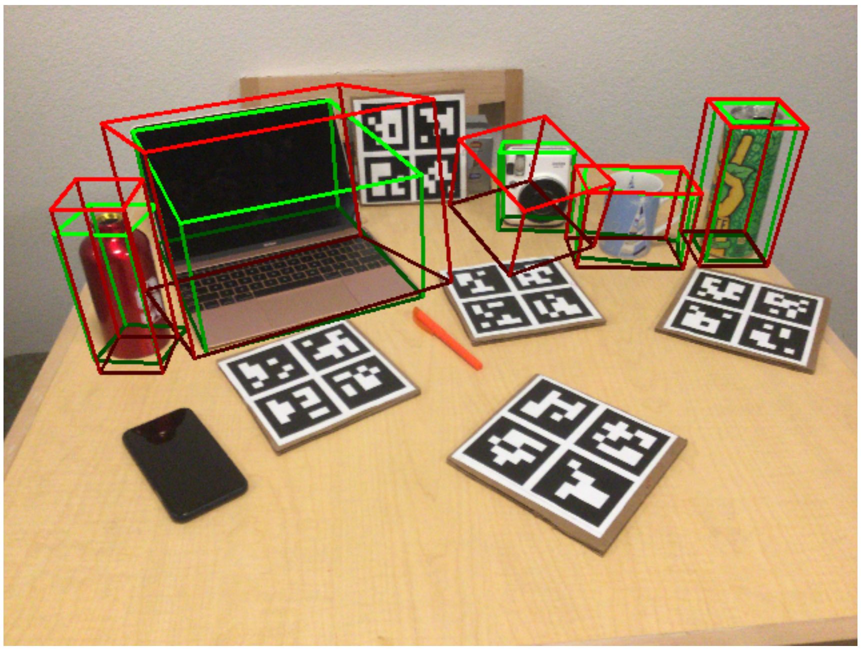


Figure 4.13: Predicted bounding boxes (red) and ground truth bounding boxes (green) plotted on frame number 0019 from test scene 2.

In the code provided in the Shape Prior Deformation Network repository 2 the results are evaluated by reading ground truth and predicted data from pickle files generated during the detection step, and then by comparing them to each other. Therefore, it was decided to apply the same logic when evaluating FS-Net[8] predictions. Each frame had its own pickle file in which following data could be found: a predicted class ID, 4x4 transformation matrix, 3x1 size (scale) matrix and handle visibility array consisting of boolean variables. Then the created pickles files were used for evaluating and visualizing the results.

The metrics being evaluated are:

- Mean Average Precision (mAP): 3D IoU_{25}, 3D IoU_{50}, 3D IoU_{75}, 5°2cm, 5°5cm, 10 °2cm, 10°5cm, 10°10cm
- Accuracy (Acc): 3D IoU_{25}, 3D IoU_{50}, 3D IoU_{75},, 5°2cm, 5°5cm, 10°2cm, 10°5cm, 10°10cm

mAP

Mean Average Precision **(mAP)** is a metric commonly used as a benchmark to measure the accuracy of the algorithm. This metric can be also explained as an average of each category of average precision (AP), which is calculated using precision-recall curve. [36]

The formula for calculating precision (**P**) and recall (**R**) was already introduced in section 4.3.2 in formula 4.3.2.

Once the precision and recall are found, the average precision is calculated using following formula:

$$AP = \sum_{n} (R_n - R_{n-1}) P_n \tag{4.4}$$

where :

- n is the rank order
- R_n is the recall at rank m
- P_n is the precision at rank m [36]

Finally, mean average precision can be found:

$$mAP = \frac{1}{N} \sum_{n}^{N} AP_n \tag{4.5}$$

where N is the total number of classes. [36]
As mentioned in 4.3.1, at this point of the project it is not possible to evaluate all of the categories, and the focus is put on one category at a time. Therefore, the variable N is set to 1 throughout the project.

IoU

IoU_X is a Intersection-over-Union metric for 3D object detection under different overlap thresholds, where the overlap ratio larger than the threshold X will be accepted. It is one of the most common metrics used in both instance and category 6D pose estimation [9].

This metric has been originally used to evaluate the performance of the methods working in 2D space, however, it can be also used for evaluation of 3D detection. The only difference between these two evaluation is that the first one operates on 2D bounding boxes, and the other on 3D bounding boxes. This metric is calculated by comparing the area of intersection to the area of union and the formula is given as follows:

$$\omega_{IoU} = \frac{B \cap \bar{B}}{B \cup \bar{B}} \tag{4.6}$$

where :

- ω_{IoU} is the overlapping ratio

- B is the estimated bounding box
- $\bar{B}$ is the ground truth bounding box [29]

Error of rotation and translation

n°m cm represents pose estimation error of rotation and translation. The rotation error less than n°and the translation error less than m cm is accepted [9]. For instance, the estimated 6D pose under the 5°5cm metric would be considered correct if its rotation error is within 5°and the translation error is below 5cm [21].

In the code, this metric is evaluated by comparing the predicted homogeneous affine transformation matrix to the ground truth one. This generates the rotation error (angle difference in degree) and translation error (L2 difference in centimeter), which are then used for the evaluation.

The translation error is calculated by finding a norm of an array generated by subtracting the ground truth translation array from the predicted one.

The rotation error is calculated differently for symmetrical and non-symmetrical objects. For the symmetrical ones, only the second column of the rotation matrix is being used and the formula looks as follows:

$$Theta_{symm} = \arccos\left(\frac{y_{pred} \cdot y_{gt}}{\|y_{pred}\| \, \|y_{gt}\|}\right) * \frac{180}{\pi} \tag{4.7}$$

where :

- $Theta_{symm}$ is the rotation error angle in degrees for symmetrical objects
- y_{pred} is the middle column of the predicted rotation matrix
- y_{gt} is the middle column of the ground truth rotation matrix

$$Theta_{nonsymm} = \arccos\left(\frac{Tr(R_{pred}R_{gt}^{T}) - 1}{2}\right) * \frac{180}{\pi} \tag{4.8}$$

where :

- $Theta_{nonsymm}$ is the rotation error angle in degrees for non-symmetrical objects
- R_{pred} is the predicted rotation matrix
- R_{gt} is the the ground truth rotation matrix

Chapter 5

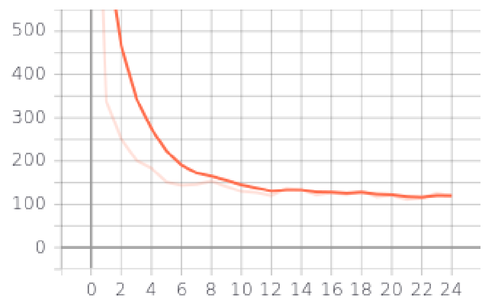
Model trainings

All of the following trainings were analyzed using produced TensorBoard graphs[1]. The evaluation of the 6D Pose Estimation was run only on True Positives samples outputted by the previous test described in 4.3.2 to separate the object detection and pose estimation task. As the validation set used for evaluating the model after each epoch (see section 4.2), a test scene number 1 has been used. Any changes applied to the network will be mentioned in each of the sections.

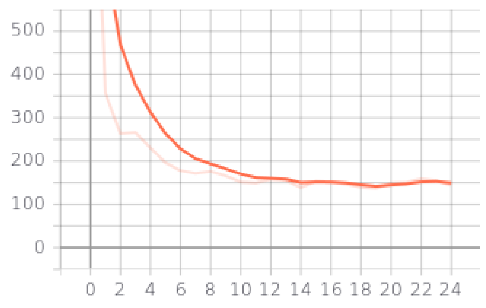
5.1 First Training

During the detection step it was noticed that although the 6D pose of the laptop is estimated properly, the estimation of the bowl and of the bottle objects is not being handled well. Since the source of the problem was not known, it was decided to re-train the laptop and bowl models but this time by plotting the metrics graphs in order to analyze them afterwards (the first trainings which generated the initial models were run using non-modified training code from FS-Net[8] repository, where no validation set was used and where no graphs were plotted).

