INTRODUCTION

The project described in this documentation consisted of developing an Instance Segmentation software using the image captured by an Intel RealSense D435 camera. It was realised in the summer semester of the 2022/23 academic year as part of the subject "Integrated decisive systems" (pl. *Zintegrowane systemy decyzyjne*).
The project team consisted of:

- Jakub Pilarski - leader
- Weronika Kapusta - vice-leader
- Andrzejewski Miłosz
- Iskrzycki Paweł
- Kołoszko Mateusz
- Koncewicz Gabriela
- Szymanek Tomasz
- Urbański Filip

The task required us to devise a method of Instance Segmentation. Afterwards, the shape of the object had to be determined based on the voxels belonging to it (by using e.g. a 3D bounding box). It was also necessary to devise our own dataset for model training.
Our initial project assumptions were as following:

- real-time work using RGBD RealSense D435 camera;
- static objects detection;
- object size limited from 0.1[m]x0.01[m]x0.01[m] to 1[m]x1[m]x1[m]
- final accuracy
  - segmentation accuracy > 80%
  - IoU > 60%

The following document serves as a proof of our work during the semester as well as an explanation on the program's operating principles and user manual. The GitLab repository containing the project's source code can be found *here*.

CONTENTS

1. REVIEW OF EXISTING SOLUTIONS

Image Segmentation is not a new concept and is widely used in various areas such as machine vision. It is also a topic of many research papers and coding projects. While conducting research for our project, we encountered many different similar solutions both in terms of overall instance segmentation as well as essential parts of our project, like dataset preparation or raw image, point cloud and stream filtration. This chapter covers some of the solutions that we either took inspiration from or that helped us understand crucial concepts.

1.1. DETR

DETR [1] (DEtection TRansformer) approaches object detection as a direct set prediction problem. It consists of a set-based global loss, which forces unique predictions via bipartite matching, and a Transformer encoder-decoder architecture. Given a fixed small set of learned object queries, DETR reasons about the relations of the objects and the global image context to directly output the final set of predictions in parallel. Due to this parallel nature, DETR is very fast and efficient.

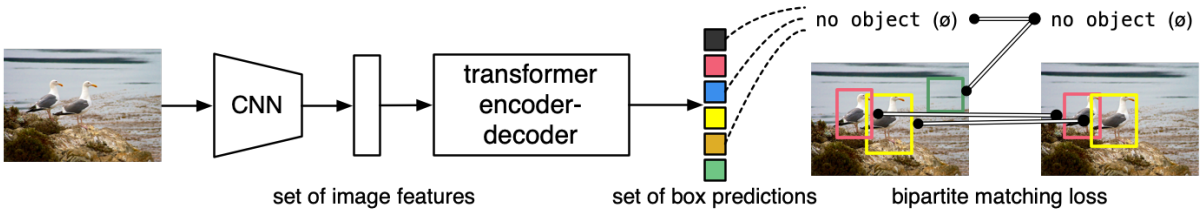

Figure 1.1: DETR's architecture as presented in the paper [1].

In the first stage (fig. 1.1), feature maps are extracted from the images via a Backbone layer. Many different models can be used, such as RestNet-50 or ResNet-101 [2]. In this way, 2-dimensional structure information is preserved. In the following stages, the data is flattened, so we get a 1-dimensional structure. After positional encoding, it is transferred to the encoder-decoder mechanism. Finally, each output is forwarded to the Feed Forward Network.

The last layer consists of 3 nodes. The normalized center coordinates of the predicted object and the predicted height and width values of the object are obtained where the Relu activation function is used. The class of the relevant object is predicted where the softmax activation function is used in the node. Thus, there is no need for Non-Maximum Suppression (NMS).

In the Positional Encoding section, the vector of each element (or token in NLP world) is re-created according to its place in the array. Thus, the same word can have different vectors at different positions in the array. In the encoder layer, the reduction of high-dimensional feature matrix to lower-dimensional feature matrices is performed. It consists of multi-head self attention, normalizer and feed forward network modules in each encoder layer. In the decoder layer, there are multi-head self attention, normalizer and feed forward network modules just like the encoder. N number of object queries are converted as output embedding. In the next stage, final estimation processes are carried out with the feed forward network.

1.2. OneFormer

OneFormer [3] is a universal image segmentation framework jointly trained on the panoptic, semantic, and instance segmentation. It outperforms existing state-of-the-arts on all three image segmentation tasks, by only training once on one panoptic dataset, instead of having to be trained separately for each task. Since 2023, it is available as a part of HuggingFace's *transformers* [4] library.