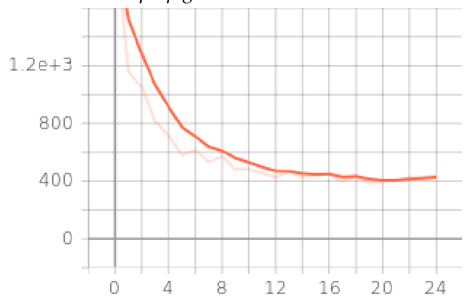
Since wrong estimation of the rotation seemed to be an issue, this section will focus on comparing the graphs which plotted rotation metrics.



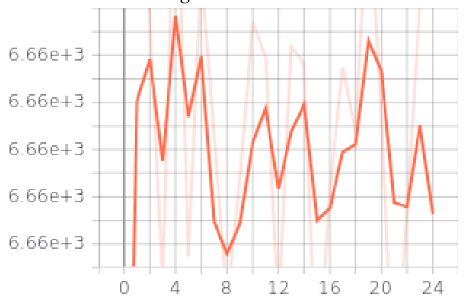
(a) *Laptop green rotation loss*



(b) *Bowl green rotation loss*



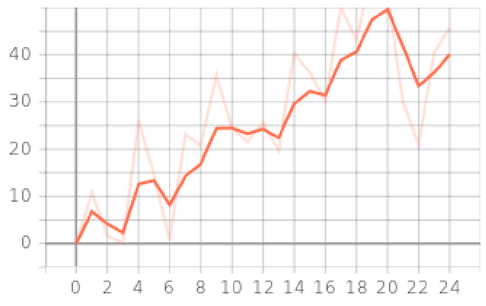
(c) *Laptop red rotation loss*



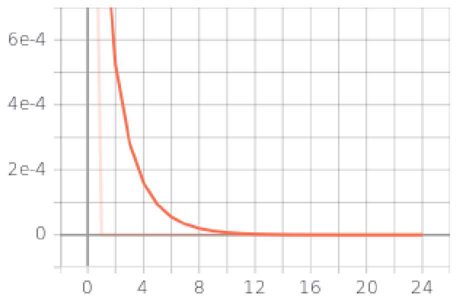
(d) *Bowl red rotation loss*

Figure 5.1: Green and red rotation loss graphs of laptop and bowl

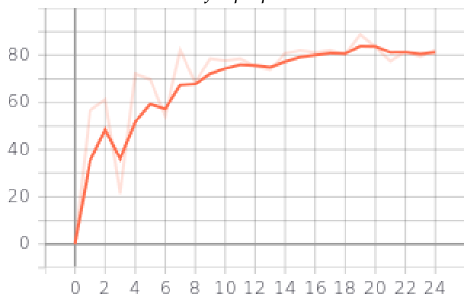
As it can be seen in figure 5.1, the laptop model seems to be learning successfully how to handle both red and green rotation. The bowl model also seems to be handling the green rotation well, but as it can be seen in the red loss graph, the model is not converging. This however can be explained and accepted, as models trained on symmetrical object, like a bowl, do not take the red rotation into consideration when estimating the 6D pose, and base the rotation estimation only on the predicted green rotation (for visualization, look at figure 2.7).



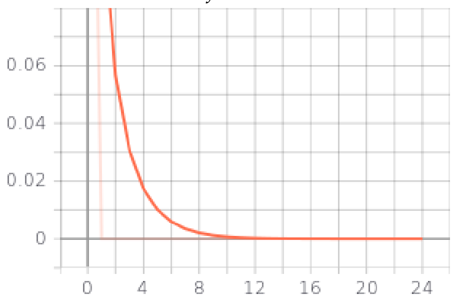
(a) *5°5cm of laptop model*



(b) *5°5cm of bowl model*



(c) *10°10cm of laptop model*



(d) *10°10cm of bowl model*

Figure 5.2: 5°5cm and 10°10cm metrics evaluated after each epoch of training bowl and laptop models.

In the figure 5.2 the comparison of two rotation and translation metrics can be seen. The graphs are not expected to be smooth, but it should be expected for the metric to increase over epochs. As it can be read from the graphs, the laptop model is learning over time and is able to evaluate the test set better with every epoch, however the bowl model is not able to predict the rotation and translation properly, which results in the metrics being equal to zero most of the time.

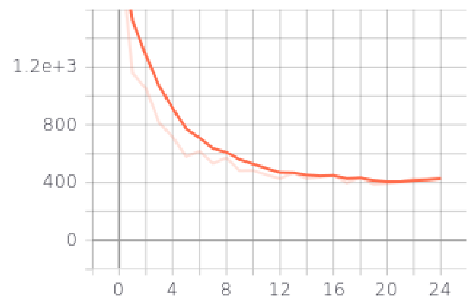
Analysis of the above graphs brings a conclusion that even though the bowl model learns how to predict the green rotation properly (based on the plotted loss function), it can not apply it properly while estimating the 6D pose of objects in the test scene, as it results in the rotation metrics being equal to zero. This suggests that there is no actual problem with the training and model itself, but with the detection part of the solution.

One of the possible reasons why the bowl model fails to estimate the 6D pose, while the laptop model does it properly using the same detection and evaluation code, was the fact that the code uses two classifiers to estimate the rotation around the x and y axis. This could turn out to be problem for symmetric objects like a bowl because the red rotation model generated for this object do not learn the pose properly (as it can be seen in figure 5.1d). Therefore, the detection code was studied more in detail in order to see how symmetric objects are being handled. After the analysis and testing different changes applied to the code, it turned out that in order to estimate the pose of symmetrical objects only the green rotation classifier needs to be used. Therefore, the code was adjusted to evaluate both symmetrical and non-symmetrical objects properly depending on the name of the category.

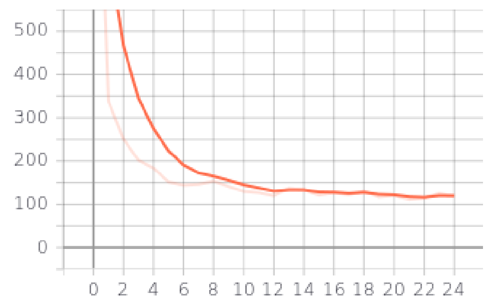
5.2 Second Training

The training was ran on three categories where each of them is supposed to represent different symmetry difficulties being: Laptop (non-symmetrical), Mug (semi-symmetrical, as the handle cannot be seen from every angle) and Bowl (symmetrical). Laptop model was already trained and evaluated properly as explained in the section 5.1, but the bowl needed to be retrained this time using proper evaluation code for the validation set. The mug model was retrained as well, as there was no graphs plotted in the initial training and the model was not evaluated on the validation set after every epoch.

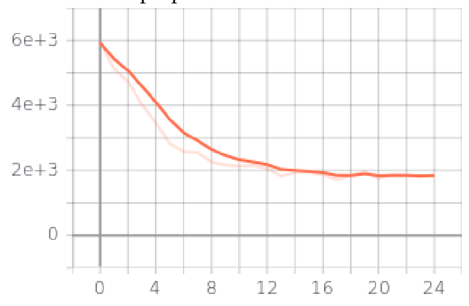
On the below figure, the comparison of red and green rotation losses of laptop, mug and a bowl is shown (figure 5.3)



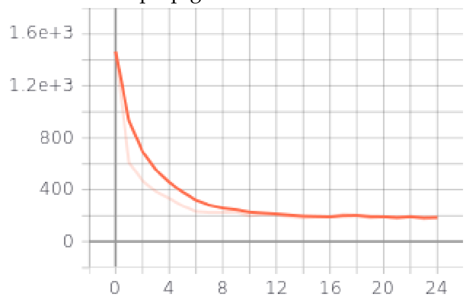
(a) Laptop red rotation loss



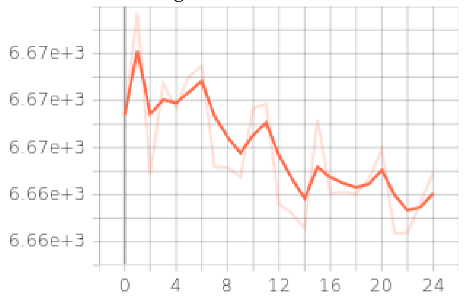
(b) Laptop green rotation loss



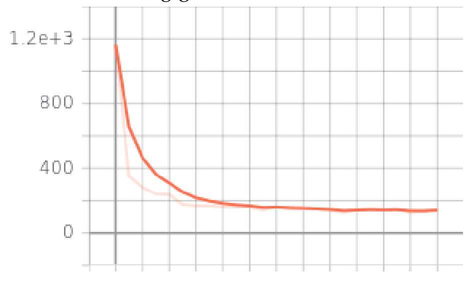
(c) Mug red rotation loss



(d) Mug green rotation loss



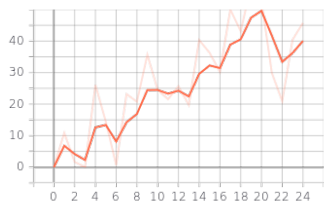
(e) Bowl red rotation loss



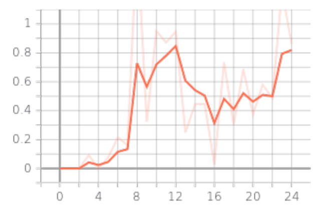
(f) Bowl green rotation loss

Figure 5.3: Green and red rotation loss for each of the categories where first row is laptop, second is a mug, and the third one is a bowl.

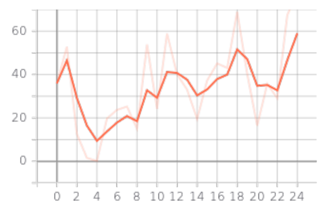
It can be read from the above graphs, all of the models seem to be learning the green rotation well. As it was already mentioned in section 5.1, the red rotation loss for the bowl model converges due to this rotation not being important and used by symmetric objects. For both laptop and mug categories the model seems to be learning how to estimate the red rotation properly, however, the laptop model seems to be learning it faster and better, as the values generated by the laptop loss function are smaller than the one generated by the mug.



(a) *Laptop*

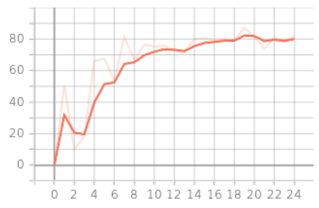


(b) *Mug*

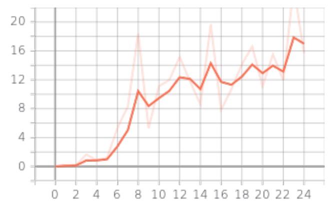


(c) *Bowl*

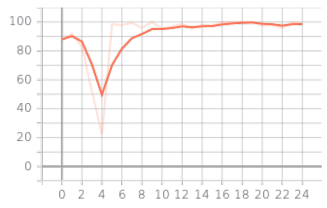
Figure 5.4: MaP 5°5cm metric.



(a) *Laptop*



(b) *Mug*



(c) *Bowl*

Figure 5.5: MaP 10°5cm metric.

In the above graphs (5.4 and 5.5), the evaluation of the mean average precision of 5°5cm and 10°5cm metrics can be seen. As it can be read, the mug category handles these metrics the worst among other categories. It takes longer time for this category to learn how to estimate the 6D pose properly when compared to laptop category. Moreover, not only it takes longer but both of the metrics evaluated on mug category are significantly lower throughout the whole training when compared to laptop and bowl categories.

Evaluation

After the models were trained, they were evaluated on the test set and the results can be seen in tables 5.1 and 5.2.

Table 5.1: MaP evaluation

Category	IoU_{25}	IoU_{50}	IoU_{75}	**5°2cm**	**5°5cm**	**10°2cm**	**10°5cm**	**10°10cm**
Laptop	99.1	87.9	23.5	3.1	22.6	5.5	50.9	71.4
Mug	100.0	91.6	39.4	0.2	0.2	3.7	5.2	5.2
Bowl	100.0	99.5	98.0	51.8	57.4	80.6	93.0	93.4

Table 5.2: Acc evaluation

Category	IoU_{25}	IoU_{50}	IoU_{75}	**5°2cm**	**5°5cm**	**10°2cm**	**10°5cm**	**10°10cm**
Laptop	99.3	91.8	46.5	15.6	40.8	21.8	69.0	80.3
Mug	100.0	95.7	62.7	4.1	4.7	19.0	22.6	22.6
Bowl	100.0	99.7	99.0	69.5	73.9	89.4	96.3	96.5

The results outputted by the model trained in this project can be compared to the ones provided in the Supplementary Material of FS-Net and they can be seen in table 5.3.[8]

Table 5.3: Results of object-wise evaluation provided by the authors of FS-Net [8]

Category	IoU_{75}	**5°5cm**	**10°5cm**	**10°10cm**
Laptop	38.05	16.59	55.7	68.59
Mug	75.34	8.74	36.98	37.06
Bowl	98.10	42.19	81.34	87.55

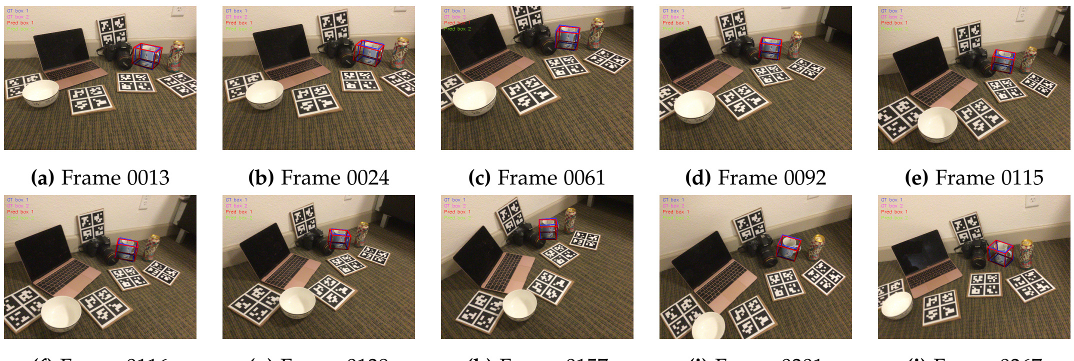
Unfortunately, the authors of the FS-Net[8] did not provide information if the results presented in the table 5.3 are based on Accuracy or MaP evaluation. However, it can be assumed that the results were given using Mean Average precision, since the results provided in figure 2.9 from Shape-Prior network were reported as MaP in their original paper [32]. Therefore, the results outputted by the newly trained model in this project will be compared to the one provided in the paper using Mean Average Precision.

The intersection over union metric at threshold equal to 75 was almost the same for the *bowl* model in both of the evaluations and it was reported to be higher in both the *laptop* and *mug* category. While the difference between the laptop evaluations is not that high, the value of this metric reported to a mug category is almost twice as high as the one outputted in this project. It is also worth noting, that the IoU at threshold values equal to 25 and 50 is high for all of the three categories, nevertheless, these values can not be compared to the results reported by the authors as they are not publicly provided.

The rotation and translation error has been reported as higher when using the model trained in this project in almost all of the metrics in both laptop and bowl categories (the only metric reported as lower is 10°5cm for the laptop category which was reported as 5% higher). This however does not apply to the mug category, where all of the rotation/translation metrics outputted by the model trained in this project are way lower than the one reported in the original paper (almost 7 times lower for 10°rotation error, and almost 47 times lower for the 5°rotation error).