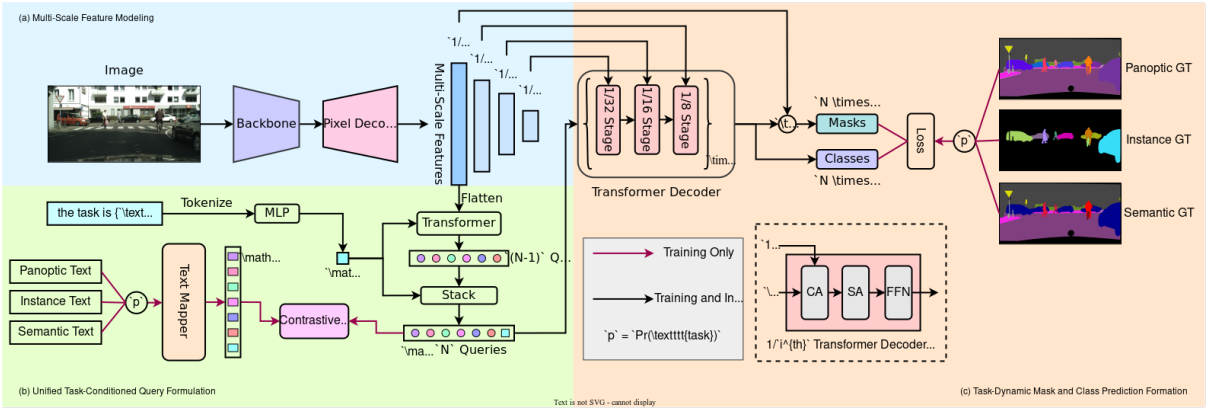

Figure 1.2: OneFormer's framework as presented in the paper [3]. The framework was implemented using Detectron2 [5] library.

The authors prepared a large number of pre-trained models with different backbones as well as trained and validated on different datasets, all of which can be downloaded from the project's github repository [6]:

- Backbones - the available models utilise commonly used backbones pretrained on the ImageNet-22K [7] to extract multi-scale feature representations from the input image. Then, a pixel decoder with a *Multi-Scale Deformable Transformer* [8, 9] based architecture aids the feature modeling by gradually up-sampling the backbone features. The Swin-Transformer [10], ConvNeXt [11], and DiNAT [12] backbones were used for the experiments.

- Datasets - the models were trained with a batch size of 16, on three popular datasets that support the three aforementioned segmentation tasks: semantic, instance and panoptic. The datasets used are ADE20K [13, 14], CityScapes [15], and COCO [16].

1.3. ISBNet

ISBnet [17] is a neural network model developed by VinAI Research, designed to tackle the task of instance segmentation, which involves identifying and delineating specific objects within an image. It leverages advanced deep learning techniques to accurately and efficiently segment objects, making it a valuable tool for computer vision applications. To address the limitations of DyCo3D [18], researchers proposed ISBNet, a cluster-free framework for 3DIS with Instance-aware Farthest Point Sampling and Box-aware Dynamic Convolution.

An auxiliary branch of the model jointly predicts the axis-aligned bounding box and the binary mask of each instance. The ground-truth axis-aligned bounding box is deduced from the existing instance mask label.

Given a point cloud, a 3D backbone is employed to extract per-point features (fig. 1.3). For DyCo3D, it first groups points into clusters based on the predicted object centroid from each point to generate a kernel for each cluster. In the meantime, the mask head transforms the per-point features into mask features for

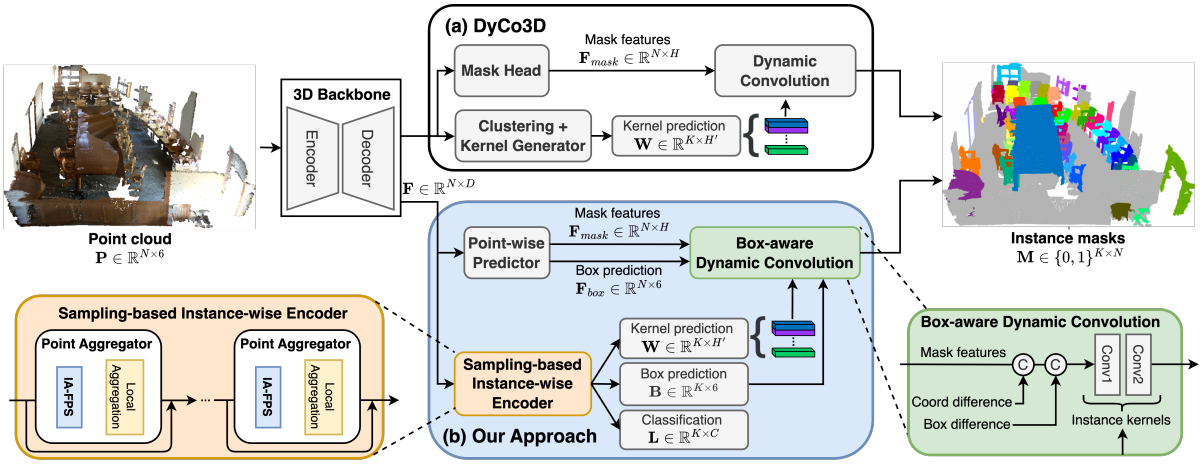

Figure 1.3: Architectures of DyCo3D (block a) and ISBNet's approach (block b), as presented in the paper [17].

dynamic convolution. In ISBNet, the clustering algorithm is replaced with a novel sampling-based instance-wise encoder to obtain faster and more robust kernel, box, and class predictions. Furthermore, a point-wise predictor replaces the mask head of DyCo3D to output the mask and box features for a new box-aware dynamic convolution to produce more accurate instance masks.