Mug evaluation test

Since mug category achieved much lower results than it expected it was decided to do an additional test. This time the evaluation code was used to evaluate ten hand-picked predictions in order to see if the results would be also as low as when evaluated on all the scenes. The picked predictions were the ones that looked the most accurate in the generated images from the previous evaluation. The chosen predictions plotted along with their matching ground truth bounding boxes can be seen in figure 5.6.



(a) Frame 0013 **(b)** Frame 0024 **(c)** Frame 0061 **(d)** Frame 0092 **(e)** Frame 0115

(f) Frame 0116 **(g)** Frame 0138 **(h)** Frame 0157 **(i)** Frame 0301 **(j)** Frame 0367

Figure 5.6: Hand picked detections used for the mug evaluation test. All of the images come from first test scene. Blue box - ground truth, red box - prediction.

It needs to be noted, that this test is not supposed to show how well the model predicts the results, since only the best visually predictions are being evaluated. Instead, running this test is supposed to help with seeing where the source of the problem is and if it lies in the evaluation code.

The evaluation run on these ten images outputted following results:

Table 5.4: MaP evaluation of 10 mug predictions

Category	IoU_{25}	IoU_{50}	IoU_{75}	**5°2cm**	**5°5cm**	**10°2cm**	**10°5cm**	**10°10cm**
Mug	100.0	100.0	82.0	10.0	10.0	36.7	82.0	82.0

As it can be seen in table 5.4, the results are much higher than in the table 5.1, which is as it should be expected to be. This time the evaluation code was run on the best found predictions and the outputted metrics achieved high scores in both IoU and rotation/translation errors. Therefore, it can be stated that the evaluation code evaluated the predictions presented in table 5.1 properly, and the problem with estimating mug 6D pose starts somewhere else.

5.3 Third Training

The third training focused only on the ′mug′ category, as the evaluation of the ′bowl′ and ′laptop′ models outputted satisfactory results. The purpose of this training was to improve the ′mug′ model in such a way that the new model would be able to generate better results which would be closer to the one presented in the FS-Net[8] paper.

The reason for which the ′mug′ model trained in the second training generated not good enough results (section 5.2) is still not found, but it is now known that the evaluation code is not the source of the problem (for more detailed explanation look at section 5.2). Therefore, it was decided to retrain the model this time on more epochs too see if that would improve the model to any extent. Another reason for taking that decision is the fact that the authors of the FS-Net[8] network trained all of their models on 50 epochs. Moreover, the batch size was changed to 6, as CUDA[25] could not allocate enough memory for any higher batch size.

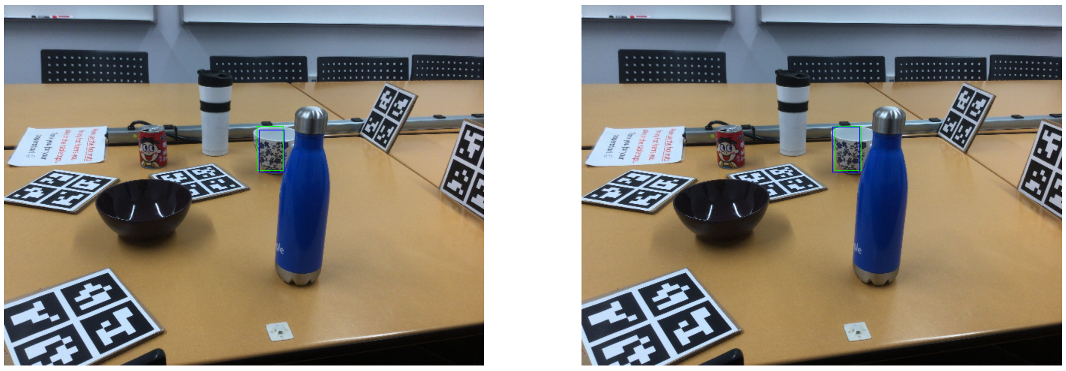
During training the model the program would run into an issue where it would try to generate a random integer within the range set from zero to the number of times a ′1′ is appearing in the ′labs′ array. At this point of training, the solution would throw an error that it cannot generate a random number in range between 0 and 0, meaning that no ones have been found in the ′labs′ array.

After further investigation it turned out that the issue is coming from a 2D bounding box augmentation technique where a function would generate new 2D bounding boxes from the ground truth data by applying translation, shear, scale and rotation. Then, the points in the newly generated bounding box would be compared to the ones in the ground truth bounding box. If there will be too few points intersecting, the solution would set the ′labs′ array as full of zeros, which would then raise the above mentioned error. This issue was hard to figure out as it was happening during different moments of training as well as on different images, and furthermore, it was happening only while training the mug category.

As mentioned above, the function used for 2D box augmentation would apply rotation, translation, scale and shear to the ground truth bounding box. The values of these geometrical transformations would be randomly chosen from a predefined range which would be the same for all of the trained models and categories. After debugging the code, it started to be visible that the mentioned error would occur when the randomly chosen values would be so high or so low, that the function would generate a bounding box which would not overlap with the ground truth box to a sufficient degree.

Changing the ranges of the geometrical factors to smaller was not an adequate solution for a few reasons. First of all, setting a range to smaller would generate less diverse bounding boxes, which could make the model worse at generalizing. Secondly, the ranges used for generating the random numbers were the same in all of the categories in all of the trainings, which meant that there is not necessarily a problem with the ranges themselves.

Therefore, it was decided to add a recursive call of the function that would be called if the generated random numbers created a bounding box that did not overlap with the ground truth bounding box to a sufficient degree. That way the random geometrical factors were generated again and the process would repeat until a satisfactory bounding box is created. An example of the how the generated bounding boxes look like can be seen in figure



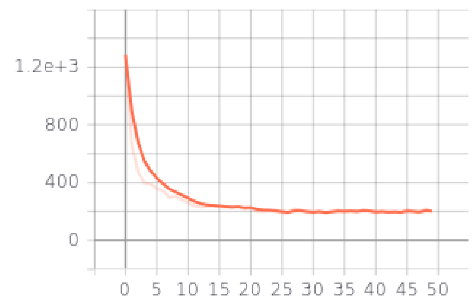
(a) 1st call of the function

(b) 2nd call of the function (1st recursive call)

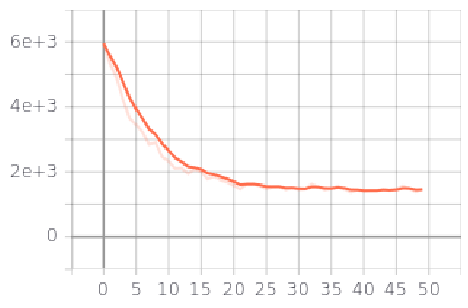
Figure 5.7: Scene 2 frame 0430. The green bounding box is the ground truth, and the blue one is the augmented bounding box. The (a) image shows the plotted bounding boxes when the function was called the first time, and (b) shows the ground truth box plotted with newly generated augmented bounding box, which this time was overlapping to a sufficient degree. As it can be seen, the problem might not lie in the poorly generated geometrical factors, but instead in not so precise, small, ground truth bounding boxes.

This method could slow down the training, however, it happened only a few times that the function needed to be called again. Although this solution might have many cons, it prevented the error from occurring and the network could continue to train without interruption.

Similarly as in the previous trainings, different loss functions and metrics have been plotted in graphs. In the figure 5.8, the red and green rotation loss can be seen.