ISBNet not only achieves the highest accuracy among these three datasets, surpassing the strongest method by +2.7/2.4/3.0 on Scan-NetV2 [19], S3DIS [20], and STPLS3D [21], but also demonstrates to be highly efficient, running at 237ms per scene on ScanNetV2.

1.4. Amodal3Det

Amodal3Det [22] is a neural network model available on GitHub and developed by the Phoenix Neural Networks team. It focuses on the challenging task of amodal 3D object detection, which involves detecting and localizing objects in 3D space, even when they are partially occluded or invisible. This model incorporates state-of-the-art techniques in deep learning and computer vision to address the complex problem of accurately detecting objects in challenging scenarios. The architecture of the Amodal3Det network is based on state-of-the-art techniques in deep learning and computer vision.

The labeled NYU Depth V2 [23] dataset is one of the most popular but very challenging dataset in the RGBD scene understanding research community. The original version provides 1449 RGB-Depth indoor scene images with dense 2D pixel wise class labels. To enrich the labeling features and encourage 3D object detection research, in the SUN RGBD dataset [24] (superset of NYUV2) researchers added extra 3D bounding boxes and room layouts to ground truth annotations. Since depth maps are imperfect in reality due to measurement noise, light reflection and absorption, and occlusion etc, they also refined the quality of depth maps by integrating multiple RGB-D frames from the NYUV2 raw video data.

Given a pair of color and depth images, the goal of the amodal 3D object detection is to identify the object instance locations and its full extent in 3D space. Objects that were used were a bathtub, bed, bookshelf, box, chair, counter, desk, door, dresser, garbage bin, lamp, monitor, nightstand, pillow, sink, sofa, table, tv, toilet.

To train the Amodal3Det model, the Phoenix Neural Networks team utilized large-scale datasets specifically curated for amodal 3D object detection. These datasets would consist of 3D scenes with various objects in different poses, occlusion levels, and viewpoints. The annotations in the training data would include the 3D bounding box information for both visible and occluded parts of the objects, providing the model with the

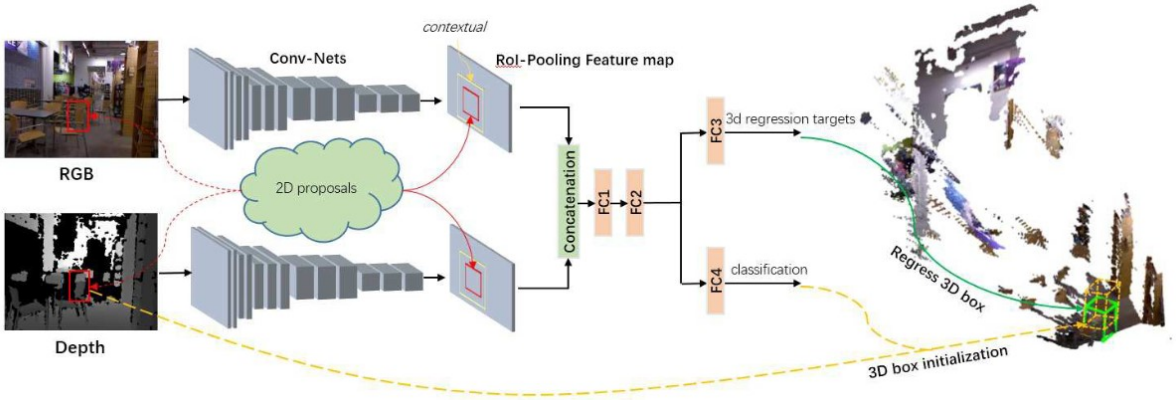

Figure 1.4: Overview of Amodal3Det's 3D object detection system. For each 2D segment proposal, firstly the localization of a 3D box is initialized (yellow dash box) based on depth information and its size according to class wise prior knowledge. Then object class and 3D regression offsets are jointly learned based on 2D features only, with the goal of obtaining the final 3D detection (green solid box) by adjusting the location, dimension, and orientation of the initial 3D box.

necessary supervision to learn the task effectively.

1.5. DeepLabV3

DeepLabV3 [25, 26] is a state-of-the-art deep learning model designed for semantic image segmentation. The underlying architecture of DeepLabV3 is based on a convolutional neural network (CNN) framework. The model consists of an encoder network, which extracts high-level features from the input image, and a decoder network, which generates dense pixel-level predictions.

The DeepLabV3 model takes an image as input. Following this action the input image is fed into a pre-trained CNN, such as ResNet [2] or MobileNet [27], which acts as the encoder network. Atrous Convolutions (dilated convolutions) are applied which increase the receptive field without reducing the spatial resolution. This allows the model to capture both local and global contextual information effectively. Next the Atrous Spatial Pyramid Pooling (ASPP) is used which involves applying atrous convolutions with different dilation rates in parallel to capture context at multiple scales. This enables the model to incorporate both fine-grained details and global context into its predictions. Then the decoder employs bilinear upsampling to increase the spatial resolution of the feature maps. Using skip connections, high-resolution future maps are combined from the encoder network with upsampled feature maps from the decoder network. This helps to preserve local information and enhance the segmentation accuracy. Finally the combined feature maps are further processed through additional convolutional layers to generate the final dense pixel-wise predictions. The output is a segmentation map, where each pixel is assigned a label corresponding to a specific object class.