(a) Mug green rotation loss

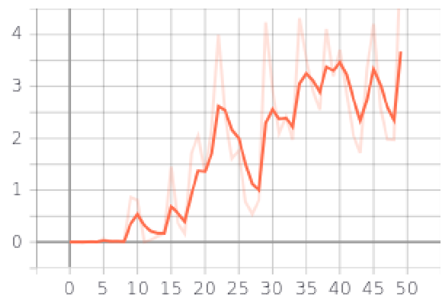


(b) Mug red rotation loss

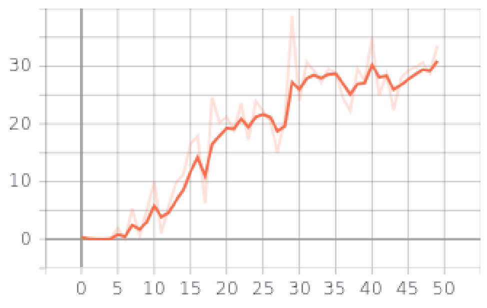
Figure 5.8: Mug rotation loss trained on 50 epochs.

As it can be read from 5.8, the values of the loss function stop decreasing significantly around 15 epochs for the green rotation and at around 25 of the red rotation. This suggests that training the model for longer than 25 epochs should not have big influence on the outcome.

Next the focus will be put on the graphs generated for the MaP 5°5cm and 10°5cm metrics, which can be seen in figure 5.9. As it can be read from these graphs, the results are starting to be more stable and better at around 30/35 epochs. Furthermore, the results seen on the below graphs can be compared to the one show in figure 5.4b and 5.5b, where it can be seen that the results generated by the model trained on 50 epochs are much higher than when trained on 25 epochs. This however was most likely due to improvement in other parts of network, as the rotation loss functions does not improve significantly over more than 30 epochs. Nevertheless, it can be stated that the model has improved and can estimate the rotation and translation of a mug more precisely.



(a) MaP 5°5cm



(b) MaP 10°5cm

Figure 5.9: Map rotation translation evaluation done using generated mug models.

Evaluation

After the new mug model was done training, it was used for evaluation on all of the scenes. The results generated by the model can be seen in table 5.5.

Table 5.5: MaP evaluation of mug predictions (50 epochs)

Category	IoU_{25}	IoU_{50}	IoU_{75}	**5°2cm**	**5°5cm**	**10°2cm**	**10°5cm**	**10°10cm**
Mug	100.0	93.3	58.5	0.5	0.7	9.6	15.4	15.6

Table 5.6: MaP evaluation of mug predictions (25 epochs)

Category	IoU_{25}	IoU_{50}	IoU_{75}	**5°2cm**	**5°5cm**	**10°2cm**	**10°5cm**	**10°10cm**
Mug	100.0	91.6	39.4	0.2	0.2	3.7	5.2	5.2

Although the generated results are not close to the one provided by the authors of FS-Net[8], they are still 3 times higher than the one generated by the model trained on 25 epochs (table 5.6) when comparing rotation and translation error metric, which shows significant improvement in training the model. Moreover, there is also a big improvement in the IoU_{75} metric which before was equal to 39.4 and now equals to 58.5.

Chapter 6

Discussion

In this chapter the solution and the course of the project is discussed.

The first part of the project was to reproduce the results achieved by the authors of the FS-Net[8] network by training and evaluating the models by the thesis's author. This task turned out to be more difficult and time-consuming than it was expected at the start of the project. The GitHub repository shared by the authors of the mentioned network consists of non-specific instructions of how to generate the data and train the model properly, which made the work on the project less efficient and longer.

Moreover, the repository is inactive, meaning that the authors do not answer the issue section and do not update the code if mistakes have been found. This of course, is not in authors' responsibility to keep the repository active, as it is already very appreciated that they decided to share the code publicly. However, it would be found helpful if the authors were actively taking part of the discussions in the issue section.

Furthermore, detection and evaluation steps turned out to be a big issue. The detection code provided in the GitHub repository was highly adjusted to work only on the 'laptop' category, and therefore the whole code needed to be modified in order for it to detect different categories. It was not possible to reuse a detection code from other 6D pose estimation network repositories, as the model generated by the FS-Net[8] consists of four separate models in pth format, while different networks were using one file for detecting and estimating the pose of all of the categories. The authors of the FS-Net [8] did not provide the evaluation code, nor did they mention from which repository the evaluation code should be used. Fortunately, these issues have been solved to a high extent, but with lot of time and effort put into it.

In this project, the focus was put on three categories: laptop, mug and bowl. It can be stated, that the results achieved by the authors of FS-Net[8] have been reproduced for a laptop and bowl category. This however does not apply to the mug category, which did not perform as well as it should be expected. Few trainings have been run in order to improve the 'mug' model, which seemed to output better results, but still not close to the one stated in the FS-Net paper. The exact reason for why only the 'mug' category is not able to estimate the rotation and translation properly has not been found, which can possibly mean that the source of the issue can be found in the data used for training.

Since most of the time has been spent on trying on reproducing the results, there was not much time left for trying to improve the trained models. This could have been potentially done by training on more data, for instance by adding images from the CAMERA dataset[34] which FS-Net[8] is currently not using, as it helped with improving the results in other category-level 6D pose estimation networks.

It is also worth mentioning that although the NOCS datasets[34] became a standard dataset used for category-level 6D pose estimation trainings, it is not a perfect choice. The objects in this dataset are placed in the very similar positions as well as similar scenarios as in all of the scenes. Although it is good enough for comparing the results generated by different networks, it is a not a dataset that would perform well for more real-life scenarios.

Chapter 7

Conclusion

The purpose of this master thesis was to research the category-level 6D object pose estimation by investigating the currently available state-of-the-art solutions. This was done by trying to reproduce the results provided in the papers, and then by attempting to improve them by applying changes to the code.

Reproducing the results turned out to be a great way of gaining knowledge about the 6D pose estimation field. At first, the research was done in order to understand the topic more which gave insights of what should be expected of the algorithm. Next, one of the networks was chosen as a start learning point, which was used for understanding how the 6D pose is being estimated once the model is produced.

Based on the conducted evaluation it can be stated that it is more difficult to estimate poses of objects of a 'mug' category than of the ones of the 'laptop' and 'bowl'. Although the intersection over union metric achieved relatively high results, estimating the rotation and translation still remains a challenge.

Moreover, it can be also said that estimating the 6D pose of symmetrical objects results in a higher $n^{\circ}m$ cm metric. This can be seen by comparing the results produced in this project, but it can also be confirmed by analyzing the data provided by the authors of FS-Net, where symmetrical objects achieved an average of 89% in $10^{\circ}10$cm metric, and non-symmetrical achieved an average of 40% . [8]

The category-level 6D object pose estimation is still an open problem with lot of space for improvement. The research conducted in this paper showed that it can be a challenging task, even when it comes to only reproducing the results. Furthermore, it showed that working with objects with more than one rotation axis can be difficult and therefore, there should be more focus put on estimating the 6D pose of non-symmetrical objects.

7.1 Future Development

In future work, it would be interesting to see if combing NOCS-Real dataset[34] with CAMERA dataset[34] would improve the models generated by the FS-Net[8]. This would add additional data to the dataset, which could possibly improve models' performance and potentially would estimate the 6D pose of objects more precisely. Moreover, a bigger variety of the objects on the category-level could be added to the dataset, for example different laptop models or taller mugs.

Another future plan is training the 2D detector such that it would be able to detect all of the categories. This would require annotating the NOCS-Real dataset[34] in YOLOv5[19] manners, which could be a time-consuming task that was not falling into the scope of this project. However, if the 2D detector would be able to detect all of the objects, a full end-to-end evaluation could be done on the test set.

Moreover, creating a completely new dataset for category-level 6D pose estimation would be also a potentially good future task. This could be a dataset with more variety of poses, for instance, with objects laying in different positions or facing the table down, with different scenarios, like a person holding an object or an object being dropped down, or with more challenging objects, like made out of glass or transparent material. That is also a case if the solution was used by Aivero[31], as the trained model needs to be adjusted to their needs.

Bibliography

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[2] Leonard Barolli, Peter Hellinckx, and Juggapong Natwichai. *Advances on P2P, Parallel, Grid, Cloud and Internet Computing Proceedings of the 14th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2019): Proceedings of the 14th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2019)*. Jan. 2020. ISBN: 978-3-030-33508-3. DOI: `10.1007/978-3-030-33509-0`.