While working with DeepLabV3 we decided to create our own dataset based on the ADE20K [14, 13], which would include objects photographed in the laboratory.

2. SOLUTION OVERVIEW

Our work on the project could be divided into 4 different aspects we focused on. The task required us to create our own dataset for neural network training, as well as to devise an algorithm calculating objects' bounding boxes. The program had to be then connected to the RGBD cameras and made into a user-friendly application. The following flowchart illustrates the working principle of the program:

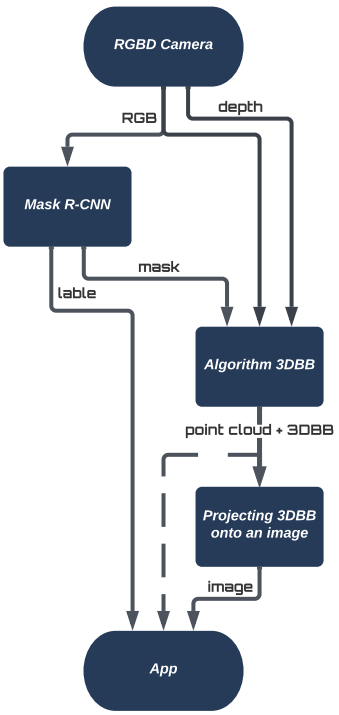

Figure 2.1: Flowchart showing the program's operating principle.

2.1. Dataset preparation

After considering the size of the photo box and the availability of certain everyday objects, we chose 7 different classes:

- mug
- box
- wine glass
- bowl
- shoe
- cap
- lamp

Our initial plans were to also include a *bottle* class, however after encountering several problems, mainly concerning the way the light rays passed through the bottles, we were forced to narrow it down.

2.1.1. Obtaining photo data

To obtain RGB and depth image data we used Intel's *RealSense Viewer* application, which is a part of Intel® RealSense™ SDK 2.0 [28]. We recorded short sequences and saved them as .bag files, from which we later extracted the RGB image and depth data in form of a point cloud.

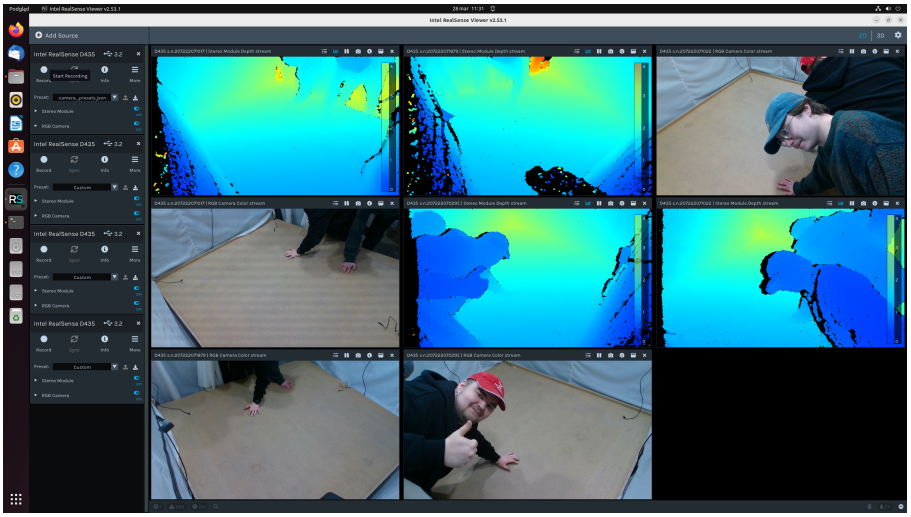

Figure 2.2: Intel's RealSense Viewer window showing capturing data from the cameras.

Preparing the data in the lab took about 15 hours. Overall we captured close to 900 images, with over 5000 single object occurrences.

2.1.2. Labelling photos

For labelling the instances we used a free online tool *labelbox.com* [29]. It enabled us to work simultaneously as a group and provided an easy way to apply masks to our images.

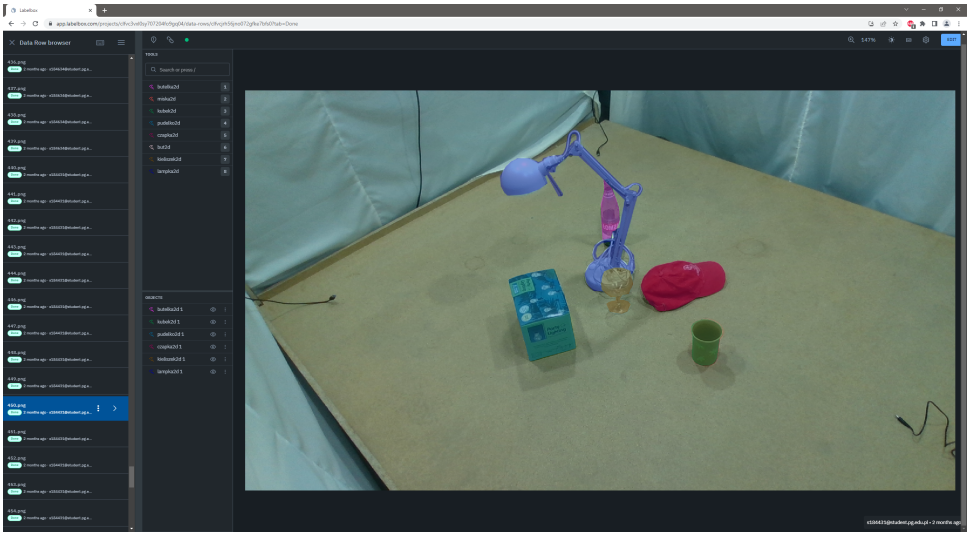

Figure 2.3: One of the images from the dataset annotated in labelbox.

Annotating, reviewing and reworking the images took 52 hours in total and was on of the most time-consuming tasks our team had to deal with.

2.2. Segmentation algorithm

Our Instance

2.2.1. MaskRCNN for object segmentation in 2D

We decided to use Mask R-CNN because it is a state-of-the-art Deep Convolutional Neural Network regarding image segmentation [30, 31]. This Neural Network detects objects in an image and generates a high-quality segmentation mask for each instance. It provides the vector of three essential factors for each object: instance mask, bounding box and label. Each of those elements is used in the following stages of image analysis (fig. 2.1).

In our solution, the network got fine-tuned [32] on the dataset we made. The version used is Mask R-CNN v2 with a ResNet-50-FPN backbone [33].

2.2.2. Integration with the RGBD cameras

We used the pyrealsense2 [34] library to get aligned RGB and depth frames from the cameras, and also to filter the depth image with temporal and spatial filtering. The RGB image is used as the input to MaskRCNN and the output masks are applied to the filtered depth image. This allows us to retrieve separate depth images of all the detected objects in the scene. These images are then used to create separate point clouds and draw bounding boxes around them using Open3D [35]. The bounding boxes' coordinates are extracted and used to project them onto the original RGB image.

2.2.3. 3D bounding boxes

After familiarising ourselves with the concept of 3D bounding boxes (*3DBB*) and the already existing methods of determining their placements [36], we have devised our own 3DBB creating method relying on the functionalities of Open3D python library [35] and processing of the point cloud. The first stage consists of extracting a part of the point cloud with our analysed object, based on its mask (fig. 2.4). To do that, we use Open3D, creating the object's bounding box (fig. 2.5).

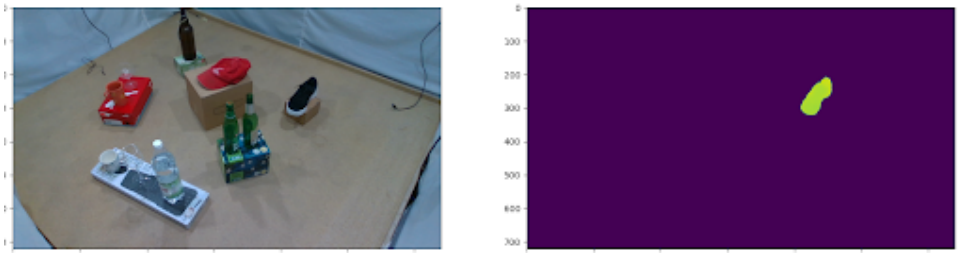

Figure 2.4: An example showing the RGB image and the mask of an object.

After that, by getting the positions of 3DBB's vertices in the 3D space, the vertices' coordinates are transformed and properly placed on the 2D image using the OpenCV python library [37]. The points are then connected by lines, comprising a 3D bounding box projected onto a two-dimensional plane (fig. 2.6.

A key piece of information used while projecting the 3DBBs is the depth data, which has to be taken into account with the loss of the Z dimension. It also allows us to determine the proportions of the boxes' edges to one another, which aids in the inspection of the projected points and automatic correction of their locations if any errors are found. The position of the 3DBB is checked in relation to the position of the object's mask, and

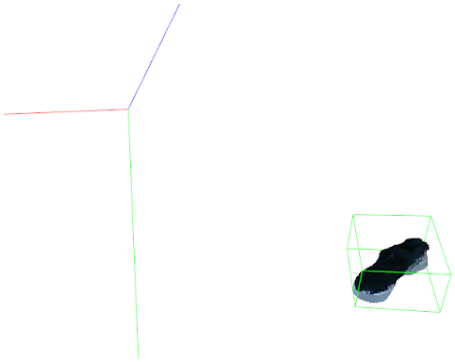

Figure 2.5: Fragment of the point cloud containing the previously shown object, with RGB image applied and a visible bounding box.

Figure 2.6: A fragment of an RGB image showing the projected bounding box for an object.

in case of too big of a deviation, the bounding box's dimensions are corrected.

Overall, we introduced a significant amount of parameters, which values were determined by trial and error, as well as variables which helped us control the correctness of the final coordinates of points on the 2D plane to properly replicate a 3D bounding box in a known environment and under known conditions. Of course, our approach is not perfect, and there are more accurate methods in existence. However, it does allow us to reconstruct an object's bounding box to a sufficient degree without using neural networks.