[3] Blossom Bastian and C. Jiji. "Detection and pose estimation of auto-rickshaws from traffic images". In: *Machine Vision and Applications* 31 (July 2020). DOI: `10.1007/s00138-020-01106-0`.

[4] Leonard Bruns and Patric Jensfelt. *On the Evaluation of RGB-D-based Categorical Pose and Shape Estimation*. Feb. 2022.

[5] Catherine Capellen, Max Schwarz, and Sven Behnke. "ConvPoseCNN: Dense Convolutional 6D Object Pose Estimation". In: Jan. 2020, pp. 162–172. DOI: `10.5220/0008990901620172`.

[6] Angel X. Chang et al. *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep. arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[7] Dengsheng Chen et al. "Learning Canonical Shape Space for Category-Level 6D Object Pose and Size Estimation". In: June 2020, pp. 11970–11979. DOI: `10.1109/CVPR42600.2020.01199`.

[8] Wei Chen et al. "FS-Net: Fast Shape-based Network for Category-Level 6D Object Pose Estimation with Decoupled Rotation Mechanism". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 1581–1590.

[9] Wei Chen et al. *FS-Net: Fast Shape-based Network for Category-Level 6D Object Pose Estimation with Decoupled Rotation Mechanism*. 2021. DOI: 10.48550/ARXIV.2103.07054. URL: https://arxiv.org/abs/2103.07054.

[10] Wei Chen et al. "G2L-Net: Global to Local Network for Real-Time 6D Pose Estimation With Embedding Vector Features". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 4232–4241.

[11] Xu Chen et al. "Category Level Object Pose Estimation via Neural Analysis-by-Synthesis". In: *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVI*. Glasgow, United Kingdom: Springer-Verlag, 2020, 139–156. ISBN: 978-3-030-58573-0. DOI: 10.1007/978-3-030-58574-7_9. URL: https://doi.org/10.1007/978-3-030-58574-7_9.

[12] *Computer science conference(journal) rank*. URL: https://www.aminer.org/ranks/conf.

[13] *Conference ranking*. URL: https://www.resurchify.com/cr/find/?query=computer%2Bvision.

[14] Thanh-Toan Do et al. "Deep-6DPose: Recovering 6D Object Pose from a Single RGB Image". In: (Feb. 2018).

[15] Clemens Eppner et al. "Lessons from the Amazon Picking Challenge: Four Aspects of Building Robotic Systems". In: June 2016. DOI: 10.15607/RSS.2016.XII.036.

[16] github. *GitHub*. 2020. URL: https://github.com/.

[17] Kaiming He et al. "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[18] Stefan Hinterstoißer et al. "Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes". In: *ACCV*. 2012.

[19] Glenn Jocher et al. *ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements*. Version v3.1. Oct. 2020. DOI: 10.5281/zenodo.4154370. URL: https://doi.org/10.5281/zenodo.4154370.

[20] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2015).

[21] Yi Li et al. "DeepIM: Deep Iterative Matching for 6D Pose Estimation: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part VI". In: (Sept. 2018), pp. 695–711. DOI: 10.1007/978-3-030-01231-1_42.

[22] Jiehong Lin et al. "DualPoseNet: Category-level 6D Object Pose and Size Estimation Using Dual Pose Network with Refined Learning of Pose Consistency". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), pp. 3540–3549.

[23] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *ECCV*. 2014.

[24] Zhi-Hao Lin, Sheng Yu Huang, and Y. Wang. "Convolution in the Cloud: Learning Deformable Kernels in 3D Graph Convolution Networks for Point Cloud Analysis". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 1797–1806.

[25] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. *CUDA, release: 10.2.89*. 2020. URL: https://developer.nvidia.com/cuda-toolkit.

[26] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: (Apr. 2018).

[27] Charles Ruizhongtai Qi et al. "Frustum PointNets for 3D Object Detection from RGB-D Data". In: (Nov. 2017).

[28] Charles Ruizhongtai Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: (Dec. 2016).

[29] Caner Sahin et al. "Instance- and Category-Level 6D Object Pose Estimation". In: *RGB-D Image Analysis and Processing*. Cham: Springer International Publishing, 2019, pp. 243–265. DOI: 10.1007/978-3-030-28603-3_11. URL: https://doi.org/10.1007/978-3-030-28603-3_11.

[30] Boris Schauerte. *Conference ranks*. URL: http://www.conferenceranks.com/#data.

[31] *Smooth AI services powered by 3D vision*. 2022. URL: https://www.aivero.com/.

[32] Meng Tian, Jr Ang, and Gim Lee. "Shape Prior Deformation for Categorical 6D Object Pose and Size Estimation". In: (July 2020).

[33] Chen Wang et al. "6-PACK: Category-level 6D Pose Tracker with Anchor-Based Keypoints". In: (Oct. 2019).

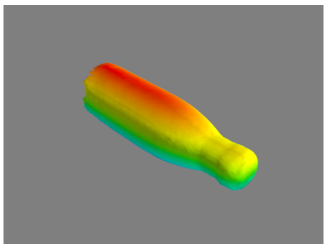
[34] He Wang et al. "Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

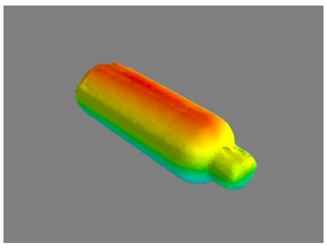
[35] Qingnan Zhou. *PyMesh/PyMesh: Geometry Processing Library for python*. URL: https://github.com/PyMesh/PyMesh.

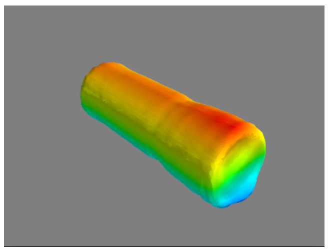
[36] Zehao Zhu et al. “A Large-Scale VR Panoramic Dataset of QR Code and Improved Detecting Algorithm”. In: *VR/AR and 3D Displays*. Ed. by Weitao Song and Feng Xu. Singapore: Springer Singapore, 2021, pp. 137–148. ISBN: 978-981-33-6549-0.

[37] Lu Zou et al. “6D-ViT: Category-Level 6D Object Pose Estimation via Transformer-based Instance Representation Learning”. In: *ArXiv* abs/2110.04792 (2021).

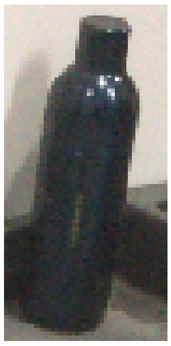
.1 Models in NOCS train set


(a) *"bottle blue google"* model


(b) *"bottle starbuck"* model


(c) *"bottle3 scene5"* model

(d) *"bottle blue google"* color image


(e) *"bottle starbuck"* color image

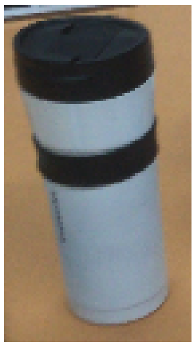
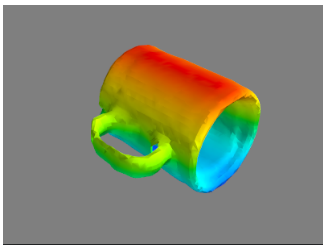
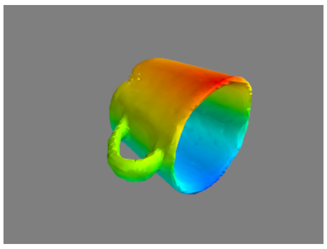
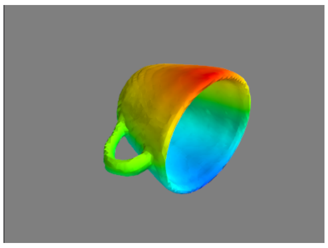

(f) *"bottle3 scene5"* color image

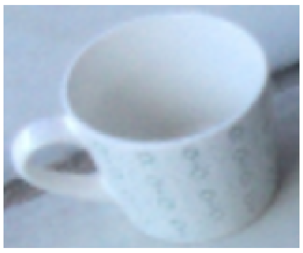
Figure 1: Training bottle models and sample color pictures of them.