3. TESTS

We discussed training 3D object detection models using proprietary datasets in the previous chapters. Once the models have been trained, the next crucial step is testing their performance and evaluating their effectiveness. This chapter will explore the various aspects of testing 3D object detection models trained on proprietary datasets.

We need a representative test dataset to evaluate the performance of our trained models. This dataset should ideally cover various scenarios, including different environmental conditions, object types, and object orientations. When using a proprietary dataset, ensuring that the test dataset is distinct and independent from the training data is essential to provide an unbiased evaluation.

To achieve this, we can separate the proprietary dataset into three subsets: the training, validation and test sets. We used the commonly used proportions of 70 to 15 to 15 percentages. The training and validation sets are used exclusively during the training phase, while the test set is reserved for evaluating the model's performance. It is essential to ensure no overlap between the two sets to maintain the integrity of the evaluation.

3.1. Evaluation Metrics

We need suitable evaluation metrics to assess the performance of 3D object detection models. These metrics assess the accuracy and robustness of the model's predictions.
We used the following metrics for 3D object detection:

- **Intersection over Union (IoU):** IoU measures the overlap between the predicted and ground truth bounding boxes. It is computed by dividing the intersection volume by the union volume between the two boxes. This metric got used for three instances: 2D bounding boxes, instance masks and 3D bounding boxes.
- **Mean Average Precision (mAP):** mAP is the Average Precision value across different object classes. It provides an overall assessment of the model's performance in multiple categories.
- **Frames per second (FPS):** the frame rate at which consecutive images (frames) are captured or displayed. This metric is not typical for 3D object detection models, but in robotic environments or autonomous vehicles, the model's speed is crucial. Obtained performance depends on many factors, like the number of objects in the scene and, most importantly, the camera's resolution. Because of that, in the summary of the model's performance, we provided the results for two resolutions.

3.2. Testing Procedure

To conduct the testing process, we feed the test dataset into the trained 2D object detection model and evaluate its predictions against the ground truth annotations. The following steps outline the testing procedure:

1. **Input Preprocessing:** Similar to the training phase, the test dataset requires preprocessing. This involves transforming images and masks' associated annotations into a suitable format compatible with the model's input requirements.
2. **Model Inference:** The preprocessed test data is passed through the instance segmentation model, which generates predictions for object classification and localisation. Our model defines localisation by two parameters: bounding box and mask.

3. **Post-processing:** The raw predictions from the model often require post-processing steps to refine and filter the results. This may involve non-maximum suppression to remove redundant bounding boxes or score thresholding to filter out low-confidence predictions.

4. **Evaluation:** Once the post-processing is complete, we compare the predicted bounding boxes, masks and labels with the ground truth annotations. We compute the evaluation metrics discussed earlier, including IoU and mAP to assess the model's performance on the test dataset.

3.3. Final statistics

The final performance of the solution is:

1. Intersection over union 2D: 81.3%
2. Intersection over union 3D: 80.9%
3. Mean average precision: 89.8%
4. Frame rate:
    - 4 - 6 FPS for 1280x720 resolution
    - 10 - 14 FPS for 640x360 resolution

4. SUMMARY

This project presents a novel proprietary 3D object detection model that demonstrates exceptional performance in terms of accuracy and efficiency. We introduce a cutting-edge model to address existing approaches' limitations and advance 3D object detection. Modern 3D detection models are based on advanced deep convolution neural networks. Our architecture is a hybrid approach using both analytical algorithms and neural networks.

Documentation begins with an overview of state-of-the-art solutions. It shows many different approaches to 3D object detection and emphasises the need for improved accuracy in detecting objects in three-dimensional space while ensuring real-time processing capabilities.

The proprietary model utilises innovative deep learning architectures and optimised algorithms. We describe the unique architectural design, which incorporates convolutional neural networks and analytical algorithms to get performance as good as possible regarding frame rate and accuracy.

The *Instance Segmentation* project was a valuable learning experience for the entire team. While working on it, we had to face different problems, both of technical and personal nature. We had to deal with problems concerning changes in the team members, both in the beginning and middle of the project and with the lab equipment not working properly.

Despite the project team's hardship during its development, we are satisfied with the results. Ultimately, we delivered a working program that fulfils all initial assumptions and leaves room for further development. While we are content, we realise some aspects of our program could be improved. We would be glad to work on these improvements in the future.

5. FURTHER DEVELOPMENT POSSIBILITIES

While working on the project, we tried to implement some solutions that did not work out in the end, mainly due to the lack of time. However, those failed attempts gave us ideas about the possible changes and improvements we could make in the future to make our program more accurate.

5.1. Multiple RGBD cameras

Attempts were made to integrate two RealSense cameras to get the same image from different angles, providing a more accurate point cloud. We did not succeed due to the problems with camera calibration, however if we were to continue the project, we would probably reattempt to do that with up to four cameras. Getting the depth data from four different angles would provide us with a very detailed point cloud with very few blind spots and an almost complete 3D reconstruction of the captured scenery, which would produce more accurate results for the 3DBB algorithm.

5.2. Neural network for 3DBB estimation

While we did experiment with different neural network models, in the end, we decided to go with an analytic approach, mostly due to the fact that a lot of models were outdated or required older versions of software and libraries which were very hard to acquire. There is a chance using a neural network to produce bounding boxes for objects would have given us more accurate results, however due to time constraints, we were forced to abandon the idea. We do, however, plan to try to run models again by setting up virtual machines with the required versions of the operating systems, libraries, software etc.

5.3. Expanding the range of detectable objects

The dataset which we worked on was supposed to consist of 8 classes, but due to different issues, it ended up containing only 7. This is a really small number of classes compared to the datasets available on the web. As of now, our datasets consist of about 900 RGBD images and over 5000 annotations of these seven classes, but we do plan to acquire more data and introduce more classes in the future.

6. ENCOUNTERED PROBLEMS AND THEIR SOLUTIONS

Working on any project does not come without any problems. Some of the ones we encountered, we managed to resolve, some we were able to omit. This chapter provides an overview of the issues we had to face.

6.1. The 'bottle' class

Initially, our dataset was supposed to consist of 8 classes, one of them being the *bottle* class. Unfortunately we did not take into the account the fact that PVC bottles allow the light to pass through them. The depth data for bottles was inaccurate and it showed *'floating particles'* in places where the light passed through the plastic surface. This has made it very hard for our program to detect and calculate bounding boxes for this class, hence why we were forced to exclude this class from our dataset. However, we do plan to add other classes to our dataset to expand it and we might look into ways of ignoring those floating fragments of the depth cloud while creating bounding boxes for semi-transparent objects.

6.2. Multiple cameras calibration - poor cable quality

Unfortunately, due to poor cable quality in the lab, some of the cameras were recognised by the software as connected by USB 2.1, and the others were shown to have a proper USB 3.2 connection. Since USB 2.1 provides worse image quality, it was an issue when trying to calibrate multiple cameras so that they could work together. While we didn't succeed in doing so thins semester, we will look into solving that problem in the future.

6.3. Starting up neural network models

While we tried to launch several different neural network models we found, we did not achieve complete success with any of them. With how quickly everything develops, lots of the models turned out to require older, outdated versions of various libraries and softawres, which we were not able to launch or acquire. While this problem remains unsolved for now, we plan to set up virtual machine environments and try to run these models again.

BIBLIOGRAPHY

[1] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," *CoRR*, vol. abs/2005.12872, 2020. [Online]. Available: https://arxiv.org/abs/2005.12872

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[3] J. Jain, J. Li, M. Chiu, A. Hassani, N. Orlov, and H. Shi, "OneFormer: One Transformer to Rule Universal Image Segmentation," *CVPR*, 2023.

[4] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-Art Natural Language Processing." Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6

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

[6] J. Jain, J. Li, M. Chiu, A. Hassani, N. Orlov, and H. Shi, "OneFormer: One Transformer to Rule Universal Image Segmentation [github repository]," 2023. [Online]. Available: https://github.com/SHI-Labs/OneFormer

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[8] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[9] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention mask transformer for universal image segmentation," 2022.

[10] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[11] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[12] A. Hassani and H. Shi, "Dilated neighborhood attention transformer," 2022. [Online]. Available: https://arxiv.org/abs/2209.15001

[13] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ade20k dataset," *International Journal of Computer Vision*, vol. 127, no. 3, pp. 302–321, 2019.

[14] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Scene parsing through ade20k dataset,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[15] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[16] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: common objects in context,” *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[17] T. D. Ngo, B.-S. Hua, and K. Nguyen, “Isbnet: a 3d point cloud instance segmentation network with instance-aware sampling and box-aware dynamic convolution,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[18] T. He, C. Shen, and A. van den Hengel, “DyCo3d: Robust instance segmentation of 3d point clouds through dynamic convolution,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[19] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, “Scannet: Richly-annotated 3d reconstructions of indoor scenes,” in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[20] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese, “Joint 2D-3D-Semantic Data for Indoor Scene Understanding,” *ArXiv e-prints*, Feb. 2017.

[21] M. Chen, Q. Hu, Z. Yu, H. THOMAS, A. Feng, Y. Hou, K. McCullough, F. Ren, and L. Soibelman, “Stpls3d: A large-scale synthetic and real aerial photogrammetry 3d point cloud dataset,” in *33rd British Machine Vision Conference 2022, BMVC 2022, London, UK, November 21-24, 2022*. BMVA Press, 2022. [Online]. Available: https://bmvc2022.mpi-inf.mpg.de/0429.pdf

[22] Z. Deng and L. J. Latecki, “Amodal detection of 3d objects: Inferring 3d bounding boxes from 2d ones in rgb-depth images,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 398–406.

[23] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, “Indoor segmentation and support inference from rgbd images,” in *ECCV*, 2012.

[24] S. Song, S. P. Lichtenberg, and J. Xiao, “Sun rgb-d: A rgb-d scene understanding benchmark suite,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2015, pp. 567–576. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298655

[25] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in *ECCV*, 2018.

[26] L. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking atrous convolution for semantic image segmentation,” *CoRR*, vol. abs/1706.05587, 2017. [Online]. Available: http://arxiv.org/abs/1706.05587

[27] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," in *ICCV*, 2019.