(a) *"mug vignesh"* model


(b) *"mug white green"* model


(c) *"mug2 scene3"* model

(d) *"mug vignesh"* color image


(e) *"mug white green"* color image

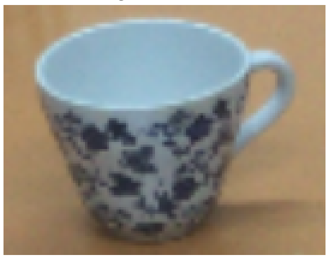
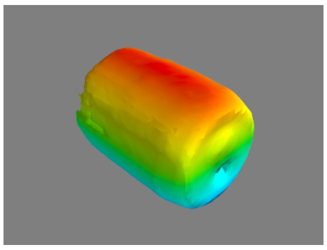
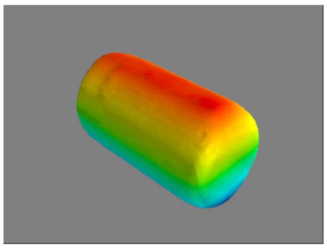
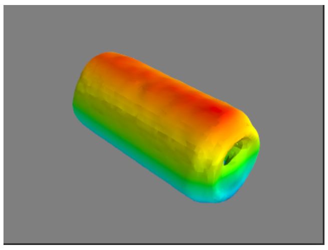

(f) *"mug2 scene3"* color image

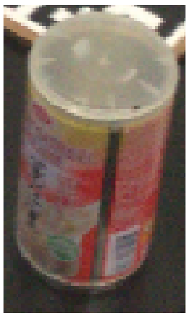
Figure 2: Training mug models and sample color pictures of them.


(a) *"can milk wangwang"* model


(b) *"can porridge"* model


(c) *"can tall yellow"* model

(d) *"can milk wangwang"* color


(e) *"can porridge"* color image

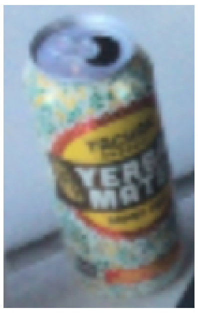
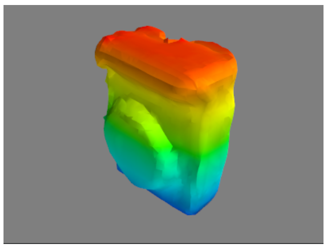
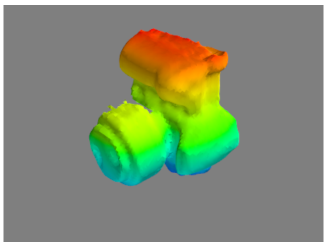
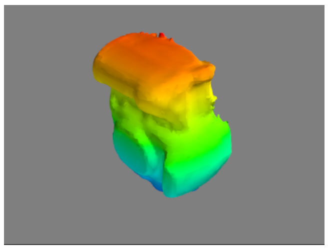

(f) *"can tall yellow"* color image

Figure 3: Training can models and sample color pictures of them.


(a) *"camera anastasia"* model


(b) *"camera dslr len"* model

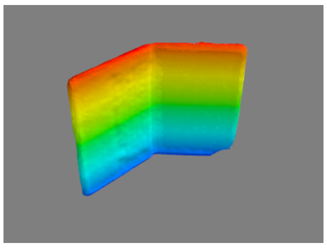

(c) *"camera dslr wo len* model

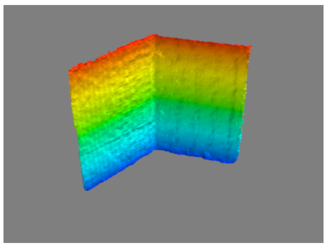
(d) *"camera anastasia"* color image

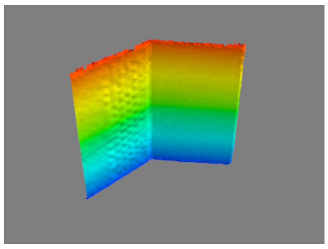
(e) *"camera dslr len"* color image

(f) *"camera dslr wo len* color image

Figure 4: Training camera models and sample color pictures of them.


(a) *"laptop air 0"* model


(b) *"laptop air 1"* model

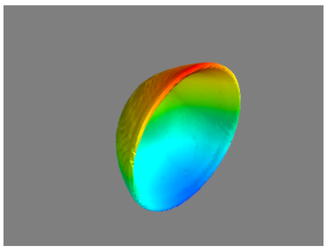

(c) *"laptop dell"* model

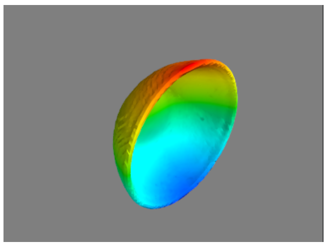
(d) *"laptop air 0"* color image

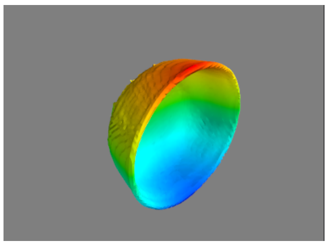
(e) *"laptop air 1"* color image

(f) *"laptop dell"* color image

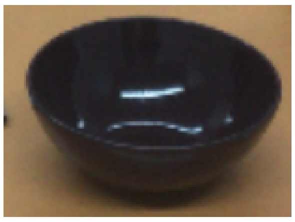
Figure 5: Training laptop models and sample color pictures of them.


(a) *"bowl blue ikea"* model


(b) *"bowl brown ikea"* model


(c) *"bowl chinese blue"* model

(d) *"bowl blue ikea"* color image


(e) *"bowl brown ikea"* color image

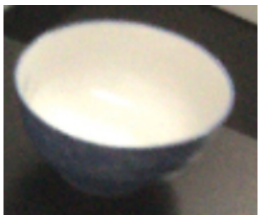
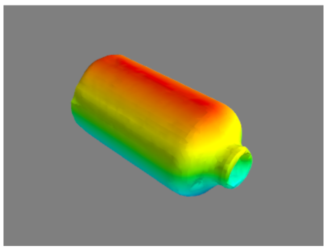
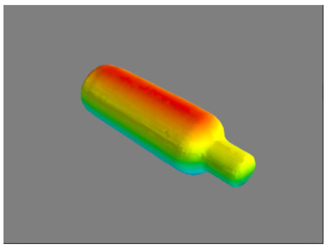
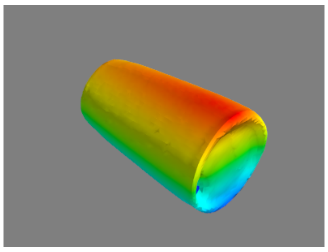

(f) *"bowl chinese"* color image

Figure 6: Training bowl models and sample color pictures of them.

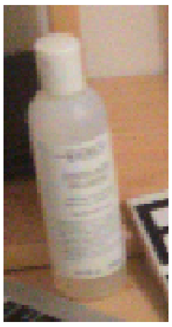
.2 Models in NOCS test set


(a) *"bottle red stanford"* model


(b) *"bottle shampoo"* model


(c) *"bottle shengjun"* model

(d) *"bottle red stanford"* color image


(e) *"bottle shampoo"* color image

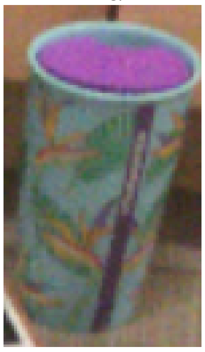
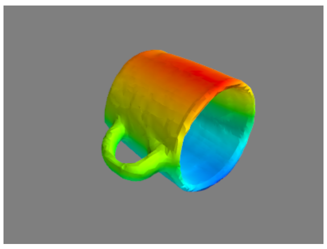
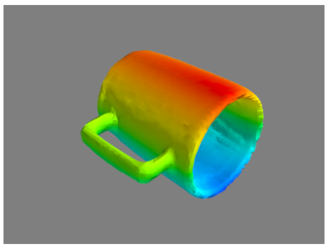
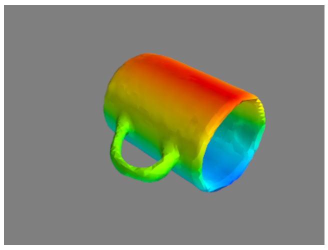

(f) *"bottle shengjun"* color image

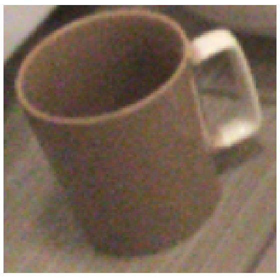
Figure 7: Test bottle models and sample color pictures of them.


(a) *"mug anastasia"* model


(b) *"mug brown starbucks"* model


(c) *"mug daniel"* model

(d) *"mug anastasia"* color image


(e) *"mug brown starbucks"* color image

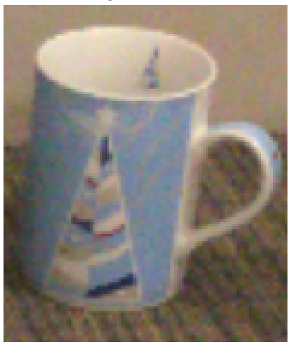
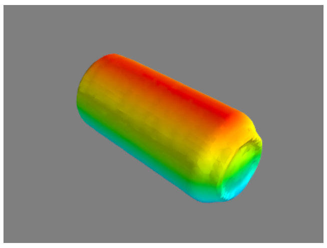
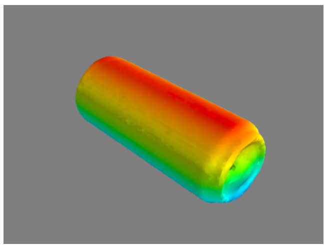
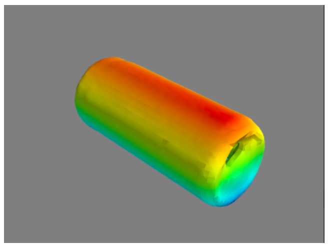
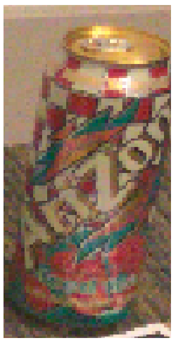

(f) *"mug daniel"* color image

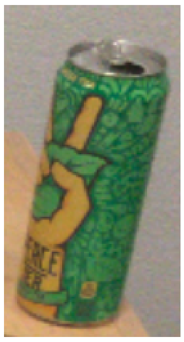
Figure 8: Test mug models and sample color pictures of them.


(a) *"can arizona tea"* model


(b) *"can green"* model


(c) *"can lotte milk"* model


(d) *"can arizona tea"* color


(e) *"can green"* color image

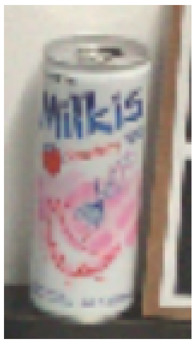
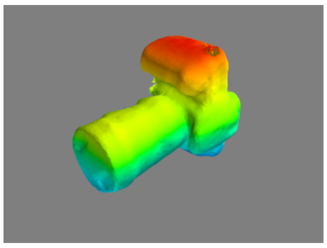
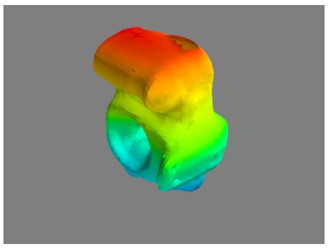
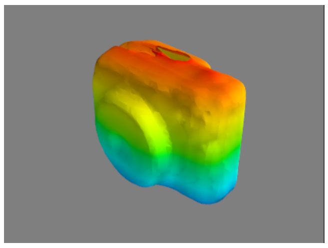

(f) *"can lotte milk"* color image

Figure 9: Test can models and sample color pictures of them.


(a) *"camera canon len"* model


(b) *"camera canon wo len"* model

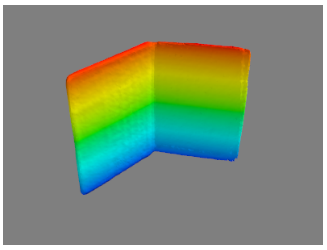

(c) *"camera shengjun* model

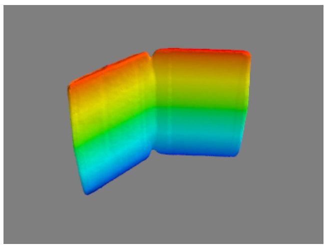
(d) *"camera canon len"* color image

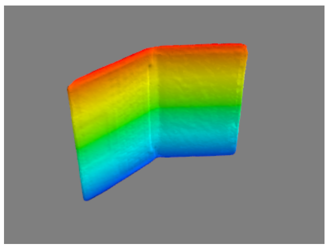
(e) *"camera canon wo len"* color

(f) *"camera shengjun"* color image

Figure 10: Test camera models and sample color pictures of them.


(a) *"laptop air bin"* model


(b) *"laptop alienware"* model


(c) *"laptop mac pro"* model

(d) *"laptop air xin"* color image

(e) *"laptop aliernware"* color image

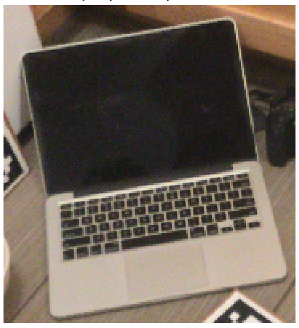
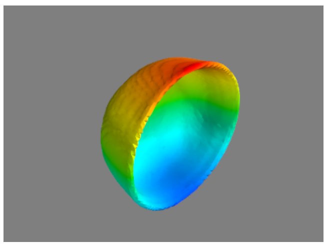
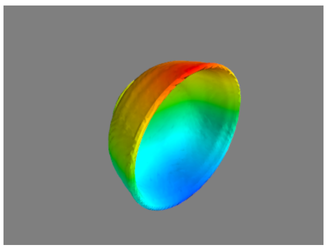
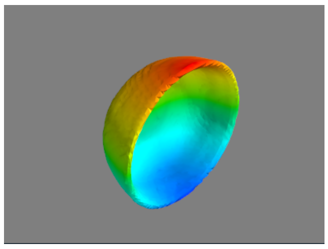

(f) *"laptop mac pro"* color image

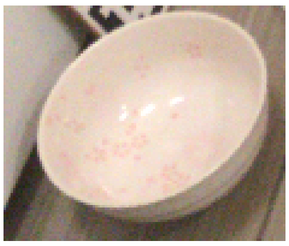
Figure 11: Test laptop models and sample color pictures of them.


(a) *"bowl blue white chinese"* model


(b) *"bowl shengjun"* model


(c) *"bowl white small"* model

(d) *"bowl blue white chinese"* color image


(e) *"bowl shengjun"* color image

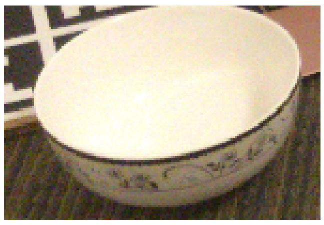

(f) *"bowl white small"* color image

Figure 12: Test bowl models and sample color pictures of them.