[28] "Intel® realsense™ sdk 2.0," https://github.com/IntelRealSense/librealsense, 2018.

[29] Labelbox, "Labelbox," 2023. [Online]. Available: https://labelbox.com

[30] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: http://arxiv.org/abs/1703.06870

[31] W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow," https://github.com/matterport/Mask_RCNN, 2017.

[32] Y. Shinya, E. Simo-Serra, and T. Suzuki, "Understanding the effects of pre-training for object detectors via eigenspectrum," *CoRR*, vol. abs/1909.04021, 2019. [Online]. Available: http://arxiv.org/abs/1909.04021

[33] Y. Li, S. Xie, X. Chen, P. Dollár, K. He, and R. B. Girshick, "Benchmarking detection transfer learning with vision transformers," *CoRR*, vol. abs/2111.11429, 2021. [Online]. Available: https://arxiv.org/abs/2111.11429

[34] I. R. Team, "pyrealsense2's documentation," https://intelrealsense.github.io/librealsense/python_docs/index.html, 2017.

[35] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[36] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3d bounding box estimation using deep learning and geometry," *CoRR*, vol. abs/1612.00496, 2016. [Online]. Available: http://arxiv.org/abs/1612.00496

[37] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[38] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "Shapenet: An information-rich 3d model repository," *CoRR*, vol. abs/1512.03012, 2015. [Online]. Available: http://arxiv.org/abs/1512.03012

[39] A. Ahmadyan, L. Zhang, A. Ablavatski, J. Wei, and M. Grundmann, "Objectron: A large scale dataset of object-centric videos in the wild with pose annotations," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021. [Online]. Available: https://arxiv.org/abs/2012.09988

[40] A. Ahmadyan, T. Hou, J. Wei, L. Zhang, A. Ablavatski, and M. Grundmann, "Instant 3d object tracking with applications in augmented reality," *CoRR*, vol. abs/2006.13194, 2020. [Online]. Available: https://arxiv.org/abs/2006.13194

[41] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas, and S. Savarese, "Objectnet3d: A large scale database for 3d object recognition," in *European Conference Computer Vision (ECCV)*, 2016.

[42] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3dmatch: Learning local geometric descriptors from rgb-d reconstructions," in *CVPR*, 2017.

[43] M. Asad, L. Fidon, and T. Vercauteren, "Econet: Efficient convolutional online likelihood network for scribble-based interactive segmentation," 2022.

[44] Y. Zhou and O. Tuzel, “Voxelnet: End-to-end learning for point cloud based 3d object detection,” *CoRR*, vol. abs/1711.06396, 2017. [Online]. Available: http://arxiv.org/abs/1711.06396

[45] S. Shi, X. Wang, and H. Li, “Pointrcnn: 3d object proposal generation and detection from point cloud,” *CoRR*, vol. abs/1812.04244, 2018. [Online]. Available: http://arxiv.org/abs/1812.04244

[46] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, “Frustum pointnets for 3d object detection from RGB-D data,” *CoRR*, vol. abs/1711.08488, 2017. [Online]. Available: http://arxiv.org/abs/1711.08488

[47] J. Lahoud and B. Ghanem, “2d-driven 3d object detection in rgb-d images,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4632–4640.

[48] S. Song and J. Xiao, “Deep sliding shapes for amodal 3d object detection in RGB-D images,” *CoRR*, vol. abs/1511.02300, 2015. [Online]. Available: http://arxiv.org/abs/1511.02300

[49] G. Han, Y. Zhu, L. Liao, H. Yao, Z. Zhao, and Q. Zheng, “Hybrid attention-based 3d object detection with differential point clouds,” *Electronics*, vol. 11, no. 23, 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/23/4010

[50] Y. Yan, Y. Mao, and B. Li, “Second: Sparsely embedded convolutional detection,” *Sensors*, vol. 18, no. 10, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/10/3337

[51] C. Xie, Y. Xiang, A. Mousavian, and D. Fox, “Unseen object instance segmentation for robotic environments,” *IEEE Transactions on Robotics (T-RO)*, vol. abs/2007.08073, 2021. [Online]. Available: https://arxiv.org/abs/2007.08073

[52] ——, “The best of both modes: Separately leveraging rgb and depth for unseen object instance segmentation,” in *Conference on Robot Learning (CoRL)*, vol. abs/1907.13236, 2019. [Online]. Available: http://arxiv.org/abs/1907.13236

[53] Y. Xiang, C. Xie, A. Mousavian, and D. Fox, “Learning RGB-D feature embeddings for unseen object instance segmentation,” *CoRR*, vol. abs/2007.15157, 2020. [Online]. Available: https://arxiv.org/abs/2007.15157

[54] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *International Journal of Robotics Research (IJRR)*, 2013.

[55] D. Stephane, “Voxelnet (tensorflow 2.0.0),” https://github.com/steph1793/Voxelnet, 2019.

[56] B. HE, “Voxelnet-pytorch,” https://github.com/skyhehe123/VoxelNet-pytorch, 2018.

[57] G. Jocher, A. Chaurasia, and J. Qiu, “YOLO by Ultralytics,” Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